Nancy-Université
Université Henri Poincaré

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

➢ Contact SCD Nancy 1 : theses.sciences@scd.uhp-nancy.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

Nancy-Université
Université Henri Poincaré

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Résolution de contraintes de déductibilité. Application à la composition de services Web sécurisés

THÈSE

présentée et soutenue publiquement le 19 September, 2011

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

Tigran Avanesov

Composition du jury

Président :	Isabelle Chrisment	
Rapporteurs :	Ralf Küsters	Professeur, Université de Trèves, DE
	Yassine Lakhnech	Professeur, Université Joseph Fourier, Grenoble, FR
Examinateurs :	Yannick Chevalier	Maître de Conférences, Université de Toulouse III, FR
	Isabelle Chrisment	Professeur, Université Henri Poincaré, Nancy, FR
	Michaël Rusinowitch	Directeur de Recherche, INRIA, Nancy, FR Directeur de thèse
	Farouk Toumani	Professeur, Université Blaise Pascal, Clermont-Ferrand, FR
	Mathieu Turuani	Chargé de Recherche, INRIA, Nancy, FR
	Luca Viganò	Professeur, Université de Vérone, IT

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Acknowledgments

I would like to thank to everyone who helped me with this thesis:
to ones who know they did (my special thanks to CRT)
and to ones who do not.

Contents

Version française, abrégée **vii**

Préface

1 Modélisation xiii
2 Contributions xiv

Systèmes de contraintes de deductibilité

0.1 Dérivations et systèmes de contraintes xxi
0.2 L'existence d'une solution conservatrice xxix

Analyse de complexité

Composition de services Web

Analyse de protocoles cryptographiques

Conclusions

Preface **lix**

1 Motivation 1
2 Modeling 6
2.1 Cryptographic primitives 7
2.2 Operations on messages 9
2.3 Communicating parties behavior 9
3 Contributions 10
3.1 Solving deducibility constraints 10
3.2 Composition of Web Services 10
3.3 Verification of cryptographic protocols 11

4 Plan of the thesis 11
4.1 Deducibility constraint systems 11
4.2 Web Services composition 12
4.3 Cryptographic Protocols 12

Part I Deducibility constraint systems **15**

Chapter 1
Introduction

1.1 Related works 18
1.1.1 Well-formed constraints 18
1.1.2 General constraints 21
1.2 Terms and notions 22
1.2.1 Term algebra 22
1.2.2 Equational theories 23
1.2.3 Derivations and constraint systems 24

Chapter 2
Solving general deducibility constraint systems

2.1 Dolev-Yao constraints extended with ACI symbol 27
2.1.1 Formal introduction to the problem 28
2.1.2 Ground case of DY+ACI 40
2.1.3 Existence of conservative solutions 43
2.1.4 Bounds on conservative solutions 47
2.2 Pure Dolev-Yao constraints 50

Chapter 3
Complexity analysis

3.1 Measure of the input 55
3.2 Ground derivability in DY+ACI is in P 57
3.3 Satisfiability of general DY+ACI constraint systems is in NP 57
3.4 Undecidability of a subterm deduction system 59

Conclusions **63**

Part II Applications **65**

Chapter 4 Web Services composition **67**

4.1 Introduction . . . 68
4.1.1 Orchestration of Web Services . . . 69
4.1.2 Choreography of Web Services . . . 71
4.1.3 Distributed orchestration of Web Services . . . 72
4.1.4 Terms . . . 73
4.2 Motivation . . . 75
4.2.1 Orchestration example . . . 75
4.2.2 Distributed orchestration example . . . 78
4.3 Distributed orchestration model . . . 82
4.3.1 Distributed orchestration problem input . . . 83
4.3.2 Execution model . . . 83
4.3.3 Problem statement . . . 88
4.4 Solution approach . . . 88
4.4.1 Reduction to deducibility constraints . . . 89
4.4.2 Constraint system resolution . . . 89
4.5 AVANTSSAR Orchestrator . . . 89
4.5.1 AVANTSSAR Validation Platform . . . 90
4.5.2 Input Format . . . 91
4.5.3 Output Format . . . 96
4.5.4 Approach overview . . . 96
4.5.5 Architecture overview . . . 99
4.5.6 Security Loopback . . . 103
4.5.7 A toy example . . . 105
4.5.8 Some experimental results . . . 109
4.6 Conclusions . . . 112

Chapter 5 Verification of cryptographic protocols **115**

5.1 Introduction . . . 115
5.1.1 Attacker Models . . . 117
5.1.2 From protocol sessions to constraints solving . . . 119
5.2 Motivation . . . 123
5.3 Formal model . . . 124
5.4 Solution approach . . . 128
5.5 XML rewriting attacks . . . 129

5.5.1 Example of discovering an attack exploiting XML format 130
5.6 Conclusions 133

Conclusion **135**

Bibliography **141**

Version française, abrégée

Préface

Les systèmes de communication, qu'il s'agisse de ventes en ligne, d'infrastructure de téléphones portables ou de vidéo à la demande par la télévision, sont utilisés de façon ubiquitaire et il est difficile de surestimer l'importance de leur rôle dans notre vie. Il est absolument nécessaire de protéger ces systèmes d'utilisations illégales et d'attaques externes, et pour atteindre cet objectif la cryptographie offre une aide précieuse. Les messages transmis par dispositifs électroniques, ou leurs parties peuvent être cryptés (pour empêcher leur lecture), signé (pour fixer l'origine du message et donc, de prévenir l'usurpation d'identité) ou hachés (pour identifier un message sans le divulguer entièrement).

Par exemple, dans les systèmes d'achat en ligne, si l'on fait l'abstraction de la cryptographie, alors uniquement la transmission d'un message, dit "de charge utile" comme un numéro de la carte bancaire avec son code de sécurité et le nom de son détenteur, suffit à finaliser le paiement. Mais une telle transmission est vulnérable contre une simple attaque d'écoute, où les données qui devraient rester secrètes peuvent être apprises par un pirate, qui peut ensuite les réutiliser pour effectuer les paiements frauduleux.

La combinaison des primitives cryptographiques (comme chiffrement, signature numérique, nombres pseudo-aléatoires) et des messages de charge utile (comme, par exemple, le numéro d'une carte bancaire) utilisés par des participants à une communication dans un ordre fixe constitue un *protocole cryptographique*. Notez que si l'on considère chaque participant séparément des autres, on peut dire que l'utilisation de la cryptographie forme une *politique de sécurité* imposée sur sa communication.

Les protocoles cryptographiques sont largement déployés dans les communications sans fil, les systèmes de paiement, messageries électroniques et bien d'autres domaines. Malheureusement, le simple fait d'utiliser des primitives cryptographiques ne garantit pas les propriétés de sécurité souhaitées: par exemple, le fait qu'un message soit chiffré ne garantit pas, en général, qu'il ne puisse être lu par un attaquant. De même, si quelqu'un est capable de répondre à un défi d'authentification exigeant normalement la connaissance d'un secret, cela ne veut pas nécessairement dire qu'il connaisse véritablement ce secret.

Par exemple, on peut se référer à SIP, *Session Initiation Protocol* [RSC+02]. Il s'agit d'un protocole largement utilisé pour la signalisation de VoIP et permet de gérer les sessions des médias, comme des appels audio/vidéo (beaucoup de gens utilisent des services VoIP basés sur SIP pour passer des appels à bas prix à l'étranger). Si nous abstrayons des détails, pour faire un appel, un utilisateur enregistré envoie une requête `invite` à un serveur proxy SIP indiquant l'adresse SIP de la personne qu'il souhaite appeler et la sienne (par exemple, alice `invite` Bob). Afin de vérifier si l'utilisateur n'est pas un imposteur, SIP s'appuie sur un mécanisme d'authentification basé sur un secret partagé (mot de passe que l'utilisateur a entré lors de son enregistrement). Le proxy peut imposer à un défi d'authentification en envoyant une donnée aléatoire (*nonce*) à laquelle l'utilisateur doit renvoyer un hachage de la concaténation du son

requête, du nonce reçue et du mot de passe[1]. Le proxy peut également calculer cette valeur, car il connaît le mot de passe et le nonce qu'il a généré et la requête de l'utilisateur, et donc vérifier la réponse. Si la valeur calculée par le proxy coïncide avec celle reçue de l'utilisateur, le proxy considère que l'utilisateur connaît son mot de passe. Il semble que seul l'utilisateur connaissant le mot de passe peut faire un appel en utilisant son compte, pour lequel il sera donc facturé. Mais ce n'est pas le cas !

Le piège est que la même requête `invite` est utilisée, par exemple, pour mettre en attente une conversation déjà établie[2]. Ainsi, si B a une conversation avec A, et B veut mettre cet appel en attente, il doit envoyer la requête `invite` indiquant l'adresse SIP de A et la sienne. À ce moment, le proxy peut envoyer un défi d'authentification pour vérifier l'identité du demandeur, auquel B doit évidement répondre. Notez que le schéma d'authentification utilisé ici est le même que celui discuté précédemment. C'est sur ce point que l'on peut obtenir une réponse à la demande d'authentification envoyée par le proxy, quand on veut usurper l'identité de B.

La manière d'obliger B à mettre une conversation (avec A) en attente peut être facilement résolue en pratique. Il suffit d'appeler B pendant une conversation déjà mise en place. Dans ce cas il est très probable que B, ayant un autre appel sur la ligne, demandera A d'attendre et mettra la conversation en attente afin de répondre au nouvel appel.

Maintenant, nous pouvons décrire l'attaque où un intrus peut usurper l'identité de Bob, un utilisateur enregistré, et passer un appel en son nom à un utilisateur arbitraire Alice, utilisant le serveur proxy de la victime. De cette manière; le paiement de l'appel sera prélevé du compte de Bob.

Le synopsis est le suivant (voir Figure 1): un attaquant lancera un appel directement à la victime (Bob) au nom d'Alice. La victime répond à l'appel entrant et immédiatement reçoit un autre appel (aussi provoqué par l'attaquant). Pour y répondre, Bob met l'attaquant en attente. Une fois que l'attaquant reçoit une requête `invite` signifiant "en attente", il passe immédiatement un appel à Alice en utilisant le proxy de Bob en se faisant passer par Bob. Le proxy va essayer de l'authentifier en envoyant le défi correspondant. Ce défi sera transmis à la victime Bob pour authentifier (au nom du proxy) Bob, puisque celui-ci a voulu mettre en attente la conversation. Bob y répond et l'intrus réutilise cette réponse en la transmettant au proxy. De cette façon, il répond parfaitement au défi de proxy, qui autorise l'appel de l'intrus vers Alice au nom et à la charge de Bob.

On pourrait penser que si le SIP est un standard approuvé par l'IETF, largement utilisé, et employant la cryptographie, alors il ne doit pas avoir de problème de sécurité. Mais il est vulnérable à différentes attaques, comme celle [AARS08, AARS09] qui vient d'être présentée ici ou des attaques diverses par déni de service [GS07].

Des attaques plus sophistiquées peuvent se fonder sur les propriétés des algorithmes cryptographiques et des opérations qu'ils utilisent [RS98, PQ01], ou sur les schémas des message [BFGP04, CLR07]. Dans ce travail, nous présenterons également un moyen de modéliser des ensembles en rajoutant un élément supplémentaire de constructions des messages. Ceux-ci peuvent être particulièrement utiles lorsque l'on considère les *communications sécurisées de services Web*, car permettent de mieux modéliser les schémas XML que les services Web emploient systématiquement.

Les services Web sécurisés est un autre domaine qui sera exploré dans cette thèse. Les services Web sont conçus pour être interopérables et composables, c'est à dire qu'ils peuvent

[1] Dans la réalité, le message qui sera haché est un peu différent, mais l'idée est la même.

[2] Il est dénommé "re-invite" dans la spécification SIP, mais le message utilise exactement le même mot-clé "invite"

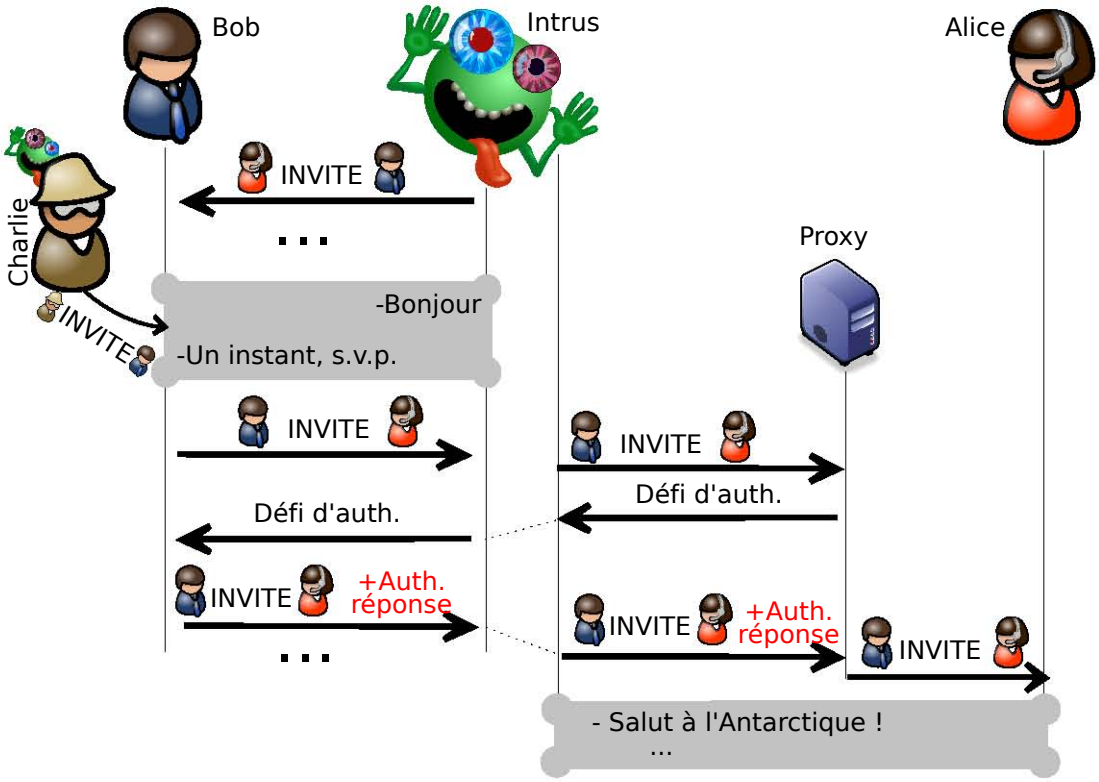


Figure 1: Attaque sur protocole SIP

coopérer et être combinés en services plus complexes (composition de services Web). Pour cela ils exposent une interface (généralement décrit dans un langage WSDL) qui devrait être utilisée pour invoquer les opérations proposées par les services.

Par exemple, supposons qu'il y ait deux services Web: l'un qui retourne une liste de tablettes PC disponibles sur le marché et répondant à des spécifications techniques minimales données, et l'autre qui, étant donné le nom d'un logiciel, retourne une configuration matérielle recommandée. Dans ce contexte, on peut imaginer leur composition: un nouveau service qui accepte le nom d'un logiciel et retourne la liste de Tablet PC de disponible sur le marché sur lesquels ce logiciel peut fonctionner correctement. Notez que ni le premier ni le deuxième service ne propose cette fonctionnalité.

Afin de garantir certaines propriétés de sécurité, comme dans le cas des protocoles, les *politiques de sécurité* sont imposées sur des services Web. Elles étendent l'interface du service Web imposant des exigences supplémentaires à l'entrée et à la sortie de chaque opération proposée par ce service Web. Par exemple, une opération de service bancaire qui reçoit un numéro de compte et un mot de passe, et retourne son état actuel ne peut accepter que des messages chiffrés à son entrée (afin de protéger le mot de passe) et n'émettre que des messages signés et chiffrés à la sortie (signé, en vue de certifier l'origine et lintégrité du message, et chiffré, afin de protéger l'état du compte de toute lecture non autorisée).

Ces restrictions, imposées par les politiques de sécurité, rendent complexe la composition automatique de services Web afin d'obtenir un comportement désiré tout en préservant les politiques de sécurité de chacun.

Il existe deux approches de base pour la composition des services Web: *chorégraphie* et *orchestration* [Pel03b]. La principale différence est liée à la manière des services Web de com-

muniquer entre eux. Dans le cas de la chorégraphie, des services Web communiquent directement les uns avec les autres en suivant un scénario global, tandis que dans le cas de l'orchestration, toutes les communications passent par une entité centrale, appelée *médiateur* (anglais: mediator) ou *orchestrateur* (anglais: orchestrator), qui distribue des requêtes aux services de la communauté et rassemble les réponses pour, éventuellement, réutiliser les données obtenues afin de construire les requêtes suivantes.

Nous nous concentrerons sur une combinaison de ces deux approches appelée *orchestration distribuée*, ou *décentralisée* [PE09, BMM06]. Là, le médiateur est distribué sur plusieurs nœuds, que nous appelons *partenaires*, chacun d'entre eux met en œuvre sa propre partie de la composition. Dans ce cadre une orchestration est un cas particulier dans lequel il n'existe qu'un seul partenaire, alors que la chorégraphie est un autre cas particulier dans lequel il existe un partenaire pour chaque service disponible.

Il y a beaucoup de travaux sur la composition automatique de services Web. La plupart d'entre eux s'abstraient de la structure des messages et considèrent uniquement des messages dans un certain ensemble fini, ou se placent au niveau des opérations (par exemple [BFHS03, BCF08, Pat09, Pad08, CGL+08]). Le modèle COLOMBO [BCD+05] permettant de travailler avec des messages d'un domaine infini et de considérer la structure des messages admet une procédure de composition automatique seulement pour un certain fragment restreint. En outre, la cryptographie n'est modélisée dans aucun des travaux cités ci-dessus.

On peut également mentionner un travail comme [BCD+09], où la cryptographie est utilisée pour faire des compositions sécurisées, mais n'a pas considéré le problème de la composition automatique lui-même; et un travail comme [MFP11], où les opérations à effectuer sur les messages sont déjà données à l'entrée du problème sous une forme de *contrat de sécurité*.

Dans notre approche, nous considérons une structure de message riche qui permet d'exprimer les politiques de sécurité appliquées sur les messages d'entrée et de sortie des opérations de services Web. Cela fait une contrainte supplémentaire lorsque l'on essaie d'appeler un service, puisque l'on doit satisfaire la politique de sécurité correspondante.

De plus, l'approche proposée permet une mise en œuvre "facile" pour le cas de l'orchestration, car il admet la réutilisation des outils existants pour la vérification de protocoles cryptographiques.

En résumé, dans ce travail nous allons examiner deux contextes de la cryptographie: la sécurité des protocoles cryptographiques et la composition des services Web sécurisés.

Comme technique sous-jacente, nous utilisons ce qu'on appelle *contraintes de déductibilité* qui expriment la possibilité de dériver un message à partir d'une connaissance donnée. Cette technique a été bien étudié dans le cadre de l'analyse de protocoles cryptographiques avec un nombre borné de sessions (par exemple, [MS01, CLC03, Shm04]). À noter qu'il y a plusieurs outils et techniques qui traitent du problème de vérification de protocoles cryptographiques avec un nombre borné de sessions (par exemple, [BMV05b, ACC07, Tur06, BHK05]). Par opposition, il existe une autre branche de recherche qui considère ce qu'on appelle *vérification non-bornée*, où le nombre de sessions du protocole n'est pas limité (par exemple [Cre08, BLP03, Bla01, EMM09]), mais nous nous concentrons sur le premier.

Les travaux mentionnés sur les contraintes de deductibilité ne considèrent que des contraintes sous une forme spéciale, appelées *bien formées*. Pour l'analyse classique des protocoles avec un nombre borné de sessions (avec un seul intrus de Dolev-Yao) cette hypothèse n'est pas restrictive, car le système de contraintes exprimant la possibilité d'une attaque sur un protocole est toujours bien formé. Cependant, si nous voulons envisager un autre modèle où des intrus multiples, chacun avec une zone d'influence locale, attaquent un protocole et n'ont aucun moyen de communiquer pendant l'attaque, les contraintes bien formées ne suffiront pas.

Dans ce travail, nous allons relâcher cette hypothèse et donner une procédure de décision

pour systèmes *généraux* de contraintes.

Notez que la situation dans le cas des protocoles ressemble à celle de la composition des services Web sécurisés: pour le cas de l'orchestration, les contraintes bien formées sont suffisantes. Mais pour le cas plus général, l'orchestration distribuée, l'hypothèse doit être relâchée.

Dans le reste de cette partie introductive nous allons d'abord décrire brièvement les choix de modélisation; puis présenterons en bref les contributions de la thèse.

1 Modélisation

Si l'on veut vérifier si une communication spécifiée est sécurisée, alors il existe principalement deux approches: *computationnelle* et *symbolique*. Le premier suppose que les messages sont des chaînes de bits et que l'intrus est une machine de Turing probabiliste arbitraires fonctionnant en temps polynomial. En général, cette approche est plus proche de la réalité, mais plus difficile à traiter, en particulier automatiquement. En revanche, l'approche symbolique s'abstrait de la représentation de chaîne de bits de messages et les considère comme des termes dans une certaine algèbre (avec éventuellement une théorie équationnelle pour refléter les propriétés de certaines opérations du monde réel). L'intrus, dans ce cas, est représenté par un ensemble de transformations abstraits qu'il peut appliquer sur les messages. Dans ce cas, les preuves de l'(in)sécurité sont plus faciles à automatiser, mais néanmoins, le problème de l'insécurité de protocoles est en général indécidable [EG83].

Pour la composition des services Web tenant compte de politiques de sécurité, il suffit de considérer seulement l'abstraction symbolique, car nous ne voulons pas que le médiateur casse le cryptage de certains messages dans le seul but d'adapter la communication.

Ainsi, dans cette thèse nous allons nous concentrer sur l'approche symbolique.

Nous allons considérer des messages construits en utilisant le cryptage symétrique et asymétrique, le hachage, la signature numérique et la concaténation (l'enchaînement). Nous considérons un ensemble tout à fait standard et naturel d'opérations abstraites qu'on peut effectuer sur les messages (règles de déduction), et qui fait normalement référence à un modèle de Dolev-Yao. C'est-à-dire,

- Un message peut être chiffré à l'aide d'un algorithme symétrique ou asymétrique si la clé et le message sont connus;
- Un message crypté à l'aide d'un algorithme symétrique (resp. asymétrique) peut être décrypté sous condition que la clé de cryptage (resp. clé privée correspondante) soit connue;
- Deux messages peuvent être concaténées;
- Un message composé par concaténation peut être divisé en ses parties;
- Un message peut être signé si une clé privée est connue (et la signature peut être vérifiée avec la clef publique correspondante);
- Un message peut être haché avec une fonction de hachage, si la fonction de hachage est connue.

Nous allons également considérer un constructeur d'ensembles:

- Les messages peuvent être groupés sous la forme d'un ensemble (on peut construire un ensemble de messages que l'on connaît), et

- D'un ensemble de messages, on peut extraire chaque élément.

Ces règles seront formalisés ultérieurement (voir, par exemple, les Tableaux 1 et 4). Les pièces de base des messages (par exemple, nom de l'agent ou sa clé publique) seront présentés sous forme de données atomiques. En utilisant les opérations ci-dessus, les messages plus complexes peuvent être construits. Par exemple, à partir de messages atomiques a, b, c on peut construire un message où b concaténé avec c et le résultat crypté par une clé symétrique obtenu par une concaténation de a et c. Ce message sera représenté dans notre formalisme comme $\mathrm{enc}\left(\mathrm{pair}\left(b, c\right), \mathrm{pair}\left(a, c\right)\right)$.

Si nous sommes capables de calculer $\mathrm{pair}\left(a, c\right)$ (par exemple, à partir des atomes a et c), alors à partir du message $\mathrm{enc}\left(\mathrm{pair}\left(b, c\right), \mathrm{pair}\left(a, c\right)\right)$, nous pouvons extraire l'atome b en utilisant les règles de déduction.

Comportement des agents communicants Le comportement des parties communicantes que nous considérons dans ce travail est "linéaire". Plus précisément, nous considérons un *séquence* de patterns de messages que l'on reçoit et renvoie.

Ce comportement est modélisé en utilisant des *strands* et sera formalisé sur la page xxxviii.

2 Contributions

En bref, nous présentons une technique pour traiter les primitives cryptographiques et les règles de déduction Dolev-Yao et montrons ce que nous pouvons faire en l'utilisant. Du point de vue des applications, dans ce travail nous avons l'intention de faire avancer l'état de l'art sur la composition des services Web en tenant compte de leurs politiques de sécurité, ainsi que celui de la vérification de protocoles cryptographiques; et d'un point de vue purement théorique, celui de la résolution de systèmes de contraintes déductibilité.

Résolution de contraintes de déductibilité Nous présentons une procédure de décision NP pour résoudre des systèmes généraux de contraintes de déductibilité dans le cas des règles de déduction de Dolev-Yao, et qui peuvent être éventuellement étendues avec les règles concernant un symbole associatif commutatif idempotente (ACI).

La seule restriction que nous avons est l'utilisation de clés atomiques pour la chiffrement asymétrique et la signature. Par contre notre modèle pour le chiffrement symétrique admet des clés complexes.

Le travail le plus proche à notre connaissance est la thèse de L. Mazaré [Maz06] où la propriété de contraintes d'être bien-formées a également été détendu, mais son modèle n'admettait pas de clés complexes. De plus, aucune théorie équationnelle n'a été examinée.

Par ailleurs, à notre connaissance, il n'y a pas de publications sur la satisfiabilité de contraintes bien formées supposant la théorie équationnelle ACI.

Composition de services Web Nous présentons un modèle pour l'orchestration distribuée de services Web sous les contraintes de leurs politiques de sécurité et *la condition de non-divulgation*.

Le comportement des services Web est représenté comme une séquence d'actions (recevoir/envoyer) qu'ils exécutent. Chaque action est annotée avec un pattern de message (un terme du premier ordre). Les actions d'un service Web sont reliées par les variables utilisées dans les patterns, c'est-à-dire que dès qu'une variable est instanciée, sa valeur sera utilisée dans les actions suivantes. De plus, les patterns de message admettent les primitives de sécurité qui permettent d'exprimer des politiques de sécurité de service Web.

Étant donnés un client spécifié comme un service Web et un ensemble de services disponibles, le problème est de construire un ensemble de médiateurs communiquants déployés dans les organisations *partenaires* qui soient en mesure de satisfaire les demandes du client. Pour ce faire, les médiateurs sont en mesure d'invoquer des services disponibles, d'appliquer les opérations de Dolev-Yao sur les messages recueillis et de communiquer entre eux. Cette communication est restreinte par les patterns de messages autorisés et la condition de non-divulgation qui interdit d'envoyer des données sensibles à d'autres partenaires.

À notre connaissance c'est le premier modèle distribué de composition de services Web qui tient compte de la structure de message et des politiques de sécurité, et qui admet une procédure de décision automatique.

Une application non-distribuée d'orchestration de services Web ne nécessitant pas de nouvelles techniques d'analyse sous-jacentes, mais toujours original et intéressant (l'automatisation de l'approche présentée dans [CMR08, CMR09]), a été mise en œuvre dans le cadre d'AVANTSSAR et validée sur plusieurs études de cas industrielles.

Vérification de protocoles cryptographiques Nous proposons un nouveau modèle d'intrus qui suppose plusieurs intrus non-communicants. Dans ce cadre nous proposons une procédure de décision pour le problème de l'insécurité de protocole (pour le cas du secret) avec le nombre borné de sessions.

Alors que l'insécurité d'une session de protocole dans notre modèle implique l'insécurité dans le modèle d'attaquant de Dolev-Yao, il peut être utile de considérer des modèles plus faibles, car l'intrus de Dolev-Yao peut être trop puissant dans les scénarios où certaines restrictions peuvent être fournies par les voies physiques, techniques ou administratives.

Nous montrerons également comment modéliser certaines attaques basées sur la présentation XML de messages grâce à l'opérateur ACI qui modélise bien des ensembles de noeuds XML.

Systèmes de contraintes de deductibilité

Introduction

Dans ce travail nous considérons un type spécial de contraintes qui permet d'exprimer la possibilité d'engendrer un message (correspondant à un pattern donné) à partir d'un ensemble de messages (dont les patterns sont aussi donnés). Ce type de contraintes s'appelle des *contraintes de déductibilité*:

Par exemple, un système de contraintes de deductibilité

$$\left\{\begin{array}{c} \{q, k, \mathrm{aenc}\,(u, k)\} \triangleright \mathrm{aenc}\,(X, k) \\ \{k, \mathrm{priv}\,(k)\,, \mathrm{aenc}\,(X, k)\} \triangleright q \end{array}\right\}$$

se comporte de deux contraintes (ici, une par ligne) et contient une seule variable X (q, k, u sont des valeurs atomiques).

Ce système est satisfiable, s'il existe une valeur $X\sigma$ (qui est représenté par un message) de la variable X telle que le message $\mathrm{aenc}\,(X\sigma, k)$ soit dérivable à partir d'un ensemble de messages $\{q, k, \mathrm{aenc}\,(u, k)\}$ et que le message q soit dérivable à partir d'un ensemble de messages $\{k, \mathrm{priv}\,(k)\,, \mathrm{aenc}\,(X\sigma, k)\}$ en utilisant les règles de déduction données (par exemple celles de Dolev-Yao mentionnées dans la Préface).

Travaux liés

La résolution des systèmes de contraintes de déductibilité est une branche de la recherche sur l'analyse de protocoles cryptographiques avec le nombre borné de sessions. La majorité écrasante de travaux considère ce que l'on appelle *les contraintes bien formées*, un sous-ensemble de systèmes de contraintes qui satisfait deux hypothèses: la *monotonie de connaissance*[3] et *l'origine de variables*[4]. En bref, la première hypothèse est en raison de la considération d'une entité communiquante centrale, l'intrus, qui n'oublie jamais la connaissance recueillie; et la deuxième est en raison du comportement déterministe d'autres participants dans le sens de la production de messages : dès que les entrées sont fixé jusqu'à une action "envoyer", alors le message suivant pour envoyer est aussi fixé.

Un peu plus formellement, la monotonie de connaissance dit qu'il existe un ordre $\leq$ sur les contraintes du système tel que si pour deux contraintes on a $(E \triangleright p) \leq (K \triangleright q)$ alors on a $E \subseteq K$. L'origine de variables dit qu'en utilisant le même ordre $\leq$, si dans une certaine contrainte $K \triangleright q$,

[3] Anglais: knowledge monotonicity
[4] Anglais: variable origination

une certaine variable X apparaît comme un sous-terme de la connaissance K, alors elle devra apparaître aussi comme un sous-terme d'un certain p tel que $(E \triangleright p) \leq (K \triangleright q)$.

Les règles de déduction basées sur le système de déduction Dolev-Yao sont très intéressantes comme ils permettent de raisonner sur l'insécurité de protocoles cryptographiques avec les primitives cryptographiques standard. Le premier travail qui a introduit une notion de contraintes est [MS01] (voir aussi [CV01]) et il considère les règles de déduction DY qui ressemblent à celle qui nous considérons dans le Tableau 4 (voir la page xxxii). Les auteurs ont montré que la propriété d'être bien formé pour les systèmes de contrainte a toujours lieu dans le cas où le système est construit pour résoudre le problème d'insécurité de protocole cryptographique en présence d'un intrus Dolev-Yao. La satisfiabilité des contraintes bien formées a été prouvée décidable, et dans [RT01] il est prouvé qu'elle est NP-complète.

Après, enrichissant le modèle standard de Dolev-Yao [DY83] avec différentes théories équationnelles comme le OU exclusif (XOR) [CKRT03a, CLC03, DLLT06], l'exponentiation modulaire et les groupes abéliens [CKRT03b, MS03, Shm04, Del06b], etc. [CKRT05, CKRT04, DLLT08] et leurs combinaisons [BMV05a, CR10b] les chercheurs obtinrent de meilleures approximations des propriétés des systèmes réels et alors devinrent capable de trouver des attaques qui n'eurent pas pu être découvertes en ne considérant que des symboles libres (par exemple, [RS98]). Un bon parcours peut être retrouvé dans [CDL06].

Cependant il n'y a que peu de travaux qui relâchent l'hypothèse de contraintes bien formées.

Une tentative de s'écarter des contraintes bien formées en considérant le système de déduction Dolev-Yao a été faite par L.Mazaré d'abord dans [Maz05]. Il a *partiellement* relâché la propriété de monotonie des les connaissances et appelé cette sorte de contraintes *quasi bien-formées* et a réclamé le NP-complétude du problème de la satisfiabilité pour ces contraintes. Si nous exprimions cette condition pour un système de contraintes $\{E_i \triangleright t_i\}_{i=1,\dots,n}$ en utilisant notre notation, alors ce serait $x \in \mathrm{Vars}(E_i) \implies \exists j < i : x \in \mathrm{Vars}(t_j) \wedge E_j \subsetneq E_i$.

Plus tard, dans sa thèse [Maz06], il a réussi à éviter l'usage de propriétés de contraintes d'être bien-formées et il a considéré les systèmes de contraintes *généraux*. Il a présenté une procédure de décision pour la satisfiabilité de tels systèmes, mais avec une restriction : les clés utilisées pour le cryptage doivent être toujours atomiques. Nous relâcherons cette restriction et montrerons que la satisfiabilité de contraintes générales Dolev-Yao est décidable et NP-difficile (voir aussi [ACRT11] et [ACRTar]). En plus, nous étendrons ce résultat en considérant la théorie équationnelle ACI.

Un autre travail à mentionner à propos des systèmes de contraintes généraux est [BCLD07], où les auteurs montre indécidabilité de tels systèmes si la signature ne contient qu'un seul symbole fonctionnel (à part des atomes), une seule règle de déduction (possibilité d'appliquer ce symbole) et la théorie équationnelle qui rende ce symbole associatif et commutatif (AC).

Termes et notions

Algèbre de termes

Nous donnerons quelques définitions liées à l'algèbre de termes (dans l'esprit de, par exemple, [Ohl02]).

Définition 0.0.1 La *signature* est un ensemble au plus dénombrable $\mathcal{F}$ de *symboles fonctionnels*, où chaque $f \in \mathcal{F}$ est associé à un nombre naturel positif appelé *arité* (le nombre d'arguments qu'il doit avoir), c'est-à-dire arité : $\mathcal{F} \mapsto \mathbb{N}^+$.

Définition 0.0.2 Soit $\mathcal{F}$ une signature, $\mathcal{X}$ un ensemble au plus dénombrable de *variables* et $\mathcal{A}$ un ensemble au plus dénombrable d'*atomes*[5]. L'ensemble de *termes* $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ est défini pour être le plus petit ensemble tel que

- $\mathcal{A} \cup \mathcal{X} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$.
- $\forall f \in \mathcal{F}$ if $n = arity(f)$ et $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ alors $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$.

Pour la simplicité, nous allons écrire $\mathcal{T}$ au lieu de $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$.

Définition 0.0.3 Pour $t \in \mathcal{T}$, le root (t) dénote le symbole de racine de t et il est définit comme:

$$\text{root}\,(t) = \begin{cases} f, & \text{si } t = f(t_1, \ldots, t_n), \text{ où } n = arity(f), \\ t, & \text{si } t \in \mathcal{X} \cup \mathcal{A}. \end{cases}$$

Définition 0.0.4 Soit t un terme. On définit un ensemble de ces *soustermes* $\text{Sub}\,(t)$ comme suit:

$$\text{Sub}\,(t) = \begin{cases} \{t\}, & \text{if } t \in \mathcal{X} \cup \mathcal{A}; \\ \{t\} \bigcup\limits_{i=1,\ldots,n} \text{Sub}\,(t_i), & \text{si } t = f(t_1, \ldots, t_n), \text{ où } n = arity(f). \end{cases}$$

Définition 0.0.5 Soit t un terme. Nous définissons l'ensemble de ses variables $\text{Vars}\,(t)$ comme $\text{Vars}\,(t) = \mathcal{X} \cap \text{Sub}\,(t)$. Terme m qui ne contient pas de variables, c'est-à-dire tel que $\text{Vars}\,(m) = \varnothing$, est appelé *clos*. L'ensemble de tous les termes clos est dénoté comme $\mathcal{T}_g$.

Définition 0.0.6 Une *substitution* σ est un mappage de $\mathcal{X}$ vers $\mathcal{T}$avec le *domaine* $\text{dom}\,(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$. Nous représentons une substitution σ comme $\{x \mapsto \sigma(x) \mid x \in \text{dom}\,(\sigma)\}$. Chaque substitution σ s'étend uniquement à la fonction $\sigma : \mathcal{T} \mapsto \mathcal{T}$, tel que

$$\sigma(t) = \begin{cases} t, & \text{if } t \in \mathcal{A}; \\ \sigma(x), & \text{if } t = x \in \mathcal{X}; \\ f(\sigma(t_1), \ldots, \sigma(t_n)), & \text{si } t = f(t_1, \ldots, t_n), \text{ où } n = arity(f). \end{cases}$$

Nous ne considérons que des substitutions σ telles que pour chaque $x \in \text{dom}\,(\sigma)$, $x \notin \text{Vars}\,(\sigma(x))$. Souvent nous utiliserons une version "postfix" $t\sigma$ de la notation au lieu de $\sigma(t)$. Nous appelons une substitution σ *close* si pour tout $x \in \text{dom}\,(\sigma)$, le terme $x\sigma$ est clos.

Exemple 1 Supposons que nous ayons trois symboles fonctionnels: f, g et h avec arités correspondantes $1, 2$ et 2, i.e. $\mathcal{F} = \{f, g, h\}$ and $arity(f) = 1$, $arity(g) = arity(h) = 2$. Supposons un ensemble d'atomes $\mathcal{A} = \{a, b, c\}$ et un ensemble de variables $\mathcal{X} = \{x, y, z\}$. Alors par exemple le terme t suivant est un élement de $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$:

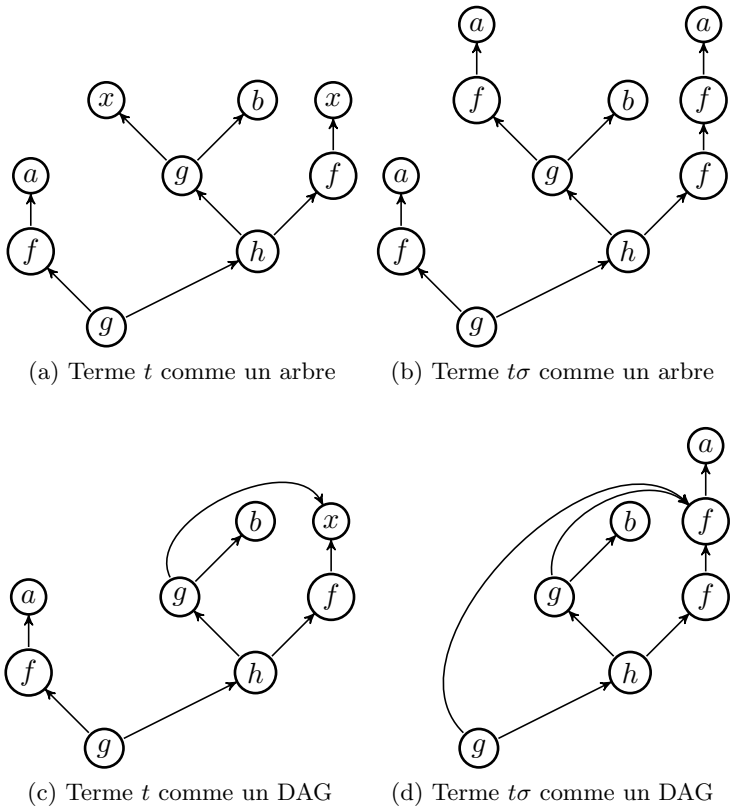
$$t = g(f(a), h(g(x, b), f(x))).$$

Un terme peut être considéré comme un arbre (voir Figure 1a).

Considérons une substitution $\sigma = \{x \mapsto f(a)\}$. Alors le terme obtenu par l'application de la substitution σ sur le terme t est

$$t\sigma = g(f(a), h(g(f(a), b), f(f(a)))).$$

[5]Les ensembles $\mathcal{F}$, $\mathcal{X}$ et $\mathcal{A}$ sont considérés comme étant disjoints

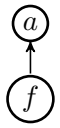


(a) Terme t comme un arbre (b) Terme $t\sigma$ comme un arbre

(c) Terme t comme un DAG (d) Terme $t\sigma$ comme un DAG

Figure 1: Représentations arborescente et en forme DAG de termes

Une représentation d'une forme d'arbre de ce terme est donné sur Figure 1b.

Informellement, chaque nœud dans cet arbre correspond à un sousterme. C'est-à-dire que si l'on prend n'importe quel sousarbre de cet arbre de t alors ce sera une représentation arborescente d'un certain sousterme de t. Par exemple, un sousarbre  représente un terme $f(a)$ qui est un sousterme de t.

L'inverse est aussi vrai : chaque sousterme du terme t correspond à un sousarbre de la représentation arborescente de t.

Il existe une représentation "plus compacte" de termes sous la forme de graphe orienté acyclique (DAG), où, de façon informelle, il n'y a pas deux nœuds différents qui représentent le même (sous-)terme. Les représentations DAG des termess t et $t\sigma$ sont données à Figures 1c et 1d.

Théories equationnelles

Comme nous travaillons avec le modèle symbolique sur des messages du monde réel, nous devons envisager une équivalence entre deux termes qui sont syntaxiquement différentes mais représentent les mêmes chaînes de bits. Pour ce faire, une notion de théorie équationnelle est introduite.

Définition 0.0.7 Une *théorie équationnelle* $\mathcal{E}$ sur un ensemble de termes $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ est un

ensemble d'*identités* $\mathcal{E} = \{u_i = v_i\}_{i\in\mathbb{I}}$ où u_i et v_i sont de termes de $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{Y})$, $\mathcal{Y} \cap \mathcal{X} = \varnothing$ et $\mathbb{I}$ est un ensemble au plus dénombrable. Une théorie équationnelle $\mathcal{E}$ engendre une certaine relation binaire minimale de *congruence* $\equiv \subseteq \mathcal{T} \times \mathcal{T}$ sur l'algèbre de termes qui contient toutes les paires de termes $p \equiv_\mathcal{E} q$ à chaque fois qu'il existe $(u = v) \in \mathcal{E}$ et substitution $\sigma : \mathcal{Y} \mapsto \mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ tels que $p = u\sigma$ et $q = v\sigma$ ($\equiv_\mathcal{E}$ est notre notation pour $\overset{*}{\leftrightarrow}_E$ utilisé dans [BN98], où vous pouvez trouver tous les détails manquants ici). La relation de congruence est une relation binaire qui est

- réflexive ($p \equiv_\mathcal{E} p$)
- symétrique ($p \equiv_\mathcal{E} q$ implique $q \equiv_\mathcal{E} p$)
- et transitive ($p \equiv_\mathcal{E} r$, $r \equiv_\mathcal{E} q$ implique $p \equiv_\mathcal{E} q$)

et compatible avec l'algèbre de termes, i.e. $t_i \equiv_\mathcal{E} s_i$ for $i = 1, \ldots, n$ implique $f(t_1, \ldots, t_n) \equiv_\mathcal{E} f(s_1, \ldots, t_n)$ pour tout n-aire symbole fonctionnel de $\mathcal{F}$. On écrit $\equiv$ au lieu de $\equiv_\mathcal{E}$ si la théorie équationnelle sous question est claire du contexte.

Définition 0.0.8 Une théorie équationnelle $\mathcal{E}$ divise l'ensemble des termes $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ dans des *classes d'équivalence*. Une classe d'équivalence $[t]_\mathcal{E}$ d'un terme t est un ensemble de tous les éléments de $\mathcal{T}$ auxquels il équivaut, soit $[t]_\mathcal{E} = \{p \in \mathcal{T} \,|\, p \equiv t\}$. Si l'on peut fixer un élément de chaque classe d'équivalence, nous pouvons l'appeler une *forme normale*. On dénote une forme normale d'une classe d'équivalence $[t]_\mathcal{E}$ par $\ulcorner t \urcorner$.

0.1 Dérivations et systèmes de contraintes

Les règles de déduction sont nécessaires pour spécifier les opérations qui peuvent être effectuées sur la représentation symbolique des messages du monde réel.

Définition 0.0.9 Une *règle* (de déduction) est un tuple de termes écrit comme $s_1, \ldots, s_k \to s$, où $s_1, \ldots, s_k, s$ sont des termes. Un *système de déduction* $\mathcal{D}$ est un ensemble de règles de déduction.

Nous supposons que les règles de déduction qui seront examinées appartiennent à un système de déduction fixe $\mathcal{D}$.

Définition 0.0.10 Une *instance close* d'une règle $d = s_1, \ldots, s_k \to s$ est une règle $l = l_1, \ldots, l_k \to r$ où $l_1, \ldots, l_k, r$ sont des termes clos et il existe une substitution close σ, telle que $l_i = s_i\sigma, \forall i = 1, \ldots, k$ et $r = s\sigma$. On appellera une instance close d'une règle une *règle close* (ou parfois tout simplement *règle* s'il n'y a pas d'ambiguïté).

Étant donné deux ensembles de termes clos E, F et une règle close $l \to r$, nous écrivons $E \to_{l\to r} F$ ssi $F = E \cup \{r\}$ et $l \subseteq E$, où l est un ensemble de termes. Nous écrivons $E \to F$ ssi il existe une règle close $l \to r$ telle que $E \to_{l\to r} F$.

Définition 0.0.11 Une *dérivation* D de longueur $n \geq 0$ est une suite d'ensembles finis de termes clos $E_0, E_1, \ldots, E_n$ tel que $E_0 \to E_1 \to \cdots \to E_n$, où $E_i = E_{i-1} \cup \{t_i\}, \forall i = \{1, \ldots, n\}$. Un terme t est *dérivable* à partir d'un ensemble de termes E ssi il existe une dérivation $D = E_0, \ldots, E_n$ telle que $E_0 = E$ et $t \in E_n$. Un ensemble de termes T est dérivable de E, ssi chaque $t \in T$ est dérivable de E. Nous noterons $\mathrm{Der}(E)$ l'ensemble de termes dérivables à partir de E.

Une dérivation représente l'évolution possible étape-par-étape d'un ensemble de termes; cette évolution est faisable dans le sens que chaque ensemble suivant est obtenu par une application de certaines règles de déduction sur les termes de l'ensemble précédent.

Définition 0.0.12 Soit E un ensemble de termes et t un terme, nous définissons le couple (E, t) notée $E \triangleright t$ comme étant une *contrainte*. Un *système de contraintes* est un ensemble

$$\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$$

où n est un entier et $E_i \triangleright t_i$ est une contrainte pour tout $i \in \{1, \ldots, n\}$.

On dénote par $\text{Vars}(\mathcal{S})$ l'ensemble des variables utilisées dans $\mathcal{S}$. On dit que $\mathcal{S}$ est normalisé, si chaque terme intervenant dans $\mathcal{S}$ est normalisé. Par $\ulcorner\mathcal{S}\urcorner$ on dénote le système de contraintes $\{\ulcorner E_i \urcorner \triangleright \ulcorner t_i \urcorner\}_{i=1,\ldots,n}$.

Définition 0.0.13 Une substitution close σ est un *modèle* d'une contrainte $E \triangleright t$ (ou σ satisfait cette contrainte), si $\ulcorner t\sigma \urcorner \in \text{Der}(\ulcorner E\sigma \urcorner)$. Une substitution close σ est un *modèle* d'un système de contraintes $\mathcal{S}$, si elle satisfait toutes les contraintes de $\mathcal{S}$ et que $\text{dom}(\sigma) = \text{Vars}(\mathcal{S})$.

Contraintes Dolev-Yao étendues avec un symbole ACI

Nous présentons une procédure de décision pour le problème de la satisfiabilité des systèmes de contraintes généraux où le système de déduction de Dolev-Yao est étendu par des règles supportant un symbole associatif-commutatif-idempotent (DY+ACI). Nous considérons les opérateurs pour l'enchaînement, cryptage symétrique et asymétrique, le décryptage, la signature, le hachage et un opérateur ACI qui sera utilisé comme un constructeur d'ensembles.

Les principales étapes pour montrer la décidabilité sont comme suit:

1. Nous présentons un algorithme pour résoudre le problème de dérivabilité (le cas de termes clos) dans le modèle DY + ACI.
2. Nous montrons que la normalisation n'affecte pas la satisfiabilité d'un système de contraintes par une substitution: soit nous normalisons une substitution ou bien un système de contraintes.
3. On prouve l'existence d'une solution conservatrice du système de contraintes satisfiable: une substitution σ qui envoie une variable à un ensemble ACI de quasi-soustermes du système de contraintes instanciées avec σ et les clés privées correspondantes aux valeurs atomiques du système de contraintes.
4. Nous donnons une borne sur la taille d'une solution conservatrice, et, comme conséquence, nous obtenons la décidabilité.

Introduction formelle au problème

Termes et notions

Tout d'abord on instancie la classe des termes que nous allons considérer pour DY + ACI.

Définition 0.0.14 Les *termes* sont définis selon la grammaire suivante:

$$\begin{array}{lll}
term & ::= & variable \mid atom \mid \text{pair}(term, term) \mid \\
 & & \quad \text{enc}(term, term) \mid \cdot(tlist) \mid \text{priv}(Keys) \mid \\
 & & \quad \text{aenc}(term, Keys) \mid \text{sig}(term, \text{priv}(Keys)) \mid \\
 & & \quad \text{apply}(atom, term) \\
Keys & ::= & variable \mid atom \\
tlist & ::= & term \mid term, tlist
\end{array}$$

Term	Description
$\mathrm{enc}(t_1, t_2)$	t_1 crypté avec clef symétrique t_2
$\mathrm{aenc}(t_1, t_2)$	t_1 crypté avec clef public asymétrique t_2
$\mathrm{pair}(t_1, t_2)$	t_1 concaténé avec t_2
$\mathrm{priv}(t_2)$	clé privée pour le clé publique t_2
$\mathrm{sig}(t_1, \mathrm{priv}(t_2))$	signature du message t_1 avec clé privée $\mathrm{priv}(t_2)$
$\mathrm{apply}(t_1, t_2)$	application d'une fonction de hashage t_1 sur message t_2
$\cdot(\{t_1, \ldots, t_n\})$	ensemble de messages $t_1, \ldots, t_n$

Tableau 1: Explication des symboles fonctionnels

où $atom \in \mathcal{A}$, $variable \in \mathcal{X}$.

La brève explication des symboles fonctionnels est donnée dans le Tableau 1.

Remarquons que nous ne permettons pas les clés complexes pour le chiffrement asymétrique. En conséquence, nous avons une restriction sur les applications de substitution: uen substitution σ ne peut pas être appliquée à un terme t, si le résultat ne respecte pas la grammaire donnée.

On dénote le terme à i-ème position d'une liste de termes L par $L[i]$. $t \in L$ est un raccourci pour $\exists i : t = L[i]$. Nous définissons aussi deux relations binaires $\subseteq$ et $\approx$ sur les listes comme suit: $L_1 \subseteq L_2$ si et seulement si $\forall t \in L_1 \implies t \in L_2$; $L_1 \approx L_2$ si et seulement si $L_1 \subseteq L_2$ et $L_2 \subseteq L_1$, et nous les étendons naturellement si L_1 ou L_2 soit un ensemble.

Définition 0.0.15 Nous considérons le symbole $\cdot$ d'être

- associatif: $\cdot(\{t_1, \ldots, t_k, \cdot(\{t_{k+1}, \ldots, t_m\}), t_{m+1}, \ldots, t_n\}) \equiv_{ACI} \cdot(\{t_1, \ldots, t_n\})$,
- commutatif: $\cdot(\{t_1, t_2\}) \equiv_{ACI} \cdot(\{t_2, t_1\})$ et
- idempotent: $\cdot(\{t_1, t_1\}) \equiv_{ACI} \cdot(\{t_1\})$

En plus, nous supposons que $\cdot(\{t\}) \equiv_{ACI} t$. C'est-à-dire, que la théorie équationnelle ACI est
$\{\cdot(\{y\}) = y, \quad \cdot(\{y_1, y_1\}) = \cdot(\{y_1\}), \quad \cdot(\{y_1, y_2\}) = \cdot(\{y_2, y_1\})\} \cup$
$\{\cdot(\{t_1, \ldots, t_k, \cdot(\{t_{k+1}, \ldots, t_m\}), t_{m+1}, \ldots, t_n\}) = \cdot(\{t_1, \ldots, t_n\})\}_{0 \le k < m \le n}$.

Définition 0.0.16 Pour tout terme $t \in \mathcal{T}$ on défini l'ensemble de ses *éléments* comme suit:

$$\mathrm{elems}(t) = \begin{cases} \bigcup_{p \in L} \mathrm{elems}(p) & \text{si } t = \cdot(L); \\ \{t\}, & \text{sinon.} \end{cases}$$

Pour une liste ou un ensemble de termes T on a $\mathrm{elems}(T) = \bigcup_{t \in T} \mathrm{elems}(t)$.

Exemple 2 Considérons un terme $t = \cdot(\{a, \cdot(\{b, a, \mathrm{pair}(a, b)\}), \mathrm{pair}(\cdot(\{b, b\}), a)\})$. L'ensemble de ces éléments est $\mathrm{elems}(t) = \{a, b, \mathrm{pair}(\cdot(\{b, b\}), a), \mathrm{pair}(a, b)\}$. Voir la Figure 2.

Définition 0.0.17 Soit $\prec$ un ordre calculable sur $\mathcal{T}$ qui est strict et total, tel que la question de savoir si $p \prec q$ peut être répondue en temps polynomial.

Définition 0.0.18 La cardinalité d'un ensemble P est notée par $|P|$.

Nous allons utiliser bin comme une généralisation de tous les opérateurs binaires:

$$\mathrm{bin} \in \{\mathrm{enc}, \mathrm{aenc}, \mathrm{pair}, \mathrm{sig}, \mathrm{apply}\}.$$

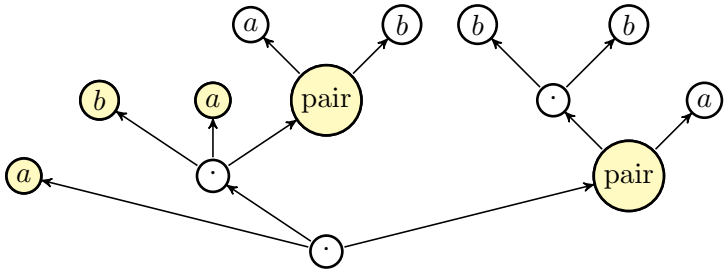
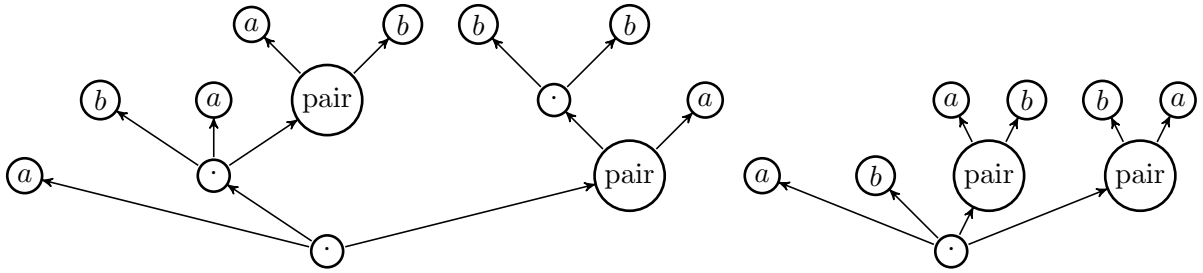


Figure 2: Illustration pour l'Exemple 2: Éléments de t (colorés)



(a) Terme t comme un arbre

(b) Terme $\ulcorner t \urcorner$ comme un arbre

Figure 3: Illustration pour l'Exemple 3: Forme normale d'un terme

Définition 0.0.19 La *forme normale* d'un terme t (dénoté par $\ulcorner t \urcorner$) est définie récursivement comme suit:

- $\ulcorner t \urcorner = t$, si $t \in \mathcal{X} \cup \mathcal{A}$
- $\ulcorner \mathrm{bin}\,(t_1, t_2) \urcorner = \mathrm{bin}\,(\ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner)$
- $\ulcorner \mathrm{priv}\,(t) \urcorner = \mathrm{priv}\,(\ulcorner t \urcorner)$
- $\ulcorner \cdot (L) \urcorner = \begin{cases} \cdot (L'), & \text{si } |\ulcorner \mathrm{elems}\,(L) \urcorner| > 1 \text{ et } L' \approx \ulcorner \mathrm{elems}\,(L) \urcorner \\ & \text{et pour tous } i < j,\ L'[i] \prec L'[j]; \\ t', & \text{si } \ulcorner \mathrm{elems}\,(L) \urcorner = \{t'\} \end{cases}$,

où pour un ensemble de termes T, $\ulcorner T \urcorner = \{\ulcorner t \urcorner : t \in T\}$.

On dit qu'un terme t est *normalisé*, ssi $t = \ulcorner t \urcorner$.

On peut voir que deux termes sont congrus modulo les propriétés ACI de $\cdot$ ssi ils ont la même forme normale.

Exemple 3 Pour le terme $t = \cdot(\{a, \cdot(\{b, a, \mathrm{pair}\,(a, b)\}), \mathrm{pair}\,(\cdot(\{b, b\}), a)\})$ on a $\ulcorner t \urcorner = \cdot(\{a, b, \mathrm{pair}\,(a, b), \mathrm{pair}\,(b, a)\})$ (if $a \prec b \prec \mathrm{pair}\,(a, b) \prec \mathrm{pair}\,(b, a)$). Voir Figure 3.

Définition 0.0.20 Soit t un terme. On défini un ensemble de *quasi-soustermes* $\mathrm{QSub}\,(t)$ comme suit:

$$\mathrm{QSub}\,(t) = \begin{cases} \{t\}, & \text{si } t \in \mathcal{X} \cup \mathcal{A}; \\ \{t\} \cup \mathrm{QSub}\,(t_1), & \text{si } t = \mathrm{priv}\,(t_1); \\ \{t\} \cup \mathrm{QSub}\,(t_1) \cup \mathrm{QSub}\,(t_2), & \text{si } t = \mathrm{bin}\,(t_1, t_2) \\ \{t\} \cup \bigcup_{p \in \mathrm{elems}(L)} \mathrm{QSub}\,(p), & \text{si } t = \cdot(L) \end{cases}$$

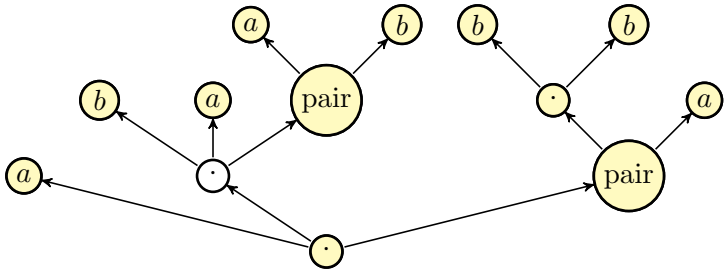


Figure 4: Illustration pour l'Exemple 4: Quasi-soustermes de t (colorés)

Si T est un ensemble de termes, alors $\mathrm{QSub}(T) = \bigcup_{t\in T} \mathrm{QSub}(t)$. Si $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$ est une système de contraintes, on défini $\mathrm{QSub}(\mathcal{S}) = \bigcup_{t\in\bigcup_{i=1}^{n} E_i\cup\{t_i\}} \mathrm{QSub}(t)$.

Exemple 4 Pour le terme $t = \cdot(\{a, \cdot(\{b, a, \mathrm{pair}(a,b)\}), \mathrm{pair}(\cdot(\{b,b\}), a)\})$ on a

$$\begin{aligned}\mathrm{QSub}(t) = \{&\cdot(\{a, \cdot(\{b, a, \mathrm{pair}(a,b)\}), \mathrm{pair}(\cdot(\{b,b\}), a)\}),\\ &a, b, \mathrm{pair}(a,b), \mathrm{pair}(\cdot(\{b,b\}), a), \cdot(\{b,b\})\}.\end{aligned}$$

Voir Figure 4.

Nous définissons $\mathrm{Sub}(t)$ comme l'ensemble des sous-termes de t et la taille (size) d'un terme, comme le nombre de ses sous-termes différents.

Définition 0.0.21 Soit T un ensemble de termes, alors $\mathrm{Sub}(T) = \bigcup_{t\in T} \mathrm{Sub}(t)$. Soit $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$ un système de contraintes, on défini $\mathrm{Sub}(\mathcal{S}) = \bigcup_{t\in\bigcup_{i=1}^{n} E_i\cup\{t_i\}} \mathrm{Sub}(t)$.

Exemple 5 Pour le terme $t = \cdot(\{a, \cdot(\{b, a, \mathrm{pair}(a,b)\}), \mathrm{pair}(\cdot(\{b,b\}), a)\})$ on a

$$\begin{aligned}\mathrm{Sub}(t) = \{&\cdot(\{a, \cdot(\{b, a, \mathrm{pair}(a,b)\}), \mathrm{pair}(\cdot(\{b,b\}), a)\}),\\ &\cdot(\{b, a, \mathrm{pair}(a,b)\}), \mathrm{pair}(\cdot(\{b,b\}), a),\\ &a, b, \mathrm{pair}(a,b), \cdot(\{b,b\})\}.\end{aligned}$$

Définition 0.0.22 On défini la taille size d'un terme t par $\mathrm{size}(t) = |\mathrm{Sub}(t)|$, pour un ensemble de termes T, $\mathrm{size}(T) = |\mathrm{Sub}(T)|$ et pour système de contraintes $\mathcal{S}$ comme $\mathrm{size}(\mathcal{S}) = |\mathrm{Sub}(\mathcal{S})|$.

À noter que cette définition n'a pas de correspondance polynomiale avec le nombre de bits nécessaires pour décrire l'objet sous question.

Soit $t_1, t_2, \ldots, t_m \in \mathcal{T}$. On défini un système de déduction Dolev-Yao modulo ACI comme un ensemble de règles de Tableau 2).

Nous supposons, ci-après, que le système de contraintes considéré $\mathcal{S}$ contient au moins un atome comme son sous-terme.

Nous désignons $\{\mathrm{priv}(t) : t \in T\}$ pour un ensemble de termes T as $\mathrm{priv}(T)$. Nous désignons $\mathrm{Vars}(\mathcal{S}) = \bigcup_{i=1}^{n} \mathrm{Vars}(E_i) \cup \mathrm{Vars}(t_i)$. On dit que $\mathcal{S}$ est normalisé, ssi pour tout $t \in \mathrm{QSub}(\mathcal{S})$, t est normalisé.

Exemple 6 Nous donnons un exemple de système de contraintes général et sa solution au sein du système de déduction DY + ACI.

$$\mathcal{S} = \left\{ \begin{array}{lcl} \mathrm{enc}(x,a), \mathrm{pair}(c,a) & \triangleright & b \\ \cdot(\{x,c\}) & \triangleright & a \end{array} \right\},$$

où $a, b, c \in \mathcal{A}$ et $x \in \mathcal{X}$.

Un exemple de modèle de $\mathcal{S}$ dans DY+ACI est $\sigma = \{x \mapsto \mathrm{enc}(\mathrm{pair}(a,b), c)\}$.

Règles de composition	Règles de décomposition
$t_1, t_2 \to \ulcorner \mathrm{enc}(t_1, t_2) \urcorner$	$\mathrm{enc}(t_1, t_2), \ulcorner t_2 \urcorner \to \ulcorner t_1 \urcorner$
$t_1, t_2 \to \ulcorner \mathrm{aenc}(t_1, t_2) \urcorner$	$\mathrm{aenc}(t_1, t_2), \ulcorner \mathrm{priv}(t_2) \urcorner \to \ulcorner t_1 \urcorner$
$t_1, t_2 \to \ulcorner \mathrm{pair}(t_1, t_2) \urcorner$	$\mathrm{pair}(t_1, t_2) \to \ulcorner t_1 \urcorner$
$t_1, \mathrm{priv}(t_2) \to \ulcorner \mathrm{sig}(t_1, \mathrm{priv}(t_2)) \urcorner$	$\mathrm{pair}(t_1, t_2) \to \ulcorner t_2 \urcorner$
$t_1, \ldots, t_m \to \ulcorner \cdot(t_1, \ldots, t_m) \urcorner$	$\cdot(t_1, \ldots, t_m) \to \ulcorner t_i \urcorner$ pour tout i
$t_1, t_2 \to \ulcorner \mathrm{apply}(t_1, t_2) \urcorner$	

Tableau 2: Règles du système de déduction DY+ACI

Définition 0.0.23 Soit $T = \{t_1, \ldots, t_k\}$ un ensemble non-vide de termes. Nous définissons $\pi(T)$ comme suit:

$$\pi(T) = \ulcorner \cdot(t_1, \ldots, t_k) \urcorner$$

Remarque: $\pi(\{t\}) = \ulcorner t \urcorner$.

Définition 0.0.24 Nous noterons un ensemble de quasi-soustermes d'un système de contraintes $\mathcal{S}$ qui ne sont pas des variables par $\mathrm{QSub}^{\circ}(S, \mathcal{X})$, i.e. $\mathrm{QSub}(\mathcal{S}) \setminus \mathcal{X} = \mathrm{QSub}^{\circ}(S, \mathcal{X})$. Quand l'ensemble de toutes les variables $\mathcal{X}$ est évident, nous utiliserons la notation courte $\mathrm{QSub}^{\circ}(S)$ au lieu de $\mathrm{QSub}^{\circ}(S, \mathcal{X})$.

Nous introduisons une transformation $\pi(H^{\mathcal{S},\sigma}(\cdot))$. Plus tard, nous allons montrer que $\pi(\mathrm{H}(\sigma))$ est également un modèle de $\mathcal{S}$, si σ est un modèle de $\mathcal{S}$.

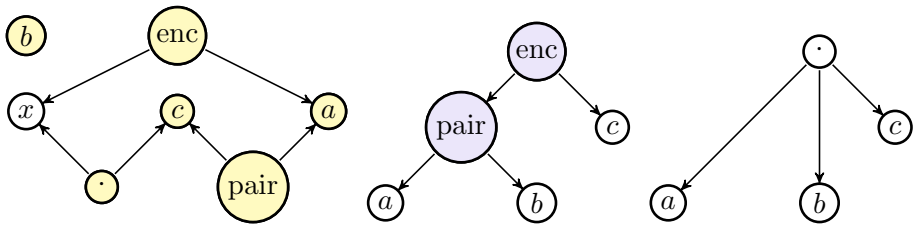
Définition 0.0.25 Soit $\mathcal{S}$ un système de contraintes satisfiable par modèle σ. Fixons $\alpha \in (\mathcal{A} \cap \mathrm{QSub}(\mathcal{S}))$. Pour $\mathcal{S}$ et σ donnés nous définissons la fonction $H^{\mathcal{S},\sigma}(\cdot) : \mathcal{T}_g \to 2^{\mathcal{T}_g}$ comme suit:

$$H^{\mathcal{S},\sigma}(t) = \begin{cases} \{\alpha\}, & \text{si } t \in (\mathcal{A} \setminus \mathrm{QSub}(\mathcal{S})); \\ \{a\}, & \text{si } t = a \in (\mathcal{A} \cap \mathrm{QSub}(\mathcal{S})); \\ \left\{\mathrm{priv}\left(\pi(H^{\mathcal{S},\sigma}(t_1))\right)\right\}, & \text{si } t = \mathrm{priv}(t_1); \\ \left\{\mathrm{bin}\left(\pi(H^{\mathcal{S},\sigma}(t_1)), \pi(H^{\mathcal{S},\sigma}(t_2))\right)\right\}, & \text{si } t = \mathrm{bin}(t_1, t_2) \\ & \ulcorner t \urcorner \in \ulcorner \mathrm{QSub}^{\circ}(S)\,\sigma \urcorner \\ H^{\mathcal{S},\sigma}(t_1) \cup H^{\mathcal{S},\sigma}(t_2), & \text{si } t = \mathrm{bin}(t_1, t_2) \\ & \wedge\ \ulcorner t \urcorner \notin \ulcorner \mathrm{QSub}^{\circ}(S)\,\sigma \urcorner \\ \bigcup_{p \in L} H^{\mathcal{S},\sigma}(p), & \text{si } t = \cdot(L). \end{cases}$$

Désormais, nous allons omettre les paramètres et nous allons écrire $H(\cdot)$ au lieu de $H^{\mathcal{S},\sigma}(\cdot)$.

Soit $T = \{t_1, \ldots, t_k\}$, $t_i \in \mathcal{T}$. On défini $\pi(H(T)) = \{\pi(H(t)) \mid t \in T\}$.
Soit $\theta = \{x_1 \mapsto t_1, \ldots, x_k \mapsto t_k\}$ une substitution. On défini $\pi(H(\theta))$ comme une substitution $\pi(H(\theta)) = \{x_1 \mapsto \pi(H(t_1)), \ldots, x_k \mapsto \pi(H(t_k))\}$.

Exemple 7 Prenons $\mathcal{S}$ et son modèle σ de l'Exemple 6 et montrons que $\pi(H(\sigma))$ est aussi un modèle de $\mathcal{S}$. $\pi(H(\mathrm{enc}(\mathrm{pair}(a, b), c))) = \pi(H(\mathrm{pair}(a, b)) \cup \{c\}) = \pi(\{a\} \cup \{b\} \cup \{c\}) = \cdot(\{a, b, c\})$ (supposons $a \prec b \prec c$). On peut voir que $\pi(\mathrm{H}(\sigma)) = \{x \mapsto \cdot(\{a, b, c\})\}$ est aussi un modèle de $\mathcal{S}$ dans DY+ACI. Voir Figure 5.



(a) $\mathrm{QSub}(\mathcal{S})$ comme un DAG; $\mathrm{QS\mathring{u}b}(\mathcal{S})$ sont colorés

(b) $x\sigma$ comme un DAG Soustermes n'ayant pas de pattern de $\mathrm{QS\mathring{u}b}(\mathcal{S})$ sont colorés

(c) $\pi(H(x\sigma))$ comme un DAG

Figure 5: Illustration pour l'Exemple 7: Travail de la fonction $H^{\mathcal{S},\sigma}(\cdot)$

Propriétés générales

Lemme 1. *Soit* $A, B, C \subseteq \mathcal{T}_g$. *Si* $A \subseteq \mathrm{Der}(B)$ *et* $B \subseteq \mathrm{Der}(C)$ *alors* $A \subseteq \mathrm{Der}(C)$.

Lemme 2. *Soit* $A, B, C, D \subseteq \mathcal{T}_g$. *Si* $A \subseteq \mathrm{Der}(B)$ *et* $C \subseteq \mathrm{Der}(D)$ *alors* $A \cup C \subseteq \mathrm{Der}(B \cup D)$.

Lemme 3. *Les propriétés qui suivent sont vraies:*

1. $\ulcorner\cdot(t,t)\urcorner = \ulcorner t\urcorner$, $\ulcorner\cdot(t_1,t_2)\urcorner = \ulcorner\cdot(t_2,t_1)\urcorner$, $\ulcorner\cdot(\cdot(t_1,t_2),t_3)\urcorner = \ulcorner\cdot(t_1,\cdot(t_2,t_3))\urcorner = \ulcorner\cdot(t_1,t_2,t_3)\urcorner$
2. *Si* t *et* $t\sigma$ *sont des termes, alors* $\ulcorner t\sigma\urcorner = \ulcorner\ulcorner t\sigma\urcorner\urcorner = \ulcorner\ulcorner t\urcorner\sigma\urcorner = \ulcorner t\ulcorner\sigma\urcorner\urcorner = \ulcorner\ulcorner t\urcorner\ulcorner\sigma\urcorner\urcorner$
3. $s \in \mathrm{QSub}(\ulcorner t\urcorner) \implies s = \ulcorner s\urcorner$
4. $\forall s \in \mathrm{Sub}(\ulcorner t\urcorner)\ \exists s' \in \mathrm{Sub}(t) : s = \ulcorner s'\urcorner$
5. $\ulcorner\mathrm{elems}(t)\urcorner = \mathrm{elems}(\ulcorner t\urcorner)$
6. $\ulcorner\cdot(\ulcorner t_1\urcorner,\ldots,\ulcorner t_m\urcorner)\urcorner = \ulcorner\cdot(t_1,\ldots,t_m)\urcorner$; $\pi(T) = \pi(\ulcorner T\urcorner)$
7. $\mathrm{elems}(\ulcorner\cdot(\ulcorner t_1\urcorner,\ldots,\ulcorner t_m\urcorner)\urcorner) = \mathrm{elems}(\cdot(\ulcorner t_1\urcorner,\ldots,\ulcorner t_m\urcorner)) = \bigcup_{i=1,\ldots,m}\mathrm{elems}(\ulcorner t_i\urcorner)$
8. $H(t) = \bigcup_{p\in\mathrm{elems}(t)} H(p)$,
9. $H(t) = H(\ulcorner t\urcorner)$
10. $\pi(H(t)) = \pi(H(\ulcorner t\urcorner)) = \ulcorner\pi(H(t))\urcorner = \ulcorner\pi(H(\ulcorner t\urcorner))\urcorner$
11. $\pi(T_1 \cup \cdots \cup T_m) = \pi(\{\pi(T_1),\ldots,\pi(T_m)\})$
12. $\mathrm{QSub}(\mathrm{QSub}(t)) = \mathrm{QSub}(t)$
13. $\mathrm{QSub}(\ulcorner t\urcorner) \subseteq \ulcorner\mathrm{QSub}(t)\urcorner$
14. $\mathrm{QSub}(t\sigma) \subseteq \mathrm{QSub}(t)\,\sigma \cup \mathrm{QSub}(\mathrm{Vars}(t)\,\sigma)$
15. $\mathrm{Sub}(t\sigma) = \mathrm{Sub}(t)\,\sigma \cup \mathrm{Sub}(\mathrm{Vars}(t)\,\sigma)$
16. $|\ulcorner T\urcorner| \leq |T|$, $|T\sigma| \leq |T|$
17. $\mathrm{elems}(t) \subseteq \mathrm{QSub}(t) \subseteq \mathrm{Sub}(t)$

Règles de composition	Règles de décomposition
$t_1, t_2 \to \ulcorner \mathrm{enc}\,(t_1, t_2) \urcorner$	$\mathrm{enc}\,(t_1, t_2)\,, \ulcorner t_2 \urcorner \to \ulcorner t_1 \urcorner$
$t_1, t_2 \to \ulcorner \mathrm{aenc}\,(t_1, t_2) \urcorner$	$\mathrm{aenc}\,(t_1, t_2)\,, \ulcorner \mathrm{priv}\,(t_2) \urcorner \to \ulcorner t_1 \urcorner$
$t_1, t_2 \to \ulcorner \mathrm{pair}\,(t_1, t_2) \urcorner$	$\mathrm{pair}\,(t_1, t_2) \to \ulcorner t_1 \urcorner$
$t_1, \mathrm{priv}\,(t_2) \to \ulcorner \mathrm{sig}\,(t_1, \mathrm{priv}\,(t_2)) \urcorner$	$\mathrm{pair}\,(t_1, t_2) \to \ulcorner t_2 \urcorner$
$t_1, \ldots, t_m \to \ulcorner \cdot\,(t_1, \ldots, t_m) \urcorner$	$t \to \ulcorner s \urcorner$ pour tout $s \in \mathrm{elems}\,(t)$, où $t = \cdot\,(L)$
$t_1, t_2 \to \ulcorner \mathrm{apply}\,(t_1, t_2) \urcorner$	

Tableau 3: Règles du système de déduction DY+ACI'

18. *Pour un terme t,* $\mathrm{size}\,(\ulcorner t \urcorner) \leq \mathrm{size}\,(t)$*;*
pour un ensemble de termes T, $\mathrm{size}\,(\ulcorner T \urcorner) \leq \mathrm{size}\,(T)$*;*
pour un système de contraintes $\mathcal{S}$, $\mathrm{size}\,(\ulcorner \mathcal{S} \urcorner) \leq \mathrm{size}\,(\mathcal{S})$

19. $\mathrm{QSub}\,(\cdot\,(\{t_1, \ldots, t_l\})) \subseteq \{\cdot\,(\{t_1, \ldots, t_l\})\} \cup \mathrm{QSub}\,(t_1) \cdots \cup \mathrm{QSub}\,(t_l)$

20. $\forall s \in \mathrm{Sub}\,(t)\ \ \mathrm{size}\,(\ulcorner t\sigma \urcorner) \geq \mathrm{size}\,(\ulcorner s\sigma \urcorner)$.

Lemme 4. *Soit $\mathcal{S}$ un système de contraintes et σ son modèle. Alors la substitution $\pi(\mathrm{H}\,(\sigma))$ est normalisée.*

Lemme 5. *Pour un terme t normalisé,* $\mathrm{QSub}\,(t) = \mathrm{Sub}\,(t)$.

Proposition 1. *Soit T un ensemble de termes $T = \{t_1, \ldots, t_k\}$. Alors $\pi(T) \in \mathrm{Der}\,(\ulcorner T \urcorner)$ et $\ulcorner T \urcorner \subseteq \mathrm{Der}\,(\{\pi(T)\})$.*

Proposition 2. *Une substitution σ est un modèle d'un système de contraintes $\mathcal{S}$ si et seulement si σ est un modèle de $\ulcorner \mathcal{S} \urcorner$.*

Proposition 3. *Une substitution σ est un modèle d'un système de contraintes $\mathcal{S}$ si et seulement si $\ulcorner \sigma \urcorner$ est un modèle de $\mathcal{S}$.*

Le cas clos de DY+ACI

Dans ce paragraphe nous considérons une question de dérivabilité clos dans DY + ACI: étant donné un terme clos normalisé t et un ensemble de termes clos normalisés E, le terme t est-il dérivable à partir de E en utilisant des opérations du Tableau 2 ? Nous présenterons un algorithme qui est capable de répondre à cette question en temps polynomial.

Considérons un système de déduction DY+ACI' (Tableau 3) équivalent à DY+ACI (Tableau 2) obtenu à partir de ce dernier en remplaçant un ensemble de règles $\cdot\,(t_1, \ldots, t_m) \to \ulcorner t_i \urcorner$ pour tout i avec un nouvel ensemble de règles $t \to \ulcorner s \urcorner$ pour tout $s \in \mathrm{elems}\,(t)$, où $t = \cdot\,(L)$.

Lemme 6. $t \in \mathrm{Der}_{DY+ACI}\,(E) \iff t \in \mathrm{Der}_{DY+ACI'}\,(E)$

L'idée de la preuve. On peut "simuler" chaque règle d'un système par des règles d'un autre. □

Algorithm 1: Vérifier la dérivabilité d'un terme

Input: Une contrainte close normalisée $E \triangleright t$

Output: $t \in \mathrm{Der}_{DY+ACI}(E)$

```
1  Let S := QSub(E) ∪ QSub(t) ∖ E;
2  Let D := E;
3  while true do
4      if il existe une règle DY l → r, telle que l ⊆ D et r ∈ S then
5          S := S ∖ {r};
6          D := D ∪ {r};
7      else
8          if il existe s ∈ S : elems(s) ⊆ D then
9              S := S ∖ {s};
10             D := D ∪ {s};
11         else
12             if il existe s ∈ D : elems(s) ⊈ D then
13                 S := S ∖ elems(s);
14                 D := D ∪ elems(s);
15             else
16                 return t ∈ D;
```

Proposition 4. *L'algorithme 1 termine et il est correct.*

Quelques remarques sur la preuve. La terminaison est évidente car pour chaque "if" réussit l'ensemble S diminue. Dans cet algorithme on essaie de trouver tout les quasi-soustermes de $E \cup \{t\}$ qui sont déductible à partir de E pour les règles de déduction DY+ACI' (on peut prouver une propriété de *localité* disant que l'on n'a pas besoin de construire des termes en dehors de $\mathrm{QSub}(E \cup \{t\})$ si le terme de but est t). Et comme DY+ACI' est "équivalent" à DY+ACI, le résultat retourné par l'algorithme est bien celui attendu. □

0.2 L'existence d'une solution conservatrice

Nous allons montrer que pour un système de contrainte satisfiable, il existe un modèle en forme spéciale (appelé *solution conservatrice*). Grosso modo, un modèle sous cette forme peut être défini de façon unique pour chaque variable par un ensemble de quasi-soustermes du système de contraintes et un ensemble d'atomes (également appartenant à ce système) qui peuvent être considéré comme des clefs publiques. Ceci limiterait l'espace de recherche d'un modèle.

Proposition 5. *Soit $\mathcal{S}$ un système de contraintes normalisé et σ son modèle . Pour tout $t \in \mathrm{QSub}(\mathcal{S})$, on a $\ulcorner t\,\pi(\mathrm{H}(\sigma))\urcorner = \ulcorner \pi(H(t\sigma))\urcorner$.*

Lemme 7. *Soit $\mathcal{S}$ un système de contraintes normalisé et σ son modèle. Pour toutes les règles de déduction DY+ACI $l_1, \ldots, l_k \to r$, on a $\pi(H(r)) \in \mathrm{Der}(\{\pi(H(l_1)), \ldots, \pi(H(l_k))\})$.*

Théorème 1. *Soit $\mathcal{S}$ un système de contraintes normalisé et σ son modèle. Alors, la substitution $\pi(\mathrm{H}(\sigma))$ satisfait $\mathcal{S}$.*

Idée d'une preuve. Découle de Proposition 5 et Lemma 7. □

Proposition 6. *Soit $\mathcal{S}$ un système de contraintes normalisé et σ son modèle tel que $\forall x, y \in \mathrm{dom}(\sigma), x \neq y \implies x\sigma \neq y\sigma$. Alors pour tout $x \in \mathrm{dom}(\pi(\mathrm{H}(\sigma)))$ il existe $k \in \mathbb{N}$ et $s_1, \ldots, s_k \in \mathring{\mathrm{QSub}}(S) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$ tels que $\mathrm{root}(s_i) \neq \cdot$ et $x\,\pi(\mathrm{H}(\sigma)) = \pi(\{s_1\,\pi(\mathrm{H}(\sigma)), \ldots, s_k\,\pi(\mathrm{H}(\sigma))\})$.*

Idée d'une preuve. Il suffit de considérer toutes les valeurs possible de s, $s \in H(x\sigma)$, selon la définition de $H(\cdot)$. □

Corollaire 1. *Soit un $\mathcal{S}$ système de contraintes normalisé et σ' son modèle tel que $\forall x, y \in \mathrm{dom}(\sigma'), x \neq y \implies x\sigma' \neq y\sigma'$. Alors il existe un modèle normalisé σ de $\mathcal{S}$ tel que pour tout $x \in \mathrm{dom}(\sigma)$ il existe $k \in \mathbb{N}$ et $s_1, \ldots, s_k \in \mathring{\mathrm{QSub}}(S) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$ tels que $x\sigma = \pi(\{s_1\sigma, \ldots, s_k\sigma\})$ et $s_i \neq s_j$, si $i \neq j$ et $s_i \neq \cdot(L)$ pour tout i.*

Un modèle normalisé ayant la propriété du Corollaire 1 sera appelé *conservateur*:

Définition 0.0.26 Une substitution σ est un *modèle conservateur* (*solution conservatrice*) du système de contraintes $\mathcal{S}$, ssi

1. σ est normalisé;
2. σ est un modèle de $\mathcal{S}$;
3. Pour tout $x \in \mathrm{dom}(\sigma)$ il existe $k \in \mathbb{N}$ et $s_1, \ldots, s_k \in \mathring{\mathrm{QSub}}(S) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$ tels que $x\sigma = \pi(\{s_1\sigma, \ldots, s_k\sigma\})$ et $s_i \neq s_j$, si $i \neq j$ et $s_i \neq \cdot(L)$ pour tout i.

Borne sur les solutions conservatrices

Pour obtenir la décidabilité, nous montrons d'abord une borne supérieure sur la taille du modèle conservateur.

Puis nous réduisons un système de contraintes satisfiable à un autre qui admet un modèle conservateur (cette réduction est faite en utilisant un nom commun pour les variables sur lesquelles les valeurs du modèle préliminaire sont égales). Par ailleurs le modèle conservateur considéré du système de contraintes obtenu peut être facilement étendu à un modèle conservateur (de la même taille!) du système de contraintes initial. Nous pouvons montrer aussi que le système de contraintes réduit n'est pas plus grand que celui d'origine. Cela signifie que le système de contraintes d'origine a un modèle qui est borné à l'égard de la taille du système de contraintes. Ainsi, nous obtenons une existence d'un modèle avec une taille bornée pour tous les systèmes de contraintes satisfiables.

Lemme 8. *Soit $\mathcal{S}$ un système de contraintes normalisé et σ son modèle conservateur. Alors, $\forall x \in \mathrm{Vars}(\mathcal{S}), \mathrm{QSub}(x\sigma) \subseteq \ulcorner \mathrm{QSub}(\mathcal{S})\,\sigma \urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$.*

Preuve. Étant donné d'une substitution close σ, nous définissons un ordre strict et total sur les variables: $x \sqsubset y \iff (\mathrm{size}(x\sigma) < \mathrm{size}(y\sigma)) \vee (\mathrm{size}(x\sigma) = \mathrm{size}(y\sigma) \wedge x \prec y)$.

Par Proposition 6 $\forall x\ x\sigma = \pi(\{s_1^x\sigma, \ldots, s_{k^x}^x\sigma\})$, où $s_i^x \in (\mathrm{QSub}(\mathcal{S}) \setminus \mathcal{X}) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$ et $s_i^x \neq \cdot(L)$. Montrons que si $y \in \mathrm{Vars}(s_i^x)$ pour certain i, alors $y \sqsubset x$. Supposons que $y \in \mathrm{Vars}(s_i^x)$ et $x \sqsubset y$. Alors $\mathrm{size}(x\sigma) = \mathrm{size}(\pi(\{s_1^x\sigma, \ldots, s_{k^x}^x\sigma\})) = \mathrm{size}(\ulcorner \cdot(s_1^x\sigma, \ldots, s_{k^x}^x\sigma) \urcorner) \geq$ (par Lemme 3) $\geq \mathrm{size}(\ulcorner s_i^x\sigma \urcorner) > \mathrm{size}(\ulcorner y\sigma \urcorner)$, parce qu'on sait que $s_i^x = \mathrm{bin}(p, q)$ or $s_i^x = \mathrm{priv}(p)$ et $y \in \mathrm{Vars}(s_i^x)$ (par exemple, pour le premier cas, $\mathrm{size}(\ulcorner s_i^x\sigma \urcorner) = \mathrm{size}(\mathrm{bin}(\ulcorner p\sigma \urcorner, \ulcorner q\sigma \urcorner)) = 1 + \mathrm{size}(\{\ulcorner p\sigma \urcorner, \ulcorner q\sigma \urcorner\})$ et comme $y \in \mathrm{Vars}(\{p, q\})$, en utilisant la propriété 20 de Lemme 3, on obtient $\mathrm{size}(\ulcorner s_i^x\sigma \urcorner) \geq 1 + \mathrm{size}(\ulcorner y\sigma \urcorner)$) Et comme $\mathrm{size}(\ulcorner y\sigma \urcorner) = \mathrm{size}(y\sigma)$, on a $y \sqsubset x$. Contradiction.

Nous alons montrer par induction la propriété principale de la lemme.

- soit $x = \min_{\sqsubset}(\mathrm{Vars}(\mathcal{S}))$. Alors $x\sigma = \pi(\{s_1^x\sigma, \ldots, s_{k^x}^x\sigma\}) = \ulcorner\cdot\left(s_1^x\sigma, \ldots, s_{k^x}^x\sigma\right)\urcorner$ et tous les s_i^x sont clos (car $\nexists y \sqsubset x$). Alors $x\sigma = \ulcorner\cdot\left(s_1^x, \ldots, s_{k^x}^x\right)\urcorner$. Nous avons $\mathrm{QSub}(x\sigma) = \left\{\ulcorner\cdot\left(s_1^x, \ldots, s_{k^x}^x\right)\urcorner\right\} \cup \mathrm{QSub}(s_1^x) \cup \cdots \cup \mathrm{QSub}\left(s_{k^x}^x\right) \subseteq \ulcorner\mathrm{QSub}(\mathcal{S})\,\sigma\urcorner \cup \mathrm{priv}(\mathcal{A} \cap \mathrm{QSub}(\mathcal{S}))$, car $\forall s \in \mathrm{QSub}(s_i^x)$, $s \in \mathcal{T}_g$ et $s \in \mathrm{QSub}(\mathcal{S})$ ou $s = \mathrm{priv}(a)$ ou $s = a$, où $a \in \mathrm{QSub}(\mathcal{S}) \cap \mathcal{A}$, donc $s = \ulcorner s\urcorner = s\sigma \in \ulcorner\mathrm{QSub}(\mathcal{S})\,\sigma\urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$ et $\ulcorner\cdot\left(s_1^x, \ldots, s_{k^x}^x\right)\urcorner = x\sigma \in \ulcorner\mathrm{QSub}(\mathcal{S})\,\sigma\urcorner$.

- Supposons pour tous $z \sqsubset y$, $\mathrm{QSub}(z\sigma) \subseteq \ulcorner\mathrm{QSub}(\mathcal{S}\sigma)\urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$.

- Montrons $\mathrm{QSub}(y\sigma) \subseteq \mathrm{QSub}(\mathcal{S}\sigma) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$.
  Nous savons que $y\sigma = \pi(\{s_1^y\sigma, \ldots, s_{k^y}^y\sigma\}) = \ulcorner\cdot\left(s_1^y\sigma, \ldots, s_{k^y}^y\sigma\right)\urcorner$ et $\forall z \in \mathrm{Vars}\left(s_i^y\right), z \sqsubset y$. Alors nous avons $\mathrm{QSub}(y\sigma) = \{y\sigma\} \cup \mathrm{QSub}\left(\ulcorner s_1^y\sigma\urcorner\right) \cup \cdots \cup \mathrm{QSub}\left(\ulcorner s_{k^y}^y\sigma\urcorner\right)$. Nous savons que $y\sigma \in \ulcorner\mathrm{QSub}(\mathcal{S})\,\sigma\urcorner$. Montrons que $\mathrm{QSub}\left(\ulcorner s_i^y\sigma\urcorner\right) \subseteq \ulcorner\mathrm{QSub}(\mathcal{S})\,\sigma\urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$. Par Lemme 3 on a $\mathrm{QSub}\left(\ulcorner s_i^y\sigma\urcorner\right) \subseteq \ulcorner\mathrm{QSub}\left(s_i^y\sigma\right)\urcorner \subseteq \ulcorner\mathrm{QSub}\left(s_i^y\right)\sigma \cup \mathrm{QSub}\left(\mathrm{Vars}\left(s_i^y\right)\sigma\right)\urcorner = \ulcorner\mathrm{QSub}\left(s_i^y\right)\sigma\urcorner \cup \mathrm{QSub}\left(\mathrm{Vars}\left(s_i^y\right)\sigma\right)$. On peut voir que $\ulcorner\mathrm{QSub}\left(s_i^y\right)\sigma\urcorner \subseteq \ulcorner\mathrm{QSub}(\mathcal{S})\,\sigma\urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$ (car $s_i^y \in \mathrm{QSub}(\mathcal{S}) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$); et par la supposition de l'induction et par la propriété prouvée ci-dessus nous avons $\mathrm{QSub}\left(\mathrm{Vars}\left(s_i^y\right)\sigma\right) \subseteq \ulcorner\mathrm{QSub}(\mathcal{S})\,\sigma\urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$. Donc, $\mathrm{QSub}(y\sigma) \subseteq \ulcorner\mathrm{QSub}(\mathcal{S})\,\sigma\urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$.

Proposition 7. *Pour un système de contraintes normalisé $\mathcal{S}$ qui a une solution conservatrice σ, $\forall x \in \mathrm{Vars}(\mathcal{S}), \mathrm{size}(x\sigma) \leq 2 \times \mathrm{size}(\mathcal{S})$.*

Proof. Comme $|\ulcorner\mathrm{Sub}(\mathcal{S})\,\sigma\urcorner| \leq |\mathrm{Sub}(\mathcal{S})\,\sigma| \leq |\mathrm{Sub}(\mathcal{S})| = \mathrm{size}(\mathcal{S})$, on a (en utilisant le fait que σ est normalisée et Lemme 8) $|\mathrm{Sub}(x\sigma)| = |\mathrm{QSub}(x\sigma)| \leq |\ulcorner\mathrm{QSub}(\mathcal{S})\,\sigma\urcorner \cup \mathrm{priv}(\mathcal{A} \cap \mathrm{QSub}(\mathcal{S}))| \leq |\ulcorner\mathrm{QSub}(\mathcal{S})\,\sigma\urcorner| + |\mathrm{priv}(\mathcal{A} \cap \mathrm{QSub}(\mathcal{S}))| \leq \mathrm{size}(\mathcal{S}) + |\mathcal{A} \cap \mathrm{QSub}(\mathcal{S})| \leq 2 \times \mathrm{size}(\mathcal{S})$; donc, $\mathrm{size}(x\sigma) \leq 2 \times \mathrm{size}(\mathcal{S})$. □

Proposition 8. *Soit $\mathcal{S}$ un système de contraintes satisfiable, il existe un modèle σ de $\mathcal{S}$ tel que $\forall x \in \mathrm{dom}(\sigma), \mathrm{size}(x\sigma) \leq 2 \times \mathrm{size}(\ulcorner\mathcal{S}\urcorner)$.*

Corollaire 2. *Un système de contraintes $\mathcal{S}$ est satisfiable si et seulement si il existe un modèle normalisé de $\mathcal{S}$ défini sur* $\mathrm{Vars}(\mathcal{S})$ *et qui mappe variables aux termes clos de $\mathcal{T}(\mathcal{A} \cap \mathrm{QSub}(\ulcorner\mathcal{S}\urcorner), \varnothing)$ avec la taille (*size*) qui ne dépasse pas* $2 \times \mathrm{size}(\mathcal{S})$.

En utilisant ce résultat, nous proposons un algorithme de satisfiabilité d'un système de contraintes (Algorithme 4).

Algorithm 2: Résolution d'un système de contraintes

Input: Un système de contraintes $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$
Output: Modèle σ, s'il en existe un; sinon $\bot$

Deviner pour chaque variable de $\mathcal{S}$ une valeur normalisée d'une substitution close σ ayant la taille qui ne dépasse pas $2 \times \mathrm{size}(\mathcal{S})$;
if *σ satisfait $E_i \triangleright t_i$ pour tous $i = 1, \ldots, n$* **then**
  return σ
else
  return $\bot$

Proposition 9. *L'algorithme 2 est correct et complet.*

Contraintes Dolev-Yao

Le résultat précédent sur la résolution de contraintes pour la théorie DY+ACI peut être projeté sur le cas DY classique (voir Tableau 4). Nous ne pourrions pas l'appliquer directement, puisque dans le modèle résultant nous pouvons avoir un symbole ACI qui n'est pas dans la signature de DY. Néanmoins, on peut prouver que le résultat fonctionne aussi pour le cas pure de Dolev-Yao.

Règles de composition	Règles de décomposition
$t_1, t_2 \to \mathrm{enc}(t_1, t_2)$	$\mathrm{enc}(t_1, t_2), t_2 \to t_1$
$t_1, t_2 \to \mathrm{aenc}(t_1, t_2)$	$\mathrm{aenc}(t_1, t_2), \mathrm{priv}(t_2) \to t_1$
$t_1, t_2 \to \mathrm{pair}(t_1, t_2)$	$\mathrm{pair}(t_1, t_2) \to t_1$
$t_1, \mathrm{priv}(t_2) \to \mathrm{sig}(t_1, \mathrm{priv}(t_2))$	$\mathrm{pair}(t_1, t_2) \to t_2$
$t_1, t_2 \to \mathrm{apply}(t_1, t_2)$	

Tableau 4: Règles du système de déduction DY

Corollaire 3. *Un système de contraintes $\mathcal{S}$ qui ne contient pas le symbole $\cdot$ est satisfiable dans DY ssi il est satisfiable dans DY+ACI.*

Analyse de complexité

Pour analyser la complexité des algorithmes que nous avons présentés, nous devons définir une mesure sur leurs entrées. Pour les termes et les ensembles de termes, nous utiliserons $\mathrm{size}(\cdot) + |\mathbb{E}(\cdot)|$, où $\mathbb{E}(\cdot)$ est un ensemble d'arêtes de représentation DAG de son argument. Autrement dit, nous utilisons ce que l'on appelle *taille DAG*. Pour les systèmes de contraintes tels que $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$ nous utiliserons $n \times \mathrm{size}(\mathcal{S}) + |\mathbb{E}(\mathcal{S})|$. La justification est donnée ci-dessous:

Définition 0.0.27 La *représentation DAG* d'un système de contraintes $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$ est un graphe taggé avec arêtes étiquetées $\mathbb{G} = \langle \mathbb{V}, \mathbb{E}, \mathrm{tag} \rangle$ ($\mathbb{V}$ est un ensemble de sommets et $\mathbb{E}$ est un ensemble d'arêtes; tag est une fonction de taggage définie sur $\mathbb{V}$) telle que:

- il existe une bijection $f : \mathbb{V} \mapsto \mathrm{Sub}(\mathcal{S})$;
- $\forall v \in \mathbb{V}\ \mathrm{tag}(v) = \langle s, m \rangle$, où
  - $s = \mathrm{root}(f(v))$;
  - m est un entier de $2n$-bit, où $m[2i-1] = 1$ ssi $f(v) \in E_i$ et $m[2i] = 1$ ssi $f(v) = t_i$.
- $v_1 \xrightarrow{1} v_2 \in \mathbb{E}$ ssi $\exists p \in \mathcal{T} : (\exists\, \mathrm{bin} : f(v_1) = \mathrm{bin}(f(v_2), p)) \vee f(v_1) = \mathrm{priv}(f(v_2))$;
- $v_1 \xrightarrow{2} v_2 \in \mathbb{E}$ ssi $\exists p \in \mathcal{T} : \exists\, \mathrm{bin} : f(v_1) = \mathrm{bin}(p, f(v_2))$;
- $v_1 \xrightarrow{i} v_2 \in \mathbb{E}$ ssi $f(v_1) = \cdot(L) \wedge L[i] = f(v_2)$;

Exemple 8 Un système de contraintes

$$\mathcal{S} = \begin{cases} \{\mathrm{enc}(a,x), \mathrm{pair}(b, \mathrm{enc}(a,a)), c\} & \triangleright a \\ \{\mathrm{priv}(b), c\} & \triangleright y \\ \{\mathrm{enc}(\mathrm{sig}(a, \mathrm{priv}(c)), y), \mathrm{aenc}(x,b)\} & \triangleright \mathrm{pair}(\mathrm{enc}(a,x), c) \end{cases}$$

sera représenté comme indiqué[6] sur la Figure 1. Les sommets de ce graphe représentent les éléments de $\mathrm{Sub}(\mathcal{S})$ représentés par leur symbole racine (première partie de ses tags) et les pointeurs vers ses fils.

Cette représentation peut être codée en ne dépassant pas $P(n \times |\mathbb{V}(\mathcal{S})| + |\mathbb{E}(\mathcal{S})|)$ bits d'espace, où n est le nombre de contraintes, $\mathbb{V}(\cdot)$ est l'ensemble de sommets et $\mathbb{E}(\cdot)$ est l'ensemble d'arêtes de la représentation DAG, et P est un polynôme à coefficients non négatifs. Comme il existe une bijection entre $\mathbb{V}(\mathcal{S})$ et $\mathrm{Sub}(\mathcal{S})$, nous obtenons $|\mathbb{V}(\mathcal{S})| = \mathrm{size}(\mathcal{S})$. Nous utiliserons donc $n \times \mathrm{size}(\mathcal{S}) + |\mathbb{E}(\mathcal{S})|$ comme une mesure de la taille d'un système de contraintes $\mathcal{S}$.

Remarquons que la représentation DAG d'un terme a une structure similaire.

[6] Label "1" (resp. "2") d'une arête est représentée par la gauche (resp. droit) du sommet-source

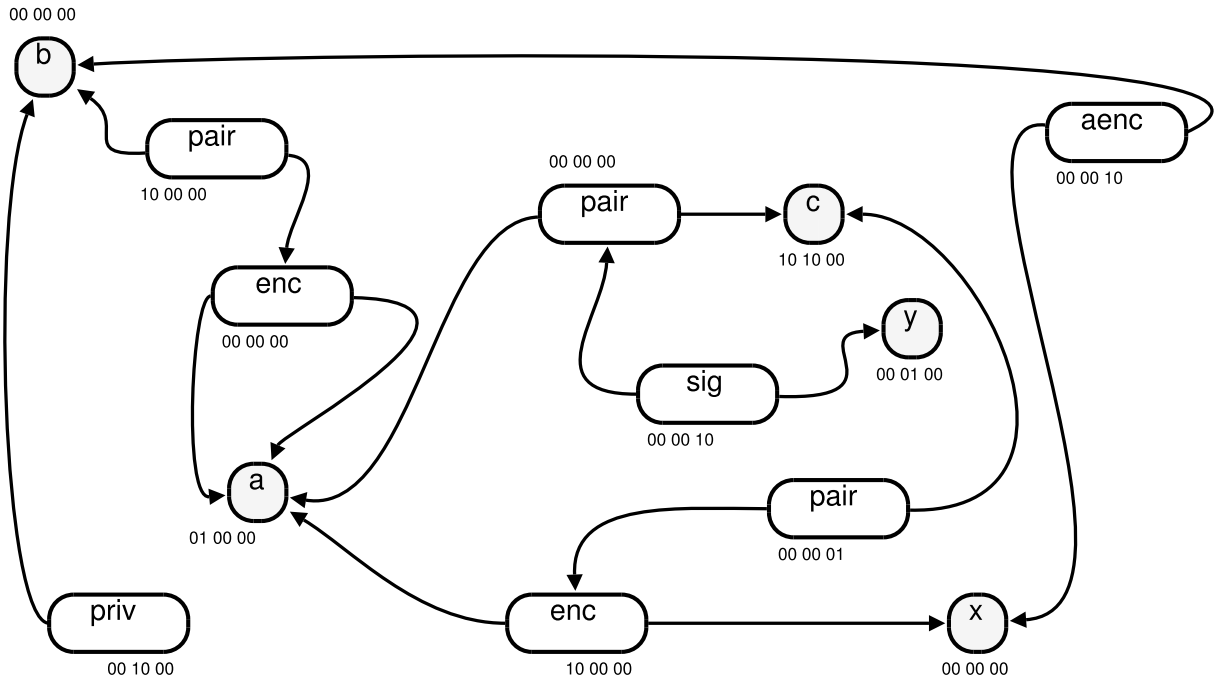


Figure 1: Représentation DAG du système de contraintes $\mathcal{S}$

Définition 0.0.28 La *mesure* d'un terme t est défini comme $\text{measure}(t) = \text{size}(t) + |\mathbb{E}(t)|$. Pour un système de contraintes $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$, sa mesure est $\text{measure}(\mathcal{S}) = n \times \text{size}(\mathcal{S}) + |\mathbb{E}(\mathcal{S})|$.

Lemme 9. *Pour un terme normalisé t, $|\mathbb{E}(t)| < (\text{size}(t))^2$. Pour un système de contraintes normalisé $\mathcal{S}$, $|\mathbb{E}(\mathcal{S})| < (\text{size}(\mathcal{S}))^2$.*

Proposition 10. *L'algorithme 1 a une complexité polynomiale en* $\text{size}(E \cup \{t\})$.

Preuve. Remarquons que pour chaque pas de l'algorithme, $|S|$ et $|D|$ ne dépassent pas $|\text{QSub}(E \cup \{t\})|$. La construction de $\text{QSub}(E) \cup \text{QSub}(t)$ prend temps linéaire en $\text{size}(E \cup \{t\})$. Le temps de construction de S est un $O(|E| \times |\text{QSub}(E) \cup \text{QSub}(t)|)$, i.e. inférieur à $O\left((\text{size}(E \cup \{t\}))^2\right)$.

La boucle principale a au maximum $|\text{QSub}(E \cup \{t\})| - |E|$ pas. La recherche d'une règle DY avec sa partie gauche dans D et partie droite dans S ne dépasse pas $O(|S| \times |D|^2)$ et donc, ne dépasse pas $O((\text{size}(E \cup \{t\}))^3)$. Le `if` suivant peut être effectué en $O(|S| \times |D| \times (\text{size}(E \cup \{t\})))$ pas et le dernier `if` peut être tenté en temps cubique. Le test faite dans `return` est linéaire. Finalement, grâce à la propriété 17 du Lemme 3, nous obtenons la complexité annoncée. □

Lemme 10. *Soit t un terme. La normalisation peut être réalisée en temps polynomial en* $\text{measure}(t)$. *De même pour un système de contraintes: la normalisation peut être réalisée en temps polynomial en* $\text{measure}(\mathcal{S})$.

Idée de la preuve pour le cas de termes. Un algorithme de normalisation fonctionne de bas en haut par aplatissement des ensembles ACI imbriqués, le tri des enfants de nœuds ACI, fusion les nœuds dupliqués, suppression les arêtes inutiles dupliquées et élimination les nœuds sans arêtes entrantes (sauf la racine de t). □

Proposition 11. *Le problème de la satisfiabilité des systèmes généraux de contraintes dans DY+ACI (résolu par Algorithme 2) est dans NP.*

Idée de la preuve. L'algorithme 2 retourne une preuve pour la problème de décision, si elle existe. Nous devons montrer que la vérification de cette preuve prend un temps polynomial par rapport à la mesure de l'entrée. Pour ce faire, nous allons normaliser $\mathcal{S}\sigma$ et ensuite appliquer l'algorithme de vérification de dérivabilité pour les termes clos. En utilisant le fait que $\text{size}(x\sigma) \leq 2 \times \text{size}(\mathcal{S})$ et que les complexités de la procédure de normalisation et la dérivabilité pour les termes clos sont polynomiales, nous pouvons sur-approximer le temps d'exécution par un polynôme en $\text{measure}(\mathcal{S})$. □

D'autre part, nous pouvons réutiliser une technique présentée dans [RT03] pour montrer que la satisfiabilité d'un système de contrainte est un problème NP-dur. En conséquence,

Théorème 2. *La satisfiabilité de systèmes généraux de contraintes DY+ACI est NP-complète.*

De plus,

Corollaire 4. *La satisfiabilité de systèmes généraux de contraintes DY est NP-complète.*

On peut aussi montrer que pour certain type de systèmes de déduction (par exemple, des théories sous-terme convergentes) le problème de la satisfiabilité de systèmes généraux de contraintes est indécidable (même pour la classe qui préserve soit la monotonie des connaissances soit origine de variables). Par contre, pour les systèmes bien formés la satisfiabilité peut être décidable (voir [Bau05] pour les théories sous-terme convergentes).

Composition de services Web

Les services Web mettent en œuvre une Architecture Orientée Services (angl.: SOA) [Gro06, OASb]. Un des principes directeurs de la SOA est la *composabilité*, c'est-à-dire, la possibilité de créer des services plus complexes à partir de services déjà existants. Ces nouveaux services doivent fournir de nouvelles fonctionnalités qui n'ont été proposées par aucun service existant. Par exemple, considérons une situation. Un agent de voyage reçoit une demande d'un client pour organiser un voyage contenant la date et l'itinéraire du voyage. Le client attend de l'agent de voyage de réserver un vol, un hôtel et une voiture dans chaque ville où il reste. L'agent de voyages doit faire les réserves correspondantes en utilisant des services disponibles comme des services Web d'Hôtel, de Vols, et de location de voiture en envoyant des requêtes appropriées, et en cas de succès, de retourner au client un rapport intégral avec les détails des réservations. Comme il s'agit d'une procédure fréquente à faire par l'agent de voyages, il pourrait l'organiser dans un service Web de telle manière que le client envoie simplement sa demande à ce nouveau service Web de Voyages et toutes les réservations nécessaires sont effectuées automatiquement. Ce service est composé, car il réutilise des services Web déjà existants, et, comme on peut le voir, il fournit de nouvelle fonctionnalité qui n'était proposé par aucun service mentionné ci-dessus.

Nous allons discuter d'une façon automatique de creéation d'un tel service composé. Notez quelques difficultés ici: puisque les services existants ne peuvent accepter que les demandes conformes à leurs spécifications, le service composé doit être capable de construire de telles requêtes, par exemple, en adaptant les données reçues du client.

Principalement, il existe deux approches pour composer des services: la chorégraphie et l'orchestration [Pel03b]. Dans l'approche chorégraphique il n'existe aucune entité centrale et chaque service Web est responsable pour mettre en œuvre sa part du service composé, la communication se fait directement entre les services. En revanche, lors d'une orchestration, toutes les communications passent par un *médiateur* (parfois aussi appelé *orchestrateur*), un noyau du processus qui regroupe les services existants afin de fournir une nouvelle fonctionnalité.

Nous allons penser à un service comme à un rôle de protocole, qui est modélisée en termes de *strand*s [FJHG99] (en bref, une séquence finie d'actions recevoir/répondre), où les messages sont représentés en tant que des termes de premier ordre. Ainsi, nous couverons deux aspects des services Web: leur description en termes de WSDL [Wor01] (actions) et leur protocole de comportement [BIPT09] (ordre qu'il faut respecter lors d'invoquer des opération de service).

Outre la liste des opérations (avec leurs patterns de messages d'entrée et de sortie), une description WSDL peut spécifier des liaisons sécurisées (comme HTTP sur SSL) et aussi une politique de sécurité en utilisant le standard WS-SecurityPolicy (WSSP) [DLHBH+02]. Par le biais de WSSP, un fournisseur de service peut fixer une politique de sécurité sur les parties des messages qui sont spécifiés dans les opérations, par exemple spécifier les parties des messages qui doivent être signées, cryptées, etc.

Bien que la majorité des travaux sur la composition automatique de services Web ne prennent pas en compte les politiques de sécurité lors de l'adaptation des messages (par exemple,

[BCD+05]), ou ne les adapte pas du tout (par exemple [BCF08, CGL+08, BCGP08, Pad08, BFHS03])[7], nous allons aller plus loin et considérons également une capacité de déduction pour le médiateur qui lui permet de ne pas procéder que par transfert des messages, mais aussi d'effectuer une adaptation de messages complexes satisfaisant les politiques de sécurité des services disponibles en plus des exigences de fonctionnalité. Ainsi, en utilisant les messages collectés lors de la communication, un médiateur peut décomposer les messages en pièces, les regrouper, en construire de nouvelles et les fournir aux services Web disponibles pour obtenir une réponse qui ne pourrait probablement pas être obtenue par des procédures plus simples.

Nous présentons une approche évolutive de la composition de services Web, que nous appelons *l'orchestration* distribuée, s'appuyant sur la notion de *partenaire* correspondant à une organisation. Nous avons dédié un partenaire pour communiquer avec le client et nous l'appelons un médiateur. Chaque partenaire dans la composition met en œuvre sa propre partie de l'orchestration (similaire aux *serveurs de chorégraphies* dans [PE09]). Dans ce cadre l'orchestration standard est un cas particulier dans lequel un seul partenaire est impliqué, alors que la chorégraphie est un autre cas dans lequel il y a un partenaire par chaque service disponible.

Cependant, même dans une telle coopération des données sensibles ne devraient pas être propagées au-delà des frontières organisationnelles (une entreprise ne partagera pas des secrets avec partenaires). C'est pourquoi nous allons introduire certaines restrictions sur la communication entre les partenaires. Nous allons montrer comment l'orchestration distribués est possible dans ce cadre restreint.

Plusieurs notions d'"orchestration distribuée" ont été préconisé dans la littérature (par exemple [BMM06]). Cependant, dans les processus inter-organisationnels, il est crucial de protéger les données sensibles de chaque organisation fournissant un service-composant dans certain composition, et notre motivation principale est de faire progresser l'état de l'art en prenant en compte des politiques de sécurité (y compris un *politique de non-divulgation*), pendant le calcul d'une composition.

Pour le cas non-distribué et sans mise en œuvre certaines idées initiales ont été présentées dans [CMR08] et [CMR09]. Plus tard, le cas non-distribué a été implanté sous le nom de AVANTSSAR Orchestrator [AVA10].

Représentation des services et de leurs politiques

Nous représentons un service Web par une séquence d'*actions* à exécuter, où une action est soit une réception soit une émission d'un pattern de message. Pourquoi un pattern? Puisque nous utilisons le modèle de comportement, les actions ultérieures exécutée par le service Web dépendrait des valeurs qui sont reçues.

Exemple 9 Considérons un service Web "identité". Si le service a reçu une valeur a, alors il doit retourner a, Mais s'il a reçu b, il doit retourner b, etc. Ceci est fait en utilisant des *variables*. Ce service Web d'identité est donc défini par "recevoir X, envoyez X".

Notez que la valeur de la variable une fois instanciée ne sera pas seulement utilisé à l'intérieur d'un opération du service Web, mais aussi dans toutes les opérations restantes.

Nous écrirons $?r$ pour la réception d'un message (ou pattern de message) r et $!s$ pour l'envoi message (pattern de message) s. Nous appelons une séquence finie d'actions un *strand* [FHG98, FHG99]. Un strand ayant un nombre d'actions pair, et qui commence par ? et alterne toujours ? avec ! est appelé un *strand* normal et représente un service Web *synchrone*, i.e. un service

[7] En revanche, ces travaux considèrent le comportement de services plus riche.

Web pour lequel chaque requête implique une réponse immédiate. Pour raison de simplicité du modèle formel on ne considère que des services disponibles synchrones.

Définition 0.0.29 Une *action* ρ est:

- soit une réception $?_f r$, où r (cf. "receive") est un terme exprimant un pattern acceptable pour un message entrant, et f (cf. "from") est un expéditeur attendu.
- ou une émission $!_t s$, où s (cf. "send") est un terme exprimant un pattern pour un message à émettre, et t (cf. "to") est un récepteur prévu.

Dans le cas où l'expéditeur (resp. récepteur) n'est pas connu ou n'est pas pertinent, nous allons écrire $!s$ (resp. $?r$).

Pour une substitution σ on définit

$$\rho\sigma = \begin{cases} ?_f(r\sigma), & \text{si } \rho = ?_f r; \\ !_t(s\sigma), & \text{si } \rho = !_t s. \end{cases}$$

Une action est clos si le terme qu'elle contient est clos.

La fonctionnalité d'un service ainsi que la politique de sécurité sont entièrement encodés dans ses patterns de messages d'entrée-sortie.

Exemple 10 Un service Web qui calcule le sinus d'une valeur donnée, et dont la politique de sécurité affirme que les messages entrants doivent être chiffrés avec sa clé publique K est représenté comme $?\,\mathrm{aenc}(X, K).!sin(X)$, alors que sans la politique, il serait représenté comme $?X.!sin(X)$.

Nous adoptons la terminologie de [BCD+03, BCD+05] et appelons *médiateur* un service qui adapte et distribue les requêtes d'un client à la communauté des services disponibles. Il peut effectuer des opérations DY sur les messages afin de les adapter. Par ailleurs, nous supposons que le médiateur possède une certaine connaissance initiale (ensemble fini de messages) qu'il peut aussi utiliser. En plus d'être un service, nous attendons du médiateur d'être *exécutable*, i.e., il doit être capable de construire tous les messages qu'il envoie à partir de ses connaissances actuelles (c'est à dire ses connaissances initiales ainsi que les messages recueillis jusqu'au moment de l'envoi du message) et en utilisant uniquement les opérations de DY (voir, par exemple, le Tableau 2 à la page xxvi).

Dans le cas de l'orchestration distribuée, où plusieurs entités sont considérées comme des médiateurs, nous les appellerons *partenaires*. Ces partenaires peuvent posséder des connaissances différentes qu'ils ne veulent pas partager mais peuvent utiliser afin d'atteindre des objectifs communs. Par ailleurs, ils peuvent avoir accès aux différents services qui ne sont pas disponibles pour les autres.

Orchestration distribuée

Exemple

Supposons qu'il y ait une demande de traduction de textes du français vers l'anglais. Supposons également que ces textes ont été obtenus par une procédure de reconnaissance automatique des documents écrits à la main, et donc, ils contiennent certaines fautes.

L'utilisateur veut envoyer un texte en français et recevoir sa traduction anglaise. Bien sûr, afin de traduire un texte, celui-ci ne devrait pas contenir d'erreurs, donc, le texte reconnu automatiquement devrait être corrigé avant. Nous pouvons l'exprimer en utilisant la spécification suivante:

$$!t.\,?en(corr(t)).$$

Nous supposons l'existence de deux services qui sont disponibles: SpellChecker — un service qui corrige l'orthographe et Translator — un service qui fait une traduction automatique d'un texte donné du français vers l'anglais. Chaque service a une liste d'utilisateurs enregistrés et ne sert que ceux-ci. Par simplicité nous considérons que chaque service n'a qu'un seule utilisateur enregistré. En plus, la politique de sécurité de Translator exige que tous les messages entrants soient chiffrés avec sa clef public.

Dans notre formalisme,

1. Spécification de SpellChecker:

$$\begin{aligned} ?\,\mathrm{aenc}\left(\mathrm{pair}\left(\mathrm{pair}\left(usr_{sc}, pwd_{sc}\right), T\right), K_{sc}\right).\\ !\,\mathrm{pair}\left(corr(T), n(T)\right), \end{aligned}$$

   où T est un texte, $corr(T)$ est le texte corrigé, $n(T)$ est le nombre de corrections effectuées, usr_{sc} est un login d'un utilisateur enregistré et pwd_{sc} et son mot de passe; K_{sc} est la clef public de SpellChecker.

2. Spécification de Translator:

$$?\,\mathrm{aenc}\left(\mathrm{pair}\left(\mathrm{pair}\left(usr_{tr}, pwd_{tr}\right), M\right), K_{tr}\right).!en(M),$$

   où M est un texte, $en(M)$ est une translation de M en anglais, usr_{tr} est un login d'un utilisateur enregistré et pwd_{tr} et son mot de passe; K_{tr} est la clef public de Translator.

Nous supposons aussi qu'il y a deux partenaires qui collaborent pour satisfaire le client. L'un est appelé Mediateur, celui peut contacter directement le client; l'autre partenaire — non. Les deux partenaire partagent une clé symétrique k et toutes les communications entre eux doivent être chiffrées avec cette clé partagée. En plus, tous les deux connaissent la clé publique de Translator K_{tr}. Par contre, il y a des connaissances différentes: Mediateur est un utilisateur enregistré de SpellChecker (et alors connaît usr_{sc} et pwd_{sc}) et non de Translator, mais Partenaire fait l'inverse: il est un utilisateur de Translator mais pas de SpellChecker (il connaît usr_{tr} et pwd_{tr}).

Le problème est de trouver un scénario de communication possible entre toutes ces parties, tel que *(i)* toutes les demandes du client sont satisfaits et *(ii)* aucun partenaire ne peut extraire une des données sensibles d'un autre partenaire à partir d'un message reçu de ce dernier. Dans notre exemple, nous pouvons considérer pwd_{sc} et pwd_{tr} comme sensibles pour Médiateur et Partenaire respectivement. Nous allons présenter une *condition de non-divulgation* qui est suffisant pour assurer *(ii)*.

Idée d'une solution Nous pouvons résoudre ce problème si le nombre d'interactions est borné. Pour cet exemple nous supposons que nous pouvons utiliser tous les services disponibles au plus une fois, et que nous permettons un seul tour de communication entre le médiateur et le partenaire (le médiateur envoie une requête et le partenaire répond).

Nous devons choisir un entrelacement des actions de client et des partenaires et des invocations de services disponibles (le nombre de tous les entrelacements possibles est fini). Par exemple, l'entrelacement choisi est:

$$\left\{\begin{array}{lll}
\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t\} \triangleright & \text{aenc}\left(\text{pair}\left(\text{pair}\left(usr_{sc}, pwd_{sc}\right), T\right), K_{sc}\right) & (1)\\
\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}\left(corr(T), n(T)\right)\} \triangleright & \text{enc}\left(X, k\right) & (2)\\
\{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}\left(X, k\right)\} \triangleright & \text{aenc}\left(\text{pair}\left(\text{pair}\left(usr_{tr}, pwd_{tr}\right), M\right), K_{tr}\right) & (3)\\
\{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}\left(X, k\right), en(M)\} \triangleright & \text{enc}\left(Y, k\right) & (4)\\
\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}\left(corr(T), n(T)\right), \text{enc}\left(Y, k\right)\} \triangleright & en(corr(t)) & (5)
\end{array}\right.$$

où T, M, X, Y sont des variables.

Figure 1: Un système de contraintes décrivant une orchestration distribuée possible.

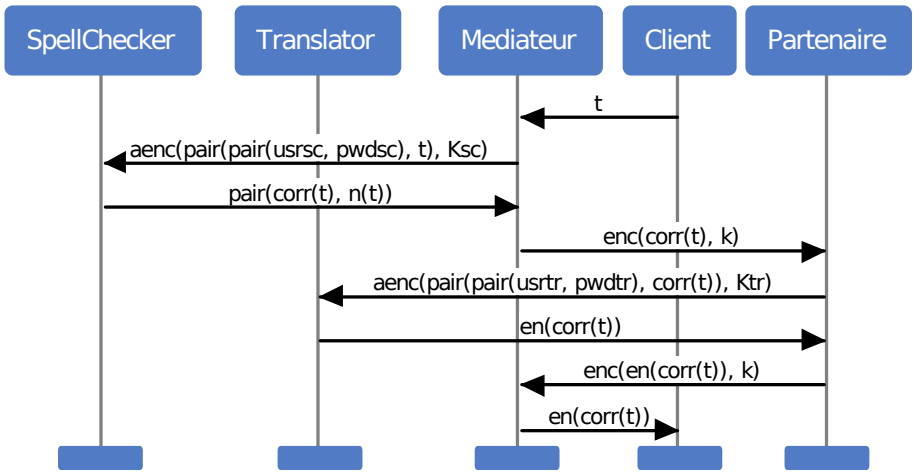


Figure 2: Solution pour l'instance d'un problème de l'orchestration distribuée

1. Client → Mediateur
2. Mediateur ↔ SpellChecker
3. Mediateur → Partenaire
4. Partenaire ↔ Translator
5. Partenaire → Mediateur
6. Mediateur → Client

où $A \to B$ est synonyme de "A envoie et B reçoit", et $A \leftrightarrow B$ est synonyme de "B est invoqué par A". Le système de contraintes correspondant, qui exprime la possibilité d'interaction qui mène à la satisfaction du client et suit l'entrelacement sélectionné, est représenté dans la Figure 1.

Par exemple, Contrainte (1) dit qu'à partir des connaissances du Médiateur (après qu'il a reçu un message t à partir du client), il doit construire un message conformant avec le pattern qui peut être accepté par le SpellChecker afin de l'invoquer.

On peut voir que le système de contraintes obtenu n'est pas bien formé. Une des solutions possibles que nous pouvons obtenir par la procédure de décision décrite précédemment, est la suivante:

$$\sigma = \{T \mapsto t, M \mapsto corr(t), X \mapsto corr(t), Y \mapsto en(corr(t))\}.$$

Cette solution représente une conversation décrite sur la Figure 2.

Notez que ni $X\sigma$ ni $Y\sigma$ (soit les messages envoyés entre le Médiateur et le Partenaire) ne contiennent de mot de passe pwd_{tr}, pwd_{sc}. Cela signifie que dans cette solution aucune information sensible n'a été divulguée par le Médiateur au Partenaire ou inversement. Pourtant il existe d'autres solution (qui nécessitent d'autres entrelacements) dans lesquelles le client est toujours satisfait, mais une information sensible a été divulguée.

Modèle

Informellement, le problème d'orchestration distribuée est énoncé comme suit: étant donnée une communauté des services disponibles, un client, un ensemble de partenaires chacun muni de ses connaissances initiales, un ensemble de voies de communication entre les partenaires et l'accessibilité des services aux partenaires, il faut trouver un scénario de communication possible entre les partenaires, les services disponibles et le client, tel que le client atteigne son état final (i.e. exécute toutes ses actions). Nous considérons l'accessibilité des services aux partenaires afin de passer plus facilement de l'orchestration distribuée à la chorégraphie, si nécessaire, puisque dans la chorégraphie chaque partenaire doit être en mesure d'accéder uniquement à son propre service, tandis que l'accès à d'autres services devraient se faire via le partenaires "attachés" à ces services.

Dans ce scénario de communication des messages diffusés entre les partenaires doivent être envoyés par les voies de communication (qui forment un ensemble prédéfini des voies de communication) et ils doivent être compatibles avec les patterns de messages définis pour chaque voie de communication. En outre, certaines politiques de non-divulgation imposées à la communication sont spécifiés comme un ensemble de données atomiques sensibles par chaque partenaire, qui ne devraient pas être extractibles à partir de messages envoyés à d'autres partenaires.

Pour assurer cette dernière condition nous allons considérer une propriété plus forte: Nous *ne permettons pas l'occurrence des données sensibles comme un sous-terme de ces messages.* En effet, comme les règles de déduction dans le Tableau 4 (page xxxii) ne produisent pas de nouveaux atomes, un partenaire n'est pas capable d'extraire des données sensibles à partir d'un message qui n'en contient pas.

Donc, nous supposons que nous nous sommes donnés:

- Un ensemble de services disponibles $\mathbb{S} = \{S_1, \ldots, S_n\}$.
  Service disponible S_i est représentée par son nom et un strand normal, i.e. $S_i = \langle i, A_i \rangle$, où $A_i = ?r^i_1.!s^i_1. \ldots . ?r^i_{e_i}.!s^i_{e_i}$.

- Un client C.
  Nous pouvons penser que le client est un service autonome disponible $\langle 0, A_0 \rangle$, mais A_0 est un strand pas nécessairement normal.

- Un ensemble de partenaires $\mathbb{P} = \{P_1, \ldots, P_k\}$ (P_1 est un Mediateur) et pour chaque partenaire P_i, un ensemble d'atomes sensibles N_i qu'il ne veut pas partager avec les autres partenaires. Partenaire P_i est alors représenté par son nom i, sa connaissance actuelle K_i et un ensemble de valeurs atomiques sensibles $N_i \subseteq \mathrm{Sub}(K_i)$, i.e. $P_i = \langle i, K_i, N_i \rangle$.

- un ensemble de voies de communication $\mathbb{C} = \{C_1, \ldots, C_u\}$ entre les partenaires.
  Voie de communication C_i est un tuple $i \xrightarrow{p} j$, où i et j sont des noms de partenaires P_i et P_j respectivement et tous les messages envoyés par cette voie de P_i vers P_j doivent être compatibles avec le pattern p. Nous supposons que $\mathrm{Sub}(p) \cap N_i = \varnothing$.

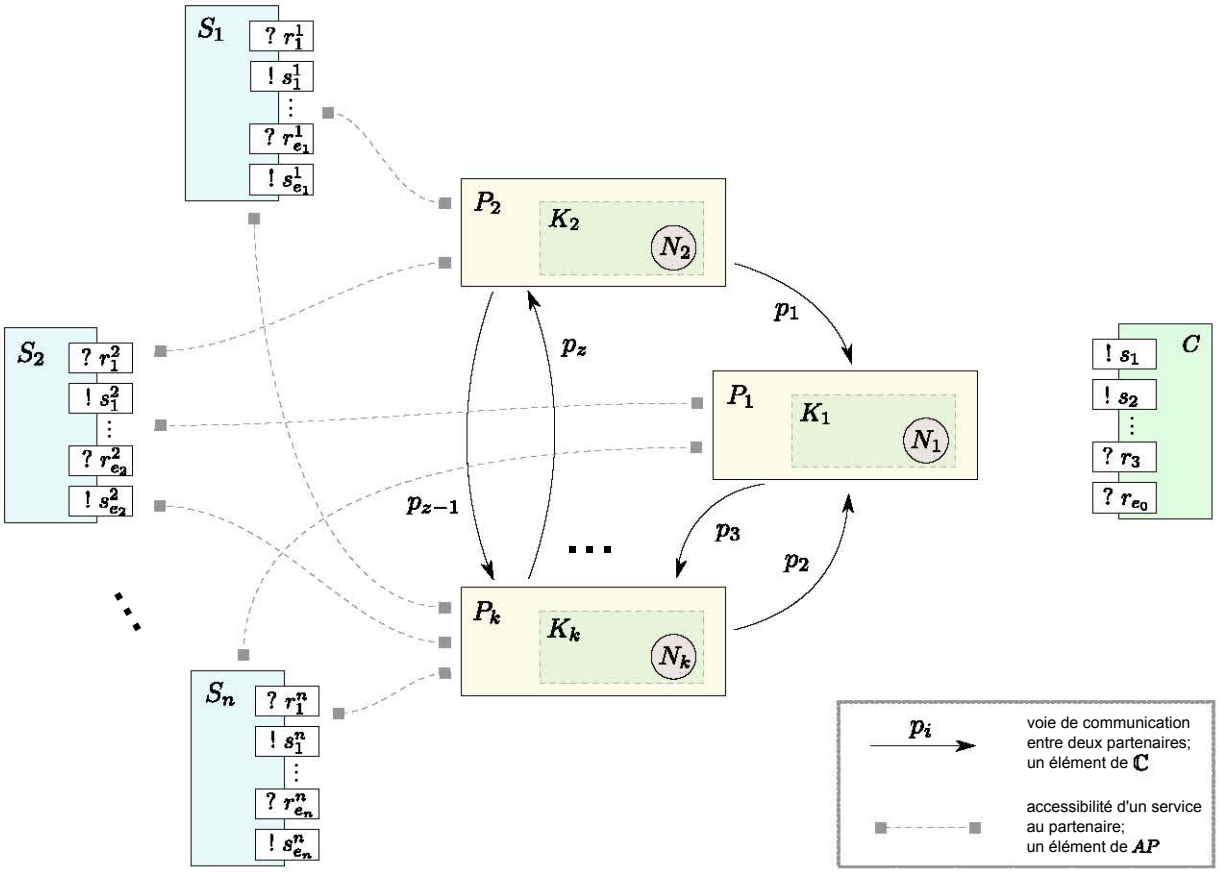


Figure 3: Exemple d'entrée générique pour le problème de l'orchestration distribuée

- Un ensemble de services accessibles par chaque partenaire $AS = \{\langle p_i, s_i\rangle_{i=1,\ldots,v}\}$.
  Le partenaire P_{p_i} peut accéder au service disponibles S_{s_i} ssi $\langle p_i, s_i\rangle \in AS$.

- Une borne supérieure sur le nombre d'interactions m.
  Nous permettons au maximum m événements de communication, comme l'invocation d'un service disponible, la transmission d'un message d'un partenaire à un autre, la transmission d'un message du client au médiateur ou l'inverse.

Nous supposons aussi que $\text{Vars}(S_i) \cap \text{Vars}(S_j) = \varnothing$, si $i \neq j$ et pour tout i, $\text{Vars}(S_i) \cap \text{Vars}(C) = \varnothing$.

L'entrée générique est illustré dans Figure 3 (la borne supérieure m n'est pas représenté).

Comme nous l'avons mentionné précédemment, l'orchestration distribuée a deux cas particuliers: l'orchestration et la chorégraphie.

L'entrée pour la première est illustrée sur la Figure 4: Il n'y a qu'un seul médiateur P_1 (et pas d'autres partenaires), par conséquent, $\mathbb{C} = \varnothing$ et pas besoin de N_1. Par ailleurs, tous les services sont accessibles par le médiateur: $AS = \{\langle 1, i\rangle_{i=1,\ldots,n}\}$.

Pour ce dernier l'entrée est représenté dans la Figure 5: il n'y a qu'un partenaire pour chaque service disponible, $k = n + 1$, et ce service n'est accessible que pour le partenaire correspondant, i.e. $AS = \{\langle i + 1, i\rangle\}_{i=1,\ldots,n}$.

Nous définissons une *condition de non-divulgation* (ou *politique de non-divulgation*) en fonction de ce que nous avons déjà annoncé auparavant: un atome sensibles d'un partenaire ne se produit jamais comme un sous-terme d'un message émis par lui. Nous allons imposer cette poli-

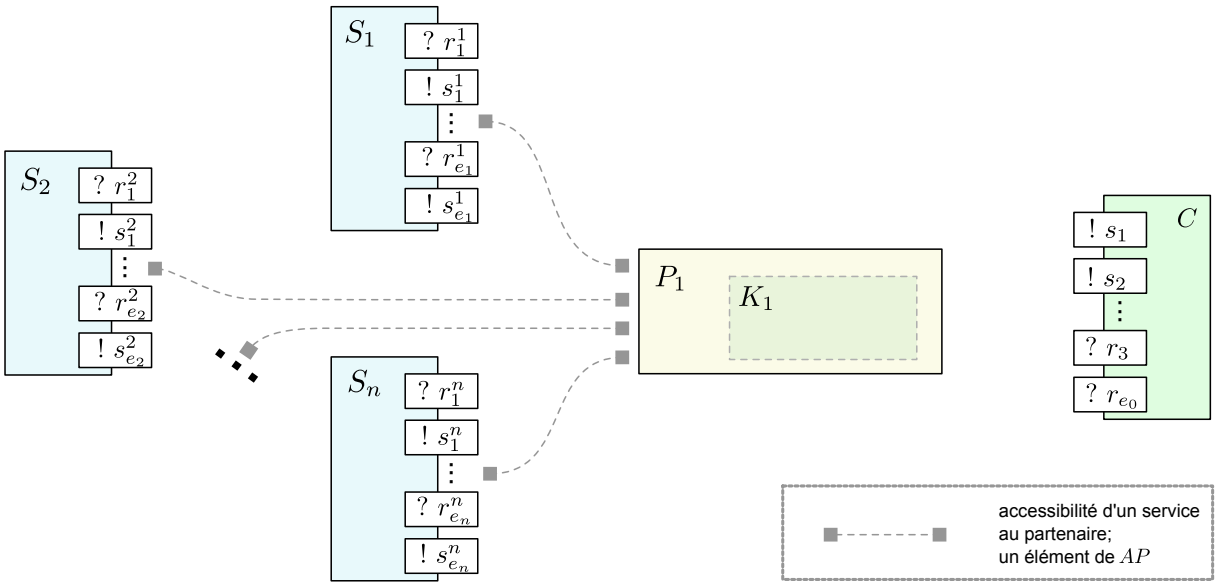


Figure 4: Un cas particulier de l'entrée: le problème d'orchestration

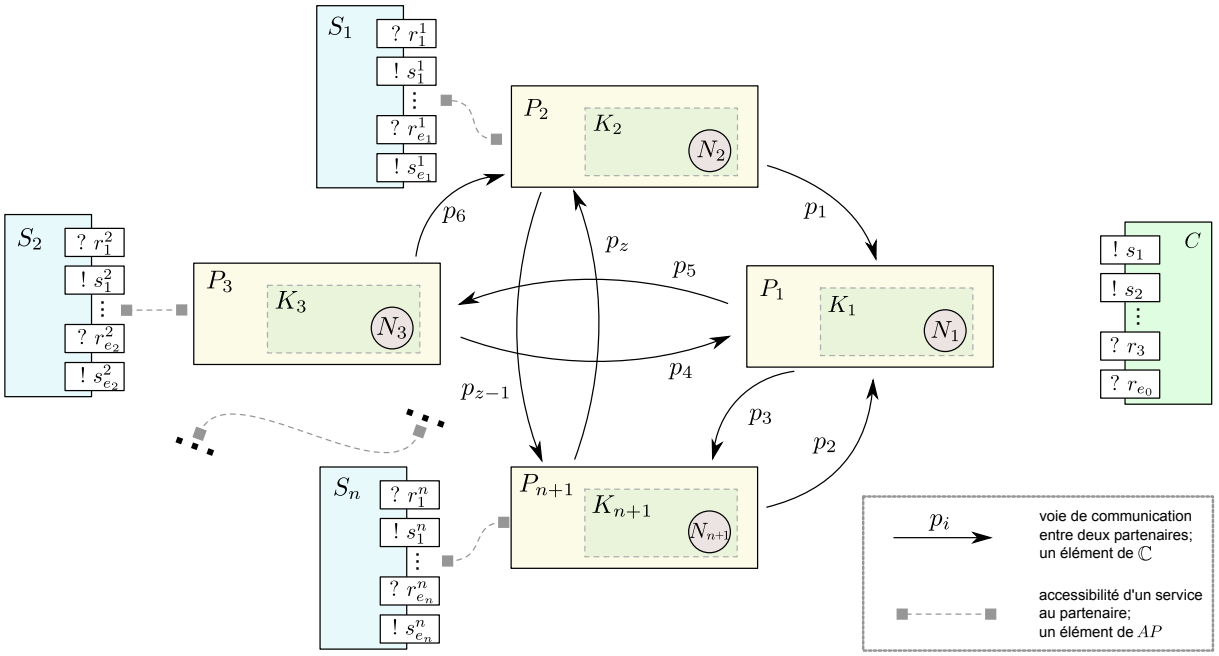


Figure 5: Un cas particulier de l'entrée: le problème de chorégraphie

$$\frac{\left\langle \left\{ \left\langle j, ?r.\,!s.\,A'_j \right\rangle \right\} \cup S', C, \{\langle i, K_i, N_i\rangle\} \cup P', \mathcal{S}, \mathcal{H} \right\rangle \qquad [\text{if } \langle i,j\rangle \in AS]}{\left\langle \left\{ \left\langle j, A'_j \right\rangle \right\} \cup S', C, \{\langle i, K_i \cup \{s\}, N_i\rangle\} \cup P', \mathcal{S} \cup \{K_i \triangleright r\}, \mathcal{H} \right\rangle} \tag{6}$$

$$\frac{\langle S, \langle 0, !s.\,A'\rangle, \{\langle 1, K_1, N_1\rangle\} \cup P', \mathcal{S}, \mathcal{H}\rangle}{\langle S, \langle 0, A'\rangle, \{\langle 1, K_1 \cup \{s\}, N_1\rangle\} \cup P', \mathcal{S}, \mathcal{H}\rangle} \tag{7}$$

$$\frac{\langle S, \langle 0, ?r.\,A'\rangle, \{\langle 1, K_1, N_1\rangle\} \cup P', \mathcal{S}, \mathcal{H}\rangle}{\langle S, \langle 0, A'\rangle, \{\langle 1, K_1, N_1\rangle\} \cup P', \mathcal{S} \cup \{K_1 \triangleright r\}, \mathcal{H}\rangle} \tag{8}$$

$$\frac{\langle S, C, \{\langle i, K_i, N_i\rangle, \langle j, K_j, N_j\rangle\} \cup P', \mathcal{S}, \mathcal{H}\rangle \qquad [\text{if } i \xrightarrow{p} j \in \mathbb{C};\ q = \text{refresh}(p)]}{\left\langle S, C, \{\langle i, K_i, N_i\rangle, \langle j, K_j \cup \{q\}, N_j\rangle\} \cup P', \mathcal{S} \cup \{K_i \triangleright q\}, \mathcal{H} \cup \{\text{Sub}(q) \cap N_i \stackrel{?}{=} \varnothing\} \right\rangle} \tag{9}$$

où refresh(t) est un terme obtenu en remplaçant toutes les variables de t par les fraîches.

Figure 6: Système de transitions

tique uniquement sur les messages émis par un partenaire à l'autre, tandis que la communication avec les services disponibles est exempt de cette condition.

Définition 0.0.30 Une *condition de non-divulgation* $\mathcal{H}$ est un ensemble d'équations $\left\{\text{Sub}(m_i) \cap N_{j_i} \stackrel{?}{=} \varnothing\right\}_{i=1,\ldots,l}$, où m_i est un terme et N_{j_i} est un ensemble d'atomes. On dit qu'une substitution close σ est une solution de (ou satisfait) $\mathcal{H}$ ssi pour tout $i = 1,\ldots,l$ une égalité $\text{Sub}(m_i\sigma) \cap N_{j_i} = \varnothing$ a lieu. Nous appellerons $\mathcal{H}$ *satisfiable* s'il admet une solution.

Maintenant, nous pouvons présenter une *configuration* qui reflète un état global de l'orchestration distribuée.

Définition 0.0.31 Une *configuration* est un tuple $\langle \{S_1, S_2, \ldots, S_n\}, C, \{P_1, \ldots, P_k\}, \mathcal{S}, \mathcal{H}\rangle$, soit un ensemble d'états actuels des services disponibles, un état du client, un ensemble d'états des partenaires, un système de contraintes et une condition de non-divulgation à être satisfaits.

Nous définissons un ensemble de transitions dans la Figure 6 qui permettent de passer d'une configuration à une autre.

La transition 4.9 exprime le fait que partenaire $P_i = \langle i, K_i, N_i\rangle$ peut invoquer un service disponible $S_j = \langle j, A_j\rangle$, ssi il est accessible et P_i est capable de construire un message (terme clos) qui soit compatible avec le pattern attendu. La réponse de S_j sera ajouté aux connaissances du partenaire. De même pour l'échange de messages entre Médiateur P_1 et le Client C, sauf que le Client peut initier un envoi; ici nous avons deux transitions: l'un pour l'envoi et un pour la réception (Transitions 7, 8). Un partenaire $P_i = \langle i, K_i, N_i\rangle$ peut envoyer un message à un partenaire P_j, ssi il existe une voie de communication $C_{ij} = i \xrightarrow{p} j \in \mathbb{C}$ entre eux telle que le partenaire P_i peut construire un message conforme à un pattern p et ce message ne contiendra pas de données sensibles (éléments de N_i) comme sous-terme (Transition 9). Dans la dernière transition, nous avons utilisé la fonction refresh$(\cdot)$ qui remplace tous les noms de variables dans un terme donné par des neufs. C'est fait pour rendre les messages envoyés par le même voie de communication indépendants.

On peut voir que la configuration contient certaines conditions à satisfaire: un système de contraintes et une condition de non-divulgation. Ainsi, passer d'une configuration contenant des conditions qui peuvent être ou sont déjà satisfaites à une autre en utilisant une transition peut

ajouter de nouvelles conditions qui éventuellement ne peuvent pas être satisfaites ou qui ne sont pas compatibles avec celles de la configuration initiale.

C'est pourquoi nous introduisons deux notions:

- *exécution symbolique*, où l'on ne s'intéresse pas à la faisabilité d'un pas, mais où il suffit d'appliquer les syntaxiquement transitions de la Figure 6 et reporter les problèmes sous la forme de conditions à satisfaire, et
- *exécution*, où nous exigeons que les conditions soient satisfaites.

Définition 0.0.32 Une séquence de longueur l de configurations commençant par la configuration initiale $\langle \mathbb{S}, C, \mathbb{P}, \varnothing, \varnothing \rangle$ et obtenue en appliquant des transitions de la Figure 6 est appelée *exécution symbolique* SE de longueur l.

Définition 0.0.33 Une *execution* E est une paire $\langle SE, \sigma \rangle$ d'une exécution symbolique SE et la substitution close σ, telle que pour la dernière configuration $\left\langle \mathbb{S}^l, C^l, \mathbb{P}^l, \mathcal{S}^l, \mathcal{H}^l \right\rangle$ de SE, $\mathcal{S}^l$ et $\mathcal{H}^l$ sont à la fois satisfaits par σ.

Alors le problème de l'orchestration distribuée est formalisée comme suit. Étant donné une entrée du problème, existe-t-il une exécution $E = \langle SE, \sigma \rangle$ de longueur $l \leq m$ de telle sorte que la séquence d'actions du client est vide à la fin, c'est-à-dire, la dernière configuration de SE est en forme suivante: $\left\langle \mathbb{S}^l, \langle 0, \varnothing \rangle, \mathbb{P}^l, \mathcal{S}^l, \mathcal{H}^l \right\rangle$?

Nous pouvons voir que l'exécution E définit un flux de messages, et donc, en particulier, nous pouvons extraire une séquence d'actions à réaliser par chaque partenaire.

Solution

Nous réduisons le problème de l'orchestration distribuée à la satisfiabilité d'un système de contraintes de déductibilité sous la condition de non-divulgation des données sensibles et ensuite décrivons une procédure de décision en vertu de l'hypothèse de nombre borné d'interactions.

Comme on peut construire un nombre fini d'exécutions symboliques différentes pour une entrée fixe d'un problème, on peut deviner une exécution symbolique avec sa configuration finale $\left\langle \mathbb{S}^l, C^l, \mathbb{P}^l, \mathcal{S}^l, \mathcal{H}^l \right\rangle$ où le Client n'a plus aucune action à effectuer (i.e. $C^l = \langle 0, \varnothing \rangle$). Ensuite, la construction de l'exécution souhaitée est équivalente à trouver une substitution close σ qui satisfasse à la fois $\mathcal{S}^l$ et $\mathcal{H}^l$.

Nous nous référons à la technique présentée auparavant pour résoudre des systèmes de contraintes dans le système de déduction de Dolev-Yao. Sous l'hypothèse non restrictive que dans l'entrée du problème il existe au moins une valeur atomique qui n'est sensible pour aucun partenaire, nous pouvons facilement adapter la technique mentionnée de telle manière que le théorème suivant soit valide:

Théorème 3. *La satisfiabilité d'un système de contraintes dans un système de déduction de Dolev-Yao sous condition de non-divulgation est dans NP.*

Idée de la preuve. Il suffit de préciser la valeur de α dans la Définition 0.0.25. Nous devons utiliser un atome (dont nous avons supposé l'existence) qui n'est sensible pour aucun des partenaires. Par conséquent, s'il existe une solution σ qui satisfasse le système de contraintes $\mathcal{S}$ et la condition de non-divulgation $\mathcal{H}$, alors $\pi(\mathrm{H}(\sigma))$ satisfera (i) $\mathcal{S}$ et (ii) $\mathcal{H}$, puisque par la définition de $\pi(\mathrm{H}(\sigma))$ on a $(\mathrm{Sub}(\pi(H(x\sigma))) \setminus \mathrm{Sub}(x\sigma)) \cap \mathcal{A} \subseteq \{\alpha\}$. □

Et ainsi,

Corollaire 5. *Le problème de l'orchestration distribuée est décidable et NP-complet.*

Analyse de protocoles cryptographiques

Attaques par réécriture XML

XML [W3C08] est un format largement utilisé qui est devenu la base de plusieurs langages. L'un d'entre eux est un format de message utilisé dans le protocole SOAP [Con07], un protocole de communication utilisé pour services Web. Comme les standards de sécurité pour services Web comme WS-SecurityPolicy sont appliqués aux messages XML, il est raisonnable de tenir compte des propriétés de la représentation XML en combinaison avec des éléments de sécurité comme le chiffrement et la signature.

Alors que les protocoles qui utilisent XML peuvent être vulnérables aux mêmes attaques que les protocoles classiques, un nouveau type d'attaque appelé *attaques par réécriture XML* (XML-rewriting attacks) [BFGP04, BFGO05] peut également être applicable. Comme remarqué dans [CLR07], le format XML nécessite de considérer plutôt un ensemble de termes pour modéliser le contenu du message qu'un seul terme. Un autre point mentionné dans le même ouvrage est que le choix de parseur de messages XML peut interférer puisque chaque partenaire peut utiliser sa propre implémentation qui peut sélectionner des nœuds différents comme réponse à la même requête. Ainsi, il est préférable d'envisager un comportement non-déterministe pour le parsing de documents XML.

Exemple

Considérons une e-boutique qui accepte les e-chèques, et nous supposons qu'elle est représentée par un service Web utilisant le protocole SOAP pour échanger des messages.

Il se compose de deux services:

- le premier expose la liste des biens à vendre avec leurs prix et traite les commandes à la réception des paiements,
- le deuxième est un service de livraison, il reçoit des informations sur les commandes déjà payées, et envoie les articles commandés à l'acheteur.

Un scénario simple pour commander un article est montré sur la Figure 1. Tout d'abord, un client envoie une commande en utilisant l'interface de l'e-boutique qui se compose d'un identificateur d'article, l'e-chèque, l'adresse de livraison et quelques commentaires. Puis, le premier service de l'e-boutique vérifie si le prix de l'article commandé correspond à la valeur du chèque reçu. Si c'est le cas, le service consomme le chèque et renvoie l'ordre au service de stock/livraison (sans l'e-chèque utilisé). Le service de stock et livraison prépare un colis avec l'article commandé et l'envoie à l'adresse donnée. Un commentaire est automatiquement

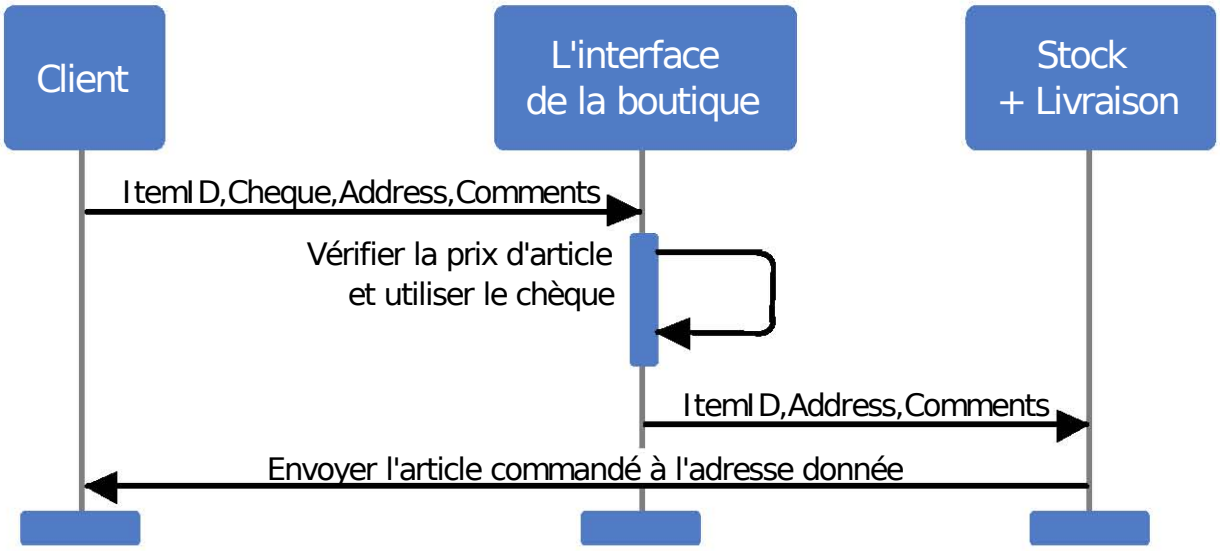


Figure 1: Scénario de commande d'un article

imprimé sur le colis pour donner quelques informations complémentaires sur la livraison (comme le code digital pour entrer le bâtiment, etc.)

Supposons que Alice ait un e-chèque de 5€. Elle a choisi un stylo simple (ItemID *simple*) pour acheter, mais elle a beaucoup aimé un autre doré qui est plus cher (ItemID *gilded*). Pouvons-nous aider Alice à obtenir ce qu'elle veut pour ce qu'elle a?

On peut formaliser le comportement des services comme suit (on écrit $(t_1 \cdot \ldots \cdot t_n)$ au lieu de $\cdot(\{t_1, \ldots, t_n\})$, tous les identificateurs qui commencent par majuscule sont des variables, ceux qui commencent par minuscule sont des valeurs atomiques sauf les symboles fonctionnels).

Pour l'interface de l'e-boutique:

$$?_{Client}(simple \cdot cheque5 \cdot IAddr \cdot IComm);$$
$$!_{Delivery}(simple \cdot IAddr \cdot IComm).$$

Pour le service de Stock et Livraison:

$$?_{Interface}(DItemID \cdot DAddr \cdot DComm);$$
$$!_{Client}\,\mathrm{sig}\,((DItemID \cdot DAddr \cdot DComm), \mathrm{priv}\,(k_s))\,,$$

où le message $\mathrm{sig}\,((DItemID \cdot DAddr \cdot DComm), \mathrm{priv}\,(k_s))$ modélise l'envoi de colis contenant un article avec identificateur $DItemID$.

Alice connaît :

$simple, gilded$:	des identificateurs d'articles;
$cheque5$:	un e-chèque de 5€;
$addr$:	son adresse;
$cmnts$:	le code digital de sa résidence;
k_s:	la clef publique de la boutique.

Pour répondre à la question posée, nous pouvons construire un système de contraintes "mixtes" (contraintes de déductibilité et des équations modulo ACI) (voir la Figure 2).

$$\left\{\begin{array}{ll} \{gilded, simple, cheque5, addr, cmnts, k_s\} \ \triangleright & \\ \qquad\qquad (simple \cdot cheque5 \cdot IAddr \cdot IComm) & (1) \\ (simple \cdot IAddr \cdot IComm) =_{ACI} \quad (DItemID \cdot DAddr \cdot DComm) & (2) \\ \{gilded, simple, cheque5, addr, cmnts, k_s, & \\ \text{sig}\,((DItemID \cdot DAddr \cdot DComm), \text{priv}\,(k_s))\} \ \triangleright & \\ \qquad\qquad \text{sig}\,((gilded \cdot addr \cdot DComm), \text{priv}\,(k_s)) & (3) \end{array}\right.$$

Figure 2: Un système de contraintes mixtes pour décrire une vulnérabilité possible dans le scénario de e-boutique

Contrainte (1) montre que Alice peut construire un message attendu par la boutique de la part d'un client. Egalité (2) représente une requête du premier au deuxième service de la boutique: la partie gauche est un message envoyé par le service d'interface, et la droite est un message attendu par le service de stock/livraison. La dernière contrainte montre qu'à partir des valeurs reçues par Alice, elle peut construire un message qui modélise la livraison de l'élément avec ItemID *gilded*.

Notez que l'égalité (2) peut être remplacée par une contrainte de déductibilité $\text{enc}\,((simple \cdot IAddr \cdot IComm), k) \triangleright \text{enc}\,((DItemID \cdot DAddr \cdot DComm), k)$. En le faisant nous pouvons trouver une solution de ce système avec la technique présentée auparavant. Par exemple, en voici une solution σ:

$IAddr$	$\mapsto addr$	$IComm$	$\mapsto (gilded \cdot cmnts)$
$DItemID$	$\mapsto gilded$	$DAddr$	$\mapsto addr$
		$DComm$	$\mapsto (simple \cdot cmnts)$

En suivant cette solution, nous voyons que Alice peut envoyer des commentaires inattendus (qui présente deux noeuds XML), et le parseur du service de livraison peut choisir une entrée avec ID *gilded*, tandis que le parseur de la première couche (interface) peut retourner une valeur de la première occurrence de ItemID, c'est-à-dire, *simple*. Une attaque-requête peut ressembler à ceci:

```
<ItemID>simple</ItemID>
<Cheque>cheque5</Cheque>
<Address>addr</Address>
<Comments>cmnts</Comments>
<ItemID>gilded</ItemID>
```

L'attaque avec la requête d'Alice est volontairement mal-formée, contenant plusieurs occurrences de lstinline!ItemID!, et s'appuie sur des différence de parsing entre les services.

Modèle de plusieurs intrus non-communicants

Bien que le modèle d'intrus actif Dolev-Yao [DY83] soit considéré le plus souvent, nous contribuons en outre avec un autre modèle d'intrus (moins absolu que celui de Dolev-Yao) où plusieurs intrus non communicants sont considérés par opposition à un seul intrus contrôlant l'ensemble du réseau.

Il est utile de considérer cet intrus "moins fort" dans des scénarios où certaines restrictions peuvent être fournies par des voies physiques, techniques ou administratives.

Dans notre modèle, on considère plusieurs intrus "locaux" non communicants, chacun ayant les capacités de l'intrus Dolev-Yao actif mais ne contrôlant que certaines voies de communication. Par ailleurs, ces intrus ne peuvent pas communiquer durant l'exécution du protocole, mais seulement quand ils quittent le protocole. Par exemple, on peut considérer le cas où un espion a réussi à installer plusieurs petits appareils malveillants sur certaines parties du réseau déployé (câbles réseau, par exemple) dans certaines organisations de telle sorte que chaque appareil contrôle ses propres voies de communication, mais n'a aucun moyen de communiquer directement avec ses semblables à cause de la configuration du réseau (configuration de routeurs) ou de conditions physiques (par exemple, murs massifs). Après un certain temps l'espion peut revenir et ramasser ses appareils. Puis il est capable de lire et de joindre les informations recueillies pour pouvoir déduire quelques secrets (à noter que les appareils ne sont pas seulement de passives oreilles indiscrètes, mais peuvent affecter l'exécution du protocole en y prenant part activement).

Nous ne présentons pas dans cette version abrégé les détails du modèle, car le formalisme ressemble beaucoup au celui utilisé dans le modèle de l'orchestration distribuée. Nous voudrions pointer cependant que ce problème, sous la condition des participants de protocole fixés, peut être réduit à une résolution d'un système général de contraintes de déductibilité, et est donc décidable et même NP-complet.

Conclusions

Résumé

Dans cette thèse, nous avons étudié une technique symbolique pour traiter des primitives cryptographiques dans le contexte des systèmes de communication. Plus précisément, nous avons considéré la satisfiabilité de ce que l'on appelle *systèmes de contraintes de déductibilité*.

Cette technique a été profondément étudiée pour les 10 dernières années dans le contexte de l'analyse de protocoles cryptographiques avec un nombre borné de sessions. Dans ce cadre, une hypothèse naturelle non restrictive est généralement considérée: le système de contraintes doit être dans une certaine forme, plus précisément, un système de contraintes doit être *bien formé*. Cette propriété est respectée à chaque fois que l'on construit un système de contraintes pour vérifier l'insécurité d'une session donnée de protocole en présence de l'intrus actif de Dolev-Yao.

Beaucoup de travaux ont été réalisés sur l'assouplissement de l'hypothèse de cryptographie parfaite en tenant compte des propriétés algébriques des opérations utilisées pour construire des messages (par exemple, XOR).

Mais ces techniques ne fonctionnent que lorsque l'on considère un modèle classique d'intrus Dolev-Yao. Si nous passons à un autre modèle où des intrus multiples non communicants attaquent une instance d'un protocole, alors le système de contraintes de déductibilité qui est construit pour modéliser ce cas n'est plus bien formé.

Ceci nous a motivé à considérer les contraintes générales, où la propriété d'être bien formé est étendue. Nous avons défini de tels systèmes de contraintes et avons montré aue le problème de la satisfiabilité de tels systèmes est décidable et même NP-complet.

Par ailleurs, nous avons également considéré un symbole ACI (i.e., associatif, commutatif et idempotent) en tant que constructeur d'ensembles. Ces propriétés sont fort utiles pour modéliser un ensemble de nœuds XML. Cela peut être extrêmement utile pour les protocoles dont les messages sont basés sur le langage XML. Par exemple, la majorité des services Web utilisent le protocole SOAP qui est basé sur XML. Ainsi, cette extension peut être employée pour raisonner sur la sécurité de services Web, où chaque service Web est considéré comme un agent jouant le rôle d'un protocole.

Nous avons présenté un exemple d'utilisation de notre technique pour modéliser une attaque basée sur la représentation XML des messages et sa détection. Ce n'est pas une nouveauté [CLR07], mais nous proposons une autre manière de détecter de ce genre d'attaques.

Comme nous l'avons mentionné auparavant, les systèmes généraux de contraintes peuvent être utilisés pour raisonner sur la sécurité de sessions de protocole en présence de plusieurs intrus actif de type de DY, où chacun contrôle seulement quelques voies de communication, mais pas le réseau en totalité. Par ailleurs, ils ne peuvent pas communiquer pendant l'exécution du protocole, mais seulement quand ils le quittent. Dans ce cadre nous avons montré que le problème du secret est décidable pour une session de protocole donnée.

L'autre domaine où nous avons appliqué notre technique de satisfaction de systèmes généraux

de contraintes de déductibilité est la composition de services Web préservant des politiques de sécurité. Nous avons considéré un modèle d'orchestration distribuée qui couvre à la fois deux approches de la composition de services Web: la chorégraphie et l'orchestration.

Ce modèle admet une technique de composition automatique que nous avons également décrite. À notre connaissance, il n'y a pas beaucoup de travaux sur la composition automatique des services Web qui prennent en compte des politiques de sécurité des services disponibles.

Par ailleurs, notre modèle restreint la communication entre les partenaires sur lesquels l'orchestration est répartie de manière à ce qu'aucune donnée sensible d'un partenaire ne peut fuir directement vers un autre.

La composition automatique dans ce cadre peut également être réduite à la satisfiabilité des systèmes de contraintes de déductibilité mais sous une *condition de non-divulgation*. Cette condition que nous avons présentée dans la thèse n'affecte pas la décidabilité. Ainsi, nous avons montré que le problème de l'orchestration distribuée automatique des services Web sous la condition de non-divulgation et prenant en compte des politiques de sécurité des services Web est décidable.

Un cas particulier du modèle présenté est une orchestration de services Web sécurisés. Si nous appliquons notre approche dans ce cas, il n'est pas nécessaire d'avoir une technique de résolution de systèmes généraux de contraintes de déductibilité. Décider de la satisfiabilité des systèmes bien formés suffirait. Ainsi, nous sommes en mesure de réutiliser les outils existants déjà capables de traiter ce problème. Un de tel outil, nous pouvons le retrouver dans le domaine de l'analyse de protocoles cryptographiques.

S'appuyant sur cet outil, nous pouvons construire un logiciel qui peut résoudre le problème de l'orchestration automatique de services Web prenant en compte les politiques de sécurité. Nous avons mentionné une telle mise en œuvre appelée AVANTSSAR Orchestrator au sein du projet Européen AVANTSSAR.

Perspectives

Il y a plusieurs perspectives concernant la satisfiabilité des contraintes générales. La technique présentée est loin d'être parfaite et universelle. Tout d'abord, on peut voir que les clés publiques prises en compte dans la signature ne peuvent avoir que des valeurs atomiques. Cette limitation à des clés publiques atomiques permet de couvrir la plupart des cas utiles, mais cette restriction désagréables peut probablement être retirée.

Deuxièmement, prendre en compte d'autres symboles avec différentes propriétés algébriques, Par exemple, même XOR ou l'exponentiation modulaire, serait un "plus", puisque les applications de telles techniques étendues peuvent mieux se rapprocher des systèmes réels.

Ensuite, l'extension des systèmes généraux de contraintes de déductibilité avec la négation (permettant des contraintes négatives de type "terme t ne doit pas être déductible à partir d'un ensemble de termes E") nous donnerait, par exemple, certaines procédures plus jolies de composition de services Web sous des contraintes différentes. Par ailleurs, en utilisant ce système de contraintes mixtes de ce genre, nous pourrions exprimer la propriété d'équité (fairness) pour les protocoles.

Une extension liée à la précédente est de considérer des inégalités de termes comme des contraintes supplémentaires. Ceci permettrait, par exemple, d'exprimer et de vérifier une violation de la propriété d'authentification dans le modèle des intrus multiples non communicants. Pour l'instant nous ne traitons que le secret dans ce cadre.

Enfin, il faudrait idéalement une mise en œuvre effective de la procédure de décision proposée.

Sans elle, l'approche présentée ne peut avoir qu'une valeur théorique. Pour cela, des méthodes heuristiques pour satisfaire des systèmes généraux de contraintes de déductibilité, qui est NP-complet, est probablement une bonne direction à explorer.

Concernant les applications de la procédure de décision présentée dans cette thèse, il est difficile de définir d'autres extensions qui soient complètement indépendant de la technique sous-jacente discutée ci-dessus. Cependant, nous pouvons remarquer la nécessité d'une étude approfondie des standards industriels sur services Web et ses correspondances avec les éléments de notre modèle.

Comme objectif à plus long terme nous pouvons citer une considération de comportement étendu de services Web admettant, par exemple, les boucles. La composition dans ce cas aurait éventuellement besoin d'un mélange de techniques basées sur des automates qui, en général, permettent de raisonner sur les comportements avec des boucles, mais habituellement ne tiennent pas compte de la structure des messages, (en particulier, les primitives cryptographiques), et l'approche présentée, présentement mal adapté aux boucles, mais qui offre un raisonnement poussé sur la structure du message en présence de primitives de cryptographie. Néanmoins, la complétude de telles approches de combinaison est loin d'être garantie.

List of Figures

1 Attaque sur protocole SIP xi

1 Représentations arborescente et en forme DAG de termes xx
2 Illustration pour l'Exemple 2: Éléments de t (colorés) xxiv
3 Illustration pour l'Exemple 3: Forme normale d'un terme xxiv
4 Illustration pour l'Exemple 4: Quasi-soustermes de t (colorés) xxv
5 Illustration pour l'Exemple 7: Travail de la fonction $H^{\mathcal{S},\sigma}(\cdot)$ xxvii

1 Représentation DAG du système de contraintes $\mathcal{S}$ xxxiv

1 Un système de contraintes décrivant une orchestration distribuée possible. xli
2 Solution pour l'instance d'un problème de l'orchestration distribuée xli
3 Exemple d'entrée générique pour le problème de l'orchestration distribuée xliii
4 Un cas particulier de l'entrée: le problème d'orchestration xliv
5 Un cas particulier de l'entrée: le problème de chorégraphie xliv
6 Système de transitions xlv

1 Scénario de commande d'un article xlviii
2 Un système de contraintes mixtes pour décrire une vulnérabilité possible dans le scénario de e-boutique xlix

1 Personal information of two users extruded from a non-secure Web site 2
2 Attack on SIP protocol 5

1.1 Tree and DAG representations of terms 24

2.1 Illustration for Example 14: Elements of t (highlighted) 29
2.2 Illustration for Example 15: Normal form of a term 30
2.3 Illustration for Example 16: Quasi-subterms of t (highlighted) 31
2.4 Illustration for Example 19: Working of $H^{\mathcal{S},\sigma}(\cdot)$ function 33
2.5 Proof plan 51

3.1 DAG-representation of constraint system $\mathcal{S}$ 56

4.1 Illustration for the orchestration example 76
4.2 A constraint system describing a possible orchestration. 77
4.3 Solution for the orchestration problem example 77
4.4 Illustration for the distributed orchestration problem example 79
4.5 A constraint system describing a possible distributed orchestration. 80

4.6 Solution for the distributed orchestration problem example 81
4.7 Generic input example for distributed orchestration problem 84
4.8 A special case of input: orchestration problem 84
4.9 A special case of input: choreography problem 85
4.10 Transition system 86
4.11 The AVANTSSAR Validation platform 91
4.12 Services as protocol roles 97
4.13 Solution to the orchestration problem 98
4.14 Goal and its putative Client 98
4.15 Tool architecture 99
4.16 Data flow inside the Orchestrator tool. 104
4.17 First and following invocations of the Orchestrator 104
4.18 Resume to generating the next orchestration 105
4.19 Toy example: available services and the Client 106
4.20 A Mediator service for DCS orchestration problem 110

5.1 Two sessions of NSPK protocol 121
5.2 A constraint system expressing a possible attack on NSPK protocol. 122
5.3 An attack to NSPK protocol 123
5.4 Untrusted routers 124
5.5 Generic input example for the coordinated attack problem 128
5.6 Ordering item scenario 131
5.7 A constraint system describing possible vulnerability in E-shop scenario 132

Reminder lists

List of some typographical conventions

We will use sometimes this box to shortly sum up some points described in the surrounding text. It is not completely equivalent to what is written, but could help to recall the content of the context and to ease the looking up over the thesis.

Summing up

This box contains short information indicating the content of the surrounding text

If we need to show a piece of a document (like XML document, or specification code), then we will display it on some light background. The style may look like the following:

```
<comment> This is an extract of some XML document </comment>
<option> It can be also some ASLan code </option>
```

List of abbreviations and synonyms

DY	Dolev-Yao
ACI	Associative Commutative Idempotent
LHS	Left-hand side
RHS	Right-hand side
DAG	Directed acyclic graph

deducibility constraint	symbolic constraint	
constraint system	system of constraints	constraints
non-disclosure condition	non-disclosure property	
security policy	policy	

List of symbols

$\mathbb{N}$	Set of integers
$\mathbb{N}^+$	Set of non-negative integers
$x \mapsto y$	"Maps x to y"
$\lvert X \rvert$	Cardinality of X
$\mathrm{dom}\,(f)$	Domain of f
$\langle a, b, c \rangle$	Tuple of three elements
$\mathcal{T}$	Set of all terms
$\mathcal{T}_g$	Set of all ground terms
$\mathcal{X}$	Set of all variables
$\mathrm{root}\,(t)$	Root symbol of term t
$\mathrm{Vars}\,(t)$	Set of variables used in t
$\ulcorner t \urcorner$	Normalized form of t
$\mathrm{Sub}\,(t)$	Set of subterms of t
$\mathrm{size}\,(t)$	Number of nodes in DAG-representation of t, $\lvert\mathrm{Sub}\,(t)\rvert$
$t\sigma$	Substitution σ applied to t
$\mathrm{Der}\,(E)$	Set of all terms derivable from set of terms E
$E \triangleright t$	Deducibility constraint: term t should be derivable from set of terms E
$\mathcal{S}$	Constraint system $\mathcal{S}$, a set of deducibility constraints
bin	Shortcut for any binary functional symbol
$\mathrm{QSub}\,(t)$	Set of quasi-subterms of term t
$\mathrm{QS\mathring{u}b}\,(t)$	Set of non-variable quasi-subterms of term t, $\mathrm{QSub}\,(t) \setminus \mathcal{X}$
$\mathrm{elems}\,(t)$	Set of "elements" of term t
$\pi(E)$	Normalized ACI-set of terms from set of terms E
$\mathrm{enc}\,(m, k)$	Encryption of m with k by symmetric scheme
$\mathrm{aenc}\,(m, k)$	Encryption of m with k by asymmetric scheme
$\mathrm{pair}\,(m_1, m_2)$	Pairing (or concatenation) of m_1 and m_2
$\mathrm{priv}\,(k)$	Private key $\mathrm{priv}\,(k)$ that corresponds to public key k
$\mathrm{sig}\,(m, \mathrm{priv}\,(k))$	Digital signature of m with private key $\mathrm{priv}\,(k)$
$\mathrm{apply}\,(f, m)$	(Hash)function f applied to m
$\cdot(\{t_1, \ldots, t_n\})$	ACI-set constructed from $t_1, \ldots, t_n$
$?_f m$	Receive m from f
$?m$	Receive m
$!_t m$	Send m to t
$!m$	Send m
$f \rightharpoonup t$	Channel from f to t

Preface

1 Motivation

The communication systems, be it online shopping, mobile phones infrastructure or a Video-on-demand television, are used ubiquitously and it is hard to overestimate their important role in our life. It is usually required to protect such systems from illegal use and external attacks, and to achieve this goal cryptography comes to the rescue. Messages transmitted by electronic devices, or parts thereof may be encrypted (with the purpose to prevent the unauthorized reading), signed (in order to fix the message origin and thus prevent the impersonation) or hashed (to prevent the message disclosure while keeping some information about it).

For example, two persons having a voice conversation over the Internet, e.g., using Skype or any other Voice-over-IP software, want to be sure that no one else but them could obtain a record of their conversation, even one who has a physical access to the wire over which the data packets containing the speech were being transmitted. In this case, the encryption could seemingly help to ensure this property, since if a message is encrypted, the original data could be obtained only by those who knows a secret key (e.g., password).

In the online shopping systems, if we abstract away the cryptography, then transmitting only a payload message like a credit card number, its security code and card holder name is enough to complete a payment. But such a transmission is vulnerable against a simple overhearing, where the data that should stay secret can be learned by some intruder and then reused to perform unauthorized payments.

A case I have personally met, where the cryptography was not used but should have been is in Web site of an organization that manages a database of PhD students (and later doctors) of multiple doctoral schools in France. For the students that want to start or continue a PhD program in these schools, it is necessary to enter different information including personal data, and, optionally, a CV that can be shown to employers. According to the information from the official site, there are currently more than 30'000 registered PhD students and doctors of different specialties. A registered user is identified by its name and password.

The first problem is that login form does not use secure connection and all data, including password is sent in plain text using HTTP POST method. Thus, anyone who is able to overhear the traffic can capture login and password of the user trying to connect. Likely, this sensitive data must be transmitted in encrypted form.

There is another vulnerability that allows to anyone to read and even change information of any registered user. After logging in, the site creates a cookie (a piece of information stored on the side of the client, here, by the browser) that contains apparently a database record number of the user. Then, if the user requests to load another page form this site (e.g., jumps from personal information page to thesis information page) in the Web interface, the value of the cookie is sent together with the new page address to the Web server. And, depending on these values (record number and requested page), the server returns a new Web page. As you can see, one may make Web server return any page proposed by the Web interface of any registered user by indicating as a value of the cookie its record number, i.e. a number from 1 to, approximately, 30000. Thus, one does not need to know a user's password for doing this. Two instances of some information that can be obtained in this way is given in Figure 1.

Moreover, one can modify information on these pages and submit the changes that will be stored in the database. Of course it may be misused: from changing CVs of others in order to get higher chances to be employed or collect personal email addresses for spamming, to using birth date and place to recover passwords of other services used by the students.

By the moment of writing these lines, the issues was not corrected. Maybe it is not a big problem, since no one is trying to get this information. But on more grave levels, like business

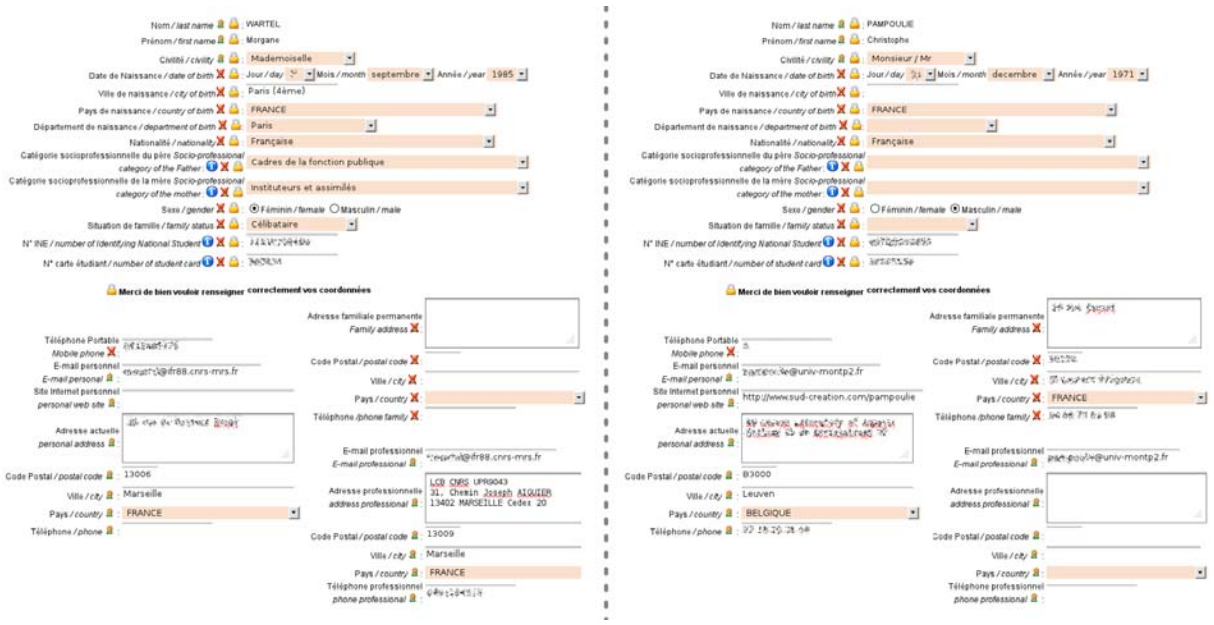


Figure 1: Personal information of two users extruded from a non-secure Web site

or government, this kind of security fails may be unforgivable. This means that the security of systems, notably communicating ones, must be verified before deployment as far as possible.

Cryptographic protocols The combination of cryptographic primitives (like encryption and digital signature), (pseudo-) random numbers and payload messages (like voice data in one of the previous examples) used by the communication participants in some fixed order forms a *cryptographic* or *security protocol*. Note that if we consider each participant separately from others, we can say that the usage of cryptography forms a *security policy* imposed on its communication.

In order to give an intuition about security protocols, let us imagine a spy story, where one secret agent (spy) must transfer some collected information to another agent (messenger) he does not know. The spy knows only a crowded public place where they have to meet, like a pub, and some hints about the appearance of the messenger, like "he will wear a yellow cap". Since this information is possibly not enough to identify unambiguously the messenger, the spy has to ensure that the person he is going to hand over the valuable information is the right one. To this end, both spy and messenger have a pre-shared (by some other means irrelevant here) question and a "right" answer for it, such that any sane person who does not know the the "right" answer cannot produce it. For example, the "right" answer to question "Could you please tell me the Zulu time?" could be "Sorry, I don't smoke when the clouds are broken." In this way the spy *authenticate* the messenger, that is, if the spy gets the "right" answer to his question, he apparently might be sure the person in a yellow cap he is talking with is the messenger. After that, the spy can hand him a small piece of paper with the information to transfer. Moreover, the information written on the paper is not in plain text, but also ciphered, since if he were kept by the enemy agents, or the paper fall into their hands, they would not be able to read the information.

This procedure can be seen as a security protocol described as follows:

$$\begin{aligned} &\text{Spy} \to \text{Messenger}: && \text{Question} \\ &\text{Messenger} \to \text{Spy}: && \text{Answer} \\ &\text{Spy} \to \text{Messenger}: && \text{enc}\,(\text{Message}, \text{Key})\,, \end{aligned}$$

where $\text{enc}\,(\text{Message}, \text{Key})$ denotes the Message cyphered with Key.

In this protocol we have two *roles*: Spy and Messenger. Normally, different agents may play these roles. For example, Sidney Reilly can play role Spy, as well as Mata Hari or Rose Greenhow. The same for the messenger, it can be Lily Mackall, some person X, or even Sidney Reilly himself. Different pairs of Spy-Messenger may have their own secret questions and answers as well as their own secret messages and keys.

Once one assigns the players to protocol roles and instantiates all other protocol variables (like Message, Key, etc.) we obtain an instance of a protocol — a *protocol session*, i.e. a set of instances of protocol roles.

We want to note that we will work with given fixed set of protocol sessions or instances of protocol roles. In this setting, we will consider the insecurity problem of cryptographic protocol sessions. There are multiple tools and techniques that deals with the cryptographic protocol verification problem with the fixed number of sessions (e.g. [BMV05b, ACC07, Tur06, BHK05]). But it is worth to mention that there exist another research branch that deals with so-called *unbounded verification*, where the number of protocol sessions is not limited (e.g. [Cre08, BLP03, Bla01, EMM09]).

The cryptographic protocols are widely deployed in wireless communications, payment systems, electronic messaging and many other domains. Unfortunately, the simple fact of using cryptographic primitives does not guarantee the desired security properties: for example, the fact that a message is encrypted does not guarantee, in general, that it cannot be read by some attacker, or, if someone is able to answer an authentication challenge that normally requires knowledge of some secret, then it does not imply he really knows this secret.

For an instance of the latter example, we can refer to SIP, a Session Initiation Protocol[8] [RSC+02]. It is a widely used signaling protocol for VoIP and it allows to manage the media sessions, like audio/video calls (a lot of people, including me, use VoIP services that are based on SIP to make cheap calls abroad). If we abstract away the details, then in order to make a call, a registered user sends `invite` request to a SIP proxy server indicating the SIP address of the person he wants to call and his own (e.g., alice `invite`s bob). In order to verify whether the user is not an impostor, SIP support a mechanism for authentication based on shared secret (i.e. password that user had entered during the registration). Proxy may perform an authentication challenge by sending some random piece of data (nonce) and the user must return a hash value (see § 2.1, p. 8) of his request, received nonce and his password[9] concatenated together. In this way, the proxy may also calculate this value, since it knows the password and the user's request, and compare to the received one. If the value calculated by the proxy coincides with one received from the user, then the proxy considers that the user knows his password (so as in order to calculate this value one must know the key). It looks like only the user who knows the key may make a call from his account, for which he will possibly be charged. But it is not the case!

[8] While SIP protocol does not exclusively deal with security, but is mainly in charge of other tasks, it uses cryptography in certain points, e.g., where it is needed to authenticate the caller.

[9] In the reality, the message to be hashed is a little bit different, but the key idea is the same.

The pitfall is that the same `invite` request[10] is used, in addition to other cases, to put an established conversation on hold. For instance, if B is having a conversation with A and B wants to put on hold this call, he has to send `invite` request indicating SIP address of A and his own. At this moment, a proxy can send an authentication challenge in order to check the identity of the requester, and in order to complete the request, B must answer it. Note that the authentication scheme used here is the same as discussed before. This is the point where one may get a response on authentication request sent by the proxy, if he wants impersonate B.

The question how to make B put a conversation (with some A) on hold can be easily solved in practice. Normally, it is enough to call B during the established conversation, such that B have another call on the line. Most likely, B will ask A to wait and put him on hold in order to answer the new incoming call.

Now we can describe the attack where an attacker can impersonate a registered user Bob and make a call to arbitrary user (or phone number) Alice, using a proxy server of the victim. Thus, if such call is paid, the money will be charged from Bob's account for the conversation between the attacker and Alice.

The synopsis is as follows (see Figure 2): an attacker will issue a call directly to the victim (Bob) in the name of Alice. The victim answers the incoming call and later on receives another call (possibly provoked by the attacker). In order to answer it, Bob puts the attacker on hold. Once the attacker receives the `invite` request specifying the "On hold", he will immediately make a call to Alice using a proxy of Bob personating Bob itself. Proxy will try to authenticate him by sending the corresponding request. This request will be forwarded to the victim to authenticate (on behalf of the proxy) Bob since he wanted to put on hold the conversation. Bob answers it and the intruder uses the answer to forward it to proxy and, in this way, reply to the proxy's challenge. Since the authentication challenge is successfully answered, the proxy will authorize this call.

Seemingly, SIP is IETF approved standard, it is widely used, it employs cryptography, but still it is vulnerable to different attacks, like presented one [AARS08, AARS09] or various denial of service attacks [GS07].

More sophisticated attacks may rely on properties of cryptographic algorithms and operations used in it [RS98, PQ01], or on the message representation schemes [BFGP04, CLR07]. In this work we will also present a way to model sets as an additional building element that can be used in messages. Assuming also the cryptography, sets can be particularly useful when considering *secured communication of Web Services,* since they normally employ XML-based messaging.

Security-aware Web Services is another domain that will be considered in this thesis. Web Services are designed to be interoperable and composable, that is, may cooperate and be combined into more complex services. To this end, they expose some interface (usually described in WSDL language) that should be used if one wants to invoke operations proposed by the services.

For example, assume two Web Services: one that returns a list of tablet PCs available on the market which satisfy given (minimal) technical specifications, and another that by a name of an application returns in some uniform way its recommended hardware requirements. In this setting one can imagine their composition: a new service that accepts an application name and returns list of available on the market tablet PCs on which this application can run smoothly. Note that neither first nor second services proposes the same functionality.

[10] It is referred as "re-invite" in SIP specification, but the message uses exactly the same keyword as "invite"

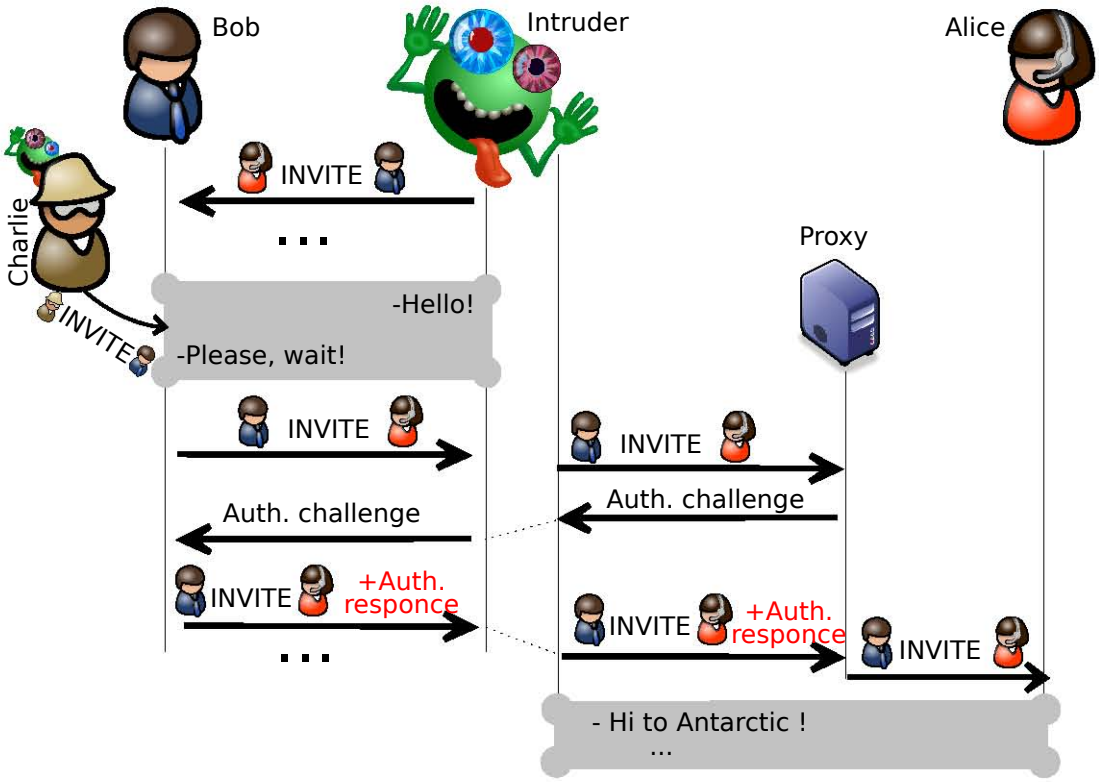


Figure 2: Attack on SIP protocol

In order to guarantee some security properties, like in case of protocols, the *security policies* are imposed on Web Services. They extend the Web Service's interface by stating additional requirements on the input and output of every operation proposed by the service. For example, an operation of banking service that receives an account number and password, and returns its current status may accept only cyphered message in its input (in order to protect the password) and both signed and encrypted output (signed, in order to certify the message origin and integrity, and cyphered, in order to protect the account status from unauthorized reading).

These restrictions, imposed by security policies, make automation of one of the underlying principles more complex. Here we talk about the problem of automatic Web Services composition, that is, make the Web Services work together in order to obtain some desired behavior.

There are two basic approaches on Web Services composition: *choreography* and *orchestration* [Pel03b]. The main difference is a way the Web Services intercommunicate. In the case of choreography, Web Services communicate directly with each other following some global scenario, while in the case of orchestration, all communication are passed through a central entity called *Mediator* or *Orchestrator*, which distribute requests to services of the community and gather the responses in order to, possibly, reuse the obtained data for constructing the following requests needed to achieve the wanted behavior.

We will concentrate on a compromise which lays between these two approaches, a so-called *distributed*, or *decentralized orchestration* [PE09, BMM06]. Here, the Mediator is distributed on several nodes, we call *partners*, each of them implements its own part of the composition. In this setting an orchestration is a special case in which only one partner is involved, whereas choreography is another case in which there is one partner per available service.

There are a lot of works on automatic composition of Web Services. Most of them abstract

away the structure of the messages and work only with atomic messages from some usually finite set, or in the level of operations (e.g. [BFHS03, BCF08, Pat09, Pad08, CGL+08]). A COLOMBO model that allows messages from infinite domain and considers message structure [BCD+05] has an automatic composition procedure only for some restricted fragment. Moreover, cryptography is not modeled in any of the works cited above.

We can also mention a work like [BCD+09], where the cryptography is used to make some composition secure, but not considered it in the problem of automatic composition itself; and work like [MFP11], where the operations required to be performed on secured messages are already given in the problem input in the form of *security contract*.

In our approach we consider a rich message structure that allows to express security policies applied on input and output messages of Web Services operations. This makes an additional constraint when trying to invoke a service, since one has to satisfy corresponding security policy.

Moreover, the proposed approach allows an "easy" implementation for the case of orchestration which we will also report, since it admits reusing existing tools for verifying cryptographic protocols.

Summing up We consider in this documents cryptography in two contexts: the security of cryptographic protocols and composition of security-aware Web Services.

We may note that the automatic reasoning on the secured messages is of rising interest, since it can provide some formal guarantees on the security of given communication or show the vulnerability ((in)security of cryptographic protocols), and a procedure for mediation the security policies of the communicating parties (composition under security constraints).

Deducibility constraints As an underlying technique we use so-called symbolic *deducibility constraints* that express a possibility to derive a message from some knowledge. This technique was well studied in the context of cryptographic protocol analysis with bounded number of sessions (e.g. [MS01, CLC03, Shm04]). These works consider constraints only in special form, called *well-formed* constraints. For the classical bound-session protocol analysis (with a single powerful Dolev-Yao intruder) this assumption is not restrictive, since the system of constraints expressing the possibility of an attack on a protocol is always well-formed. On the other hand, if we want to consider another model where multiple intruders each with a local influence area attack a protocol and have no means to communicate during the attack, the well-formed constraints are not enough.

In this work we will relax this assumption and give a decision procedure for *general* constraint systems.

Note that the similar holds for the security-aware Web Services composition: for the case of orchestration, the well-formed constraints are enough. But for more general case, distributed orchestration, the assumption must be relaxed. We give more details in corresponding chapters.

In the rest of this introductory part we first shortly describe the modeling choices (what and how we model), and then shortly present the contributions and plan of the thesis.

2 Modeling

If we consider a question of verifying whether the specified communication is secure, there exists mainly two approaches: *computational* and *symbolic*. The former assumes that the messages are bit-strings and the intruder is an arbitrary probabilistic polynomial-time Turing machine. In general, this approach is closer to reality, but harder to deal with, especially automatically.

The latter abstracts away from the bit-string representation of messages and considers them as terms in some term algebra (possibly with an equational theory to reflect properties of some real-world operations). The intruder in this case is represented by a set of transformations he can apply on the messages. In this case the (in)security proofs are easier to automatize, but nevertheless, the problem of protocol insecurity is in general undecidable [EG83].

For the composition of Web Services taking into account their security policies, it is enough to consider only symbolic abstraction, since we don't want the mediator to break an encryption of some message simply in order to adapt the communication.

Thus, in this thesis we will concentrate on the symbolic approach.

2.1 Cryptographic primitives

Since we want to reason about secured messages, we have to consider cryptographic primitives. We list the ones we model by giving a short description of each.

Public-key encryption

Also called asymmetric encryption, this kind of cryptographic encryption schemes suppose the existence of two keys:

public key which is used as an input to asymmetric encryption algorithm in order to cypher a given message. These keys are supposed to be publicly known (what is reflected in the name).

private key another key that corresponds to the public one. It is used as input to asymmetric decryption algorithm together with encrypted message in order to decrypt it. Supposed to be known only by the owner.

A typical usage of this encryption scheme is to transmit a message in such a way that no one except the addressee could read it. Since the public key and its owner is supposed to be publicly known (using third trust party), anyone can use this key to encrypt a message and send it to the key's owner. Moreover, one should not worry if someone overhears such message, since the only person possessing the private key is the addressee and thus only the latter can decrypt it.

Theoretically, it is possible to infer a private key given a public one, but this relies on a computationally difficult problem like factoring large integer numbers or calculating discrete logarithms. In our model, like in many other symbolic models, we suppose that this calculation is unfeasible.

As an example we recall two well-known schemes based on difficulty of discrete logarithm problem are Diffie-Hellman [DH76] and ElGamal [ElG85], and one based on factoring large integers is RSA [RSA78].

The encryption of message m using public-key cryptography scheme with key k will be later referred as $\mathrm{aenc}(m, k)$ (where aenc stands for asymmetric encryption), and the corresponding private key as $\mathrm{priv}(k)$.

Digital signature

Another useful application of public key cryptography is a digital signature. It can be seen as an inverse use of public-key encryption, that is, an owner of a private key can generate such piece of data (called *signature*) for his message that anyone having a corresponding public key can verify

that this signature was produced exactly for the given message and by someone possessing the private key.

It pursues several goals:

- It *authenticate* a signer, since only the owner of the private key could produce the signature.
- From the other side, it provide a *non-repudiation* property of origin, that is the signer cannot deny having signed this message.
- It may serve to guarantee the message *integrity*, that is if the message was changed by some third party, the signature becomes invalid.

Usually, to reduce the size of the signature, the message is first hashed and then signed. Thus, having only the signature it is impossible to infer the message itself.

As an example of well known digital signature schemes we can recall RSA-based PKCS#1 [JK03] and ElGamal-based DSA [GFD09].

The digital signature with signature key priv (k) of message m will be referred as sig $(m, \text{priv}\,(k))$ (where sig stands for signature). A verification key corresponding to signature key priv (k) is k.

Symmetric encryption

In the contrast to public-key encryption, the symmetric encryption deals with one key both for encryption and decryption called *symmetric key*. Thus, the symmetric key must be shared, if two users want to secure their communication using symmetric schemes.

Normally, to share a symmetric key a public key cryptography is used: first, a symmetric key is transferred in encrypted form from one entity to another, and then this key is used to encrypt the further communication. This is quite beneficial combination, since symmetric cryptography algorithms are typically faster than public-key ones.

As an example of symmetric key schemes we can recall *AES* [DR02] and BlowFish [Sch94].

The symmetric encryption of message m with key k will be referred later as enc (m, k) (where enc stands for encryption).

Hashing

Hash value of a message, also called a *message digest*, is a piece of data usually of a fixed length generated using hash function from a given arbitrary message.

The hash function is supposed to have the following properties:

- It is easy to calculate its value on any message;
- It is infeasible to find a message that have the given hash value;
- It is infeasible to find a collision of hash function, i.e. two messages that have the same hash value.

Generally speaking, the two last properties are not true, and theoretically one can violate them. But currently this requires huge computational resources and time; that is why it is supposed to be infeasible.

The idea of using hash values is to have some information about the message without knowing the message itself. For example, it is used when producing digital signature: instead of running digital signature scheme over the whole message, it is usually run over its digest. In this way all the desired properties are preserved, while the size of the signature as well as the time for the signature calculation are decreased.

As we could see in one of the introductory example, hash is also used for authentication. Two parties sharing a secret k may authenticate themselves by a simple procedure: one sends a random value x to another and he returns a hash value of some message that involves both k and x. The first party can calculate the same value and check with the received one. Since no one can deduce k from this hash value, k stays secret. Moreover, as only one who knows k is able to calculate this value, the first party may be sure of the identity of the respondent. This idea is used in hash-based message authentication code (HMAC) [BCK96].

Some wide-spread examples of hash-functions are MD5 [Riv92], SHA-512 [rH06].

The hashing will be referred as apply (h, m) (stands for apply (hash)function h to message m) in Part I and as $h(m)$ in Part II.

2.2 Operations on messages

We introduced the cryptographic primitives we are going to consider. But what about their usage? How the messages may be constructed and analyzed by the communicating participants, i.e. what operations may be performed on the messages?

We consider a quite standard and natural set of abstract operations (*deduction rules*) that are normally referred to a *Dolev-Yao* model. That is,

- A message may be encrypted using symmetric or asymmetric algorithms if the key and the message are known;
- An encrypted message may be decrypted using symmetric (resp. asymmetric) algorithm under condition that the encryption key (resp. corresponding private key) is known;
- Two messages can be concatenated;
- A concatenated message may be split into its parts;
- A message may be signed if some private key is known (and the signature might be checked);
- A message may be hashed with specific hash function, if the hash function is known.

We will also consider a set constructor:

- Messages may be grouped to a set (one can build a set from messages he knows), and
- From a set of messages, one can extract each element.

These rules will be formalized later (see, e.g., Tables 2.1 and 2.4).

The basic pieces of information (e.g. agent's name or his public key) will be presented as atomic data. Using operations shown above, more complex messages may be constructed. For example, from atomic messages a, b, c one may build a message where b concatenated with c and the result encrypted by a symmetric key obtained by concatenation of a and c. This message will be denoted in our formalism as enc $(\text{pair}\,(b, c)\,, \text{pair}\,(a, c))$.

In the meantime, if we able to compute pair (a, c) (for example, from atoms a and c), then from the message enc $(\text{pair}\,(b, c)\,, \text{pair}\,(a, c))$ we can extract the value represented by atom b using the deduction rules.

2.3 Communicating parties behavior

The behavior of the communicating parties we consider in this work is "linear". More precisely, we consider a *sequence* of message patterns one receives and sends. If we refer to the example with spies presented at page 2, then Spy has a sequence of three actions to do: send question,

receive answer and send an encrypted message. Similarly, the Messenger's sequence is: receive question, send answer and receive an encrypted message.

This behavior is modeled using *strands* and will be formalized in Part II (§ 4.1.4 at p. 73).

Running ahead, we want to note that the linear behavior could be enriched with non-deterministic choice points which allow to define branching behavior of the communicating parties. That would require some changes in the models we present, but will not affect the decidability.

3 Contributions

In short, we present a technique for dealing with cryptographic primitives and Dolev-Yao deduction rules and show what we can do using it. From the point of view of applications, in this work we intend to advance the state of the art on the composition of Web Services taking into account their security policies, and the verification of cryptographic protocols; and from purely theoretical point of view — on the resolution of deducibility constraint systems.

More details follow.

3.1 Solving deducibility constraints

We present an NP decision procedure for solving general deducibility constraint systems for the case of Dolev-Yao deduction rules and corresponding signature optionally extended with associative commutative idempotent (ACI) symbol.

The only restriction we have is the use of atomic keys for asymmetric encryption and signature, but our model for symmetric encryption admits complex keys.

The closest work as for our knowledge is the PhD thesis of L. Mazaré [Maz06] where the well-formedness property for considered constraints was also relaxed, but his model did not admit complex keys. Moreover, no equational theory was considered.

Moreover, we are not aware about any work on satisfiability of well-formed constraints that consider ACI equational theory.

3.2 Composition of Web Services

We present a model for distributed orchestration of Web Services under constraints of their security policies and *non-disclosure condition*.

The behavior of Web Services is represented as a sequence of actions (receive/send) that they execute. Each action is annotated with a message pattern (first-order term). The actions of a Web Service are linked through the variables used in the patterns, that is, once a variable is instantiated, its value should be used in the rest of actions. Moreover, message patterns admit security primitives that allow one to express Web Services's security policies.

Given a client specified as a stand-alone Web Service and a set of available services, the problem is to build a set of communicating mediators deployed in *partner* organizations that are able to satisfy all client's requests. For doing this, the mediators are able to invoke available services, internally apply Dolev-Yao like operations on the collected messages as well as on initially given ones and communicate between each other. This communication is restricted by allowed message patterns and non-disclosure condition that forbids sending the sensitive data to other partners.

Up to our knowledge this is the first distributed model of Web Services composition which takes into account message structure and security primitives expressing security policies that admits automatic decision procedure.

A non-distributed case of Web Services orchestration not requiring new techniques, but still novel and interesting (automation of approach presented in [CMR08, CMR09]), was implemented in the context of AVANTSSAR and validated on several industrial case studies.

3.3 Verification of cryptographic protocols

We propose a new intruder model that assumes multiple non-communicating intruders and present in this setting a decision procedure for the protocol insecurity problem (for the case of secrecy) with bounded number of sessions.

While the insecurity of a protocol session in our model implies an insecurity in Dolev-Yao attacker model, it is useful to consider such a weaker model, since the Dolev-Yao intruder may be too powerful in scenarios, where some restrictions may be provided by physical, technical and administrative ways.

We also show how to model some attacks based on XML presentation of messages thanks to ACI operator that admit to model sets of XML nodes.

4 Plan of the thesis

4.1 Deducibility constraint systems

Part I is devoted to decision procedure for satisfiability of general constraint system with regard to DY deduction rules and ACI equational theory. This part provides a basis for the techniques presented in Part II.

After presenting the idea about the nature of constraints we consider, we give a short overview of related works in § 1.1. Then we introduce some basic general formalisms, like term algebra and constraint systems in § 1.2.

In Chapter 2 we present a decision procedure for satisfiability of general constraint systems for DY and DY+ACI deduction rules.

First, we formalize signature, equational theory and deduction system we are working on and also introduce specific notions that will be used throughout the part (§ 2.1.1). In the same section we present some general properties with proofs that depict different relations between the introduced notions and that will be used further for main proofs.

In § 2.1.2 we present an algorithm for checking[11] whether a given ground term can be derived from a given set of ground terms within DY+ACI theory and prove its correctness.

In § 2.1.3 we show existence of so-called *conservative solution*, that is, a solution that can be unambiguously defined only by a set of quasi-subterms of a constraint system and a set of public keys that are also used in the constraint system.

This give us some bound on a search space for such solutions which is calculated in § 2.1.4. Using this bound we present a non-deterministic algorithm to find a solution. Since this algorithm is proved to be correct and complete, we obtain a decidability result for general constraints within DY+ACI theory.

In § 2.2 we reduce the result to the pure DY deduction system (without ACI symbol).

The objective of Chapter 3 is to identify the complexity class of the presented algorithms which appears to be NP-complete.

[11] This problem is sometimes called *intruder deduction problem*.

First, a measure of input is formalized and justified (§ 3.1). Then, mainly using general properties presented in § 2.1.1 and the result of the following section, it was shown that satisfiability of general constraint systems is in NP and moreover, reusing a proof for well-formed constraints, it is possible to show that the problem is NP-complete (§ 3.3).

In § 3.2 we show that a problem of ground derivability presented in § 2.1.2 is polynomial thereby solving the intruder deduction problem in the presence of ACI symbol. This result in addition to be a stand-alone one, was used beforehand in the previous section.

Finally, in order to underline the importance of well-formedness assumption we show undecidability of general constraints for subterm deduction systems (which is known to be decidable for the well-formed case) and then conclude.

4.2 Web Services composition

The chapter is dedicated to a problem of Web Services composition, more precisely, a problem of automatic generation of distributed orchestration that takes into account security policies of available services and prevent a direct leakage of sensitive data between the partners that cooperate in order to ensure the distributed orchestration.

We introduce the problem of automatic Web Services composition and its variations, that is, shortly present the approaches proposed by different research groups in § 4.1. We also explain some notions we use, like Mediator, and show how we model Web Services and why.

Then, in § 4.2 we present an example of orchestration and then extend it to a distributed case. For both cases an instance of solution procedure is explained in semi-formal way in order to give some initial ideas.

We formalize the problem in § 4.3, that is, its input and output and present also an execution model as a transition system. We also give a procedure for reducing distributed orchestration problem to a resolution of (general) deducibility constraint systems with non-disclosure condition in § 4.4. The resolution itself relies on results given in Part I.

In § 4.5 we report an implementation called AVANTSSAR Orchestrator of the presented technique for non-distributed case of Web Services orchestration. First we describe the context in which the Orchestrator is used as a part of AVANTSSAR Validation Platform. Then shortly describe input/output language of the tool, in particular how Web Services are represented and how to specify an orchestration problem. Then, we present a general idea of the Orchestrator, that is, how orchestration problem can be reduced to a state reachability problem in cryptographic protocol analysis. After this, we describe the tool architecture and give some details on its components. We also consider a simple example of orchestration problem and show how it can be modeled and solved with AVANTSSAR Orchestrator. Finally, we reported some case studies on which the tool has been assessed.

A summary, advantages and disadvantages of our approach as well as possible research directions for extending this work are given in § 4.6.

4.3 Cryptographic Protocols

In Chapter 5 we show the application of general constraint systems in the domain of cryptographic protocols.

First, in § 5.1 we give a short review of intruder models which are used when analyzing the security of cryptographic protocols. In the same section we also recall how well-formed constraints are used to solve protocol insecurity problem for the fixed protocol sessions in a case of active Dolev-Yao intruder.

Then, we present an example motivating a new model of intruder. This model support multiple local intruders that are not able to communicate during attack. However, at some point when the intruders quit the protocol execution, the collected by these intruders information might be combined in order to deduce some secret.

This model is formalized in § 5.3 and reduced to the satisfiability of constraint systems, which is shown in § 5.4.

Taking advantage of ACI symbol considered in Part I, we show in § 5.5 how one can model a special kind of attacks on protocols which use XML format for messages.

Traditionally, we finish the chapter with conclusions, § 5.6.

Part I

Deducibility constraint systems

Chapter 1

Introduction

In general, a constraint is an additional condition that a solution of some problem must satisfy. For example, a problem to find a value of X such that $\sin(X) = 1$ has infinitely many solutions $X = \pi/2 + 2\pi n$ for any $n \in \mathbb{N}$. On the other hand, we may not be interested in all solutions, but only in those that satisfy additional condition, like $X \in [0, \pi]$. This condition is called a *constraint* to the initial problem and restricts the set of possible solutions to the only one $X = \pi/2$.

In this work we will consider another kind of constraints that will be used to restrict possible communications. That is, if we are asked to construct some communication, the additional constraint should make this communication feasible with regard to some given rules of the "game".

Let us consider an example. Suppose there is a crazy-about-security professor who rejects any message one tries to mail him if it is not encrypted with some secret asymmetric key k. The professor is not completely crazy, and he knows the private key $\mathrm{priv}(k)$, thus is able to decrypt his mails. His student wants to ask him a question q. What message should he send such that the professor at least will not reject it and could read the question?

One of the evident solutions could be to send his question q encrypted with key k (in our formalism this message is presented as $\mathrm{aenc}(q, k)$). But can the student really produce this message? It depends on data he possesses and a set of operations he can perform for constructing messages. For example, if he does not know the secret key k, normally he will be unable to encrypt his question with unknown key (matter of his *knowledge*). The same will hold, if he knows the key, but don't know how to encrypt (matter of *deduction rules*). One must take into account these two aspects in order to make the communication feasible.

Let us return to the second part of the question we asked: the professor should be able to read the question q. Since the student knows another message he copied out one day from the professor's screen — an encrypted email $\mathrm{aenc}(u, k)$ sent by someone else, he could send this message and it will be accepted by the professor, but in this way the question q student want to ask will not attain the addressee.

To represent the first condition of the question we asked, we present the following *deducibility constraint*:

$$\{q, k, \mathrm{aenc}(u, k)\} \triangleright \mathrm{aenc}(X, k) \tag{1.1}$$

It says that from the knowledge of the student $q, k, \mathrm{aenc}(p, k)$ (not too much even as for a student, but OK for the example) he is able to construct some message $\mathrm{aenc}(X, k)$ that we are sure will not be rejected by the professor. Here X is a variable such that whatever value it has, professor will accept the message since it is encrypted with k. Thus, we must find such value

of X that after substituting it the message can be produced by the student. The rules that the student can apply are ones we discussed earlier (see p. 9), in particular, he can encrypt messages. Note here we can have infinitely many solutions, like, e.g., $X \mapsto u$ — student simply transfers a stolen message; $X \mapsto k$ — student sends key k encrypted with itself; $X \mapsto \text{pair}(k, q)$ — student sends key k concatenated with his question q, all encrypted with k; $X \mapsto q$ — student sends his question q encrypted with k; $X \mapsto \text{aenc}(q, k)$ — student sends his question q encrypted twice with key k, etc.

To represent the second condition of the question we asked, we have to consider another deducibility constraint:

$$\{k, \text{priv}(k), \text{aenc}(X, k)\} \triangleright q \tag{1.2}$$

That is, using the same set of rules, in particular asymmetric decryption with private key $\text{priv}(k)$, whether the professor can infer question q from the received message and his knowledge.

We must consider constraints 1.1 and 1.2 together as a *deducibility constraint system*

$$\begin{Bmatrix} \{q, k, \text{aenc}(u, k)\} \triangleright \text{aenc}(X, k) \\ \{k, \text{priv}(k), \text{aenc}(X, k)\} \triangleright q \end{Bmatrix}$$

in order to find an appropriate value for X that will answer our question.

This part is dedicated to solving such deducibility constraint systems.

1.1 Related works

Solving deducibility constraint[12] systems is a research branch of cryptographic protocol analysis with bounded number of sessions. The overwhelming majority of works consider so-called *well-formed constraints*, a subset of constraint systems that satisfies two assumptions: knowledge monotonicity and variable origination. In short, the first is due to considering one central communicating entity, the intruder, that never forgets the collected knowledge; and the second is due to the deterministic behavior of other participants in the sense of message production: once the inputs up to a sending action are set, the next message to send is fixed.

Summing up

In this section we will list works dealing with different deducibility constraint systems.

Enriching standard Dolev-Yao intruder model [DY83] with different equational theories like exclusive OR, modular exponentiation, Abelian groups, etc. [CKRT05, DLLT08] and their combinations [BMV05a, CR10b] helps to find flaws that could not be detected considering free symbols only.

In this section we give a small survey of literature[13] that tackles the problem of satisfiability of constraint systems. We group it by considered deduction systems and equational theory.

1.1.1 Well-formed constraints

As was mentioned above, the wast majority of works consider well-formed constraints, and only few tries to relax this property. This is quite natural, since to reason about protocol insecurity with Dolev-Yao intruder, it is enough to consider only this restricted class of constraint systems (see, e.g., § 5.1.2 at p. 119).

A well-formed constraint system is a constraint system that satisfies two well-known properties, namely

[12] Also referred as *symbolic inference constraints* [Shm04] and *deductibility constraints* [Del06b]

[13] Some additional details can be found in [CDL06]

- (*Knowledge*) *monotonicity* and
- (*Variable*) *origination property.*

Knowledge monotonicity says that there exist a linear ordering $\leq$ on constraints in a system, such that if for two constraints $(E \triangleright p) \leq (K \triangleright q)$, then $E \subseteq K$. It shows that intruder's knowledge increases: E and K represent it in different steps.

The variable origination says that using the same order $\leq$, if in some constraint $K \triangleright q$ some variable X appears as a subterm in knowledge K then it should also appear as a subterm in some p such that $(E \triangleright p) \leq (K \triangleright q)$. It shows that the intruder can have in his knowledge only variables whose values he previously "instantiated".

The formal definitions of these two properties are presented in § 3.4, p. 60.

Pure Dolev-Yao (DY) deduction system

The deduction rules based on DY deduction system is one of the most interesting since it allows one to reason about cryptographic protocol insecurity with standard cryptographic primitives.

The first work that introduced a notion of constraints is [MS01] (see also [CV01]) and it considers DY deduction rules very similar to one we consider in Table 2.4 (see page 51): they allow to construct a message by pairing two messages, encrypt using symmetric and asymmetric encryptions, sign a message and hash it; it is also allowed to split a paired message into parts and decrypt an encrypted message under condition that the needed key is known.

In this work, the authors also first defined the property of a constraint system to be well-formed. They have shown that this property holds for constraint systems that they build in order to solve the cryptographic protocol insecurity problem under conditions of a Dolev-Yao intruder.

A reduction procedure to solve a system of constraints was presented: a set of rewriting rules that is applied non-deterministically and can reduce the initial system of constraints into some satisfiable *simple* form[14] (where the left-hand side of each remaining constraint is a variable) if and only if the initial constraint system is satisfiable.

Satisfiability of well-formed constraints was proved decidable and NP-complete in [RT01] but with a different technique.

DY and Exclusive OR (XOR)

Since some cryptographic protocols use a XOR operator ($\oplus$), it is useful to consider its properties, that is XOR being an associative ($((a \oplus b) \oplus c) = (a \oplus (b \oplus c))$), commutative ($a \oplus b = b \oplus a$) and nilpotent ($a \oplus a = 0$) symbol. A classical example is Bull-Otway Recursive Authentication protocol [BO97]. This protocol was proved to be correct [Pau97] when XOR was considered as a free symbol, but later, an attack exploiting properties of XOR was found [RS98].

Motivated by this example, relaxing the perfect cryptography assumption has been considered in [CKRT03a] and [CLC03] in order to take into account XOR's properties. In contrast to [CKRT03a], the authors of [CLC03] used explicitly the notion of "constraint" when dealing with the protocol insecurity problem, but they only proved the decidability of their approach without giving a complexity class, while in the former, the problem was shown to be in NP.

Well-formed constraint systems over the Dolev-Yao signature extended with XOR and a homomorphic function h was considered in [DLLT06]. One of the motivating facts was the usage of a checksum function in Wired Equivalent Privacy protocol (WEP) [Wir97], that has

[14] Referred in later works as *solved form*

homomorphic property with regard to XOR: $h(X \oplus Y) = h(X) \oplus h(Y)$. Satisfiability of such constraint system was proved to be decidable.

DY, modular exponentiation and Abelian groups

The modular exponentiation is widely used in cryptographic public key schemes like Diffie-Hellman and RSA. From here growths the interest to this operation. For example, several flaws were detected in some variants of Group Diffie-Hellman algorithms (GDH) due to some properties of exponentiation [PQ01].

The modular exponentiation has the following two properties: $a^1 = a$ and $(a^b)^c = a^{b \times c}$. The exponents form an Abelian group w.r.t. multiplication $\times$, that is $\times$ is an associative, commutative operator; there exists an identity element 1 ($a \times 1 = 1 \times a = a$), and every element has an inverse ($\forall a \exists a^{-1} : a \times a^{-1} = 1$).

Some restricted cases of modular exponentiation are considered in [CKRT03b] (variables cannot be substituted by products) and [MS03] (base of exponentiation is a fixed constant). In both works products are allowed only in exponents. In the latter the decision procedure was not presented, but only reduction to a system of quadratic Diophantine equations, however in the further work [Shm04] V.Shmatikov succeeded to show the decidability in a restricted case. Note that this work considers multiplication (Abelian group operator) also outside the exponents.

In general, if a distributivity property of the modular exponentiation is also assumed, the satisfiability of well-formed constraints is undecidable due to undecidability of unification modulo the considered equational theory [KNW03].

The same problem for the case of Dolev-Yao theory enriched with Abelian group operator with homomorphism (AGh) is shown to be undecidable in [Del06b] by reducing a formulation of the Hilbert 10th problem to resolution of well-formed deducibility constraints satisfiability problem modulo AGh.

DY and Prefix rule

Some cryptographic schemes (mainly symmetric) may be used in a mode called Cipher Block Chaining (CBC). The idea is to split plaintext to blocks and encrypt each block separately, but every next block of plaintext is previously XORed with the previous encrypted block. This mode has an obvious property that can be used: if the plaintext m consists of several data elements (e.g. $m = \mathrm{pair}(a, b)$) and the first one has the length of a block, then from the encrypted chain $\mathrm{enc}(m, k) = \mathrm{enc}(\mathrm{pair}(a, b), k)$ one can slice a first element encrypted by the same key without knowing it.

Using this argument an additional to the standard Dolev-Yao rule was considered in in [CKRT05]: $\mathrm{enc}(\mathrm{pair}(X, Y), K) \to \mathrm{enc}(X, K)$. The problem of protocol insecurity with fixed number of sessions (which is equivalent to satisfiability of well-formed constraints) was proved to be in NP also in this case.

Commutativity of public-key encryption

An interesting property that we can meet, for example, in a well-known asymmetric cryptographic scheme RSA [RSA78] is the *commuting encryption* (if the RSA scheme uses some fixed modulus). Since an encryption in such scheme is a modular exponentiation $\mathrm{aenc}(m, k) = m^k \bmod n$, then $\mathrm{aenc}(\mathrm{aenc}(m, k_1), k_2) = (m^{k_1} \bmod n)^{k_2} \bmod n = (m^{k_2} \bmod n)^{k_1} \bmod n =$ $\mathrm{aenc}(\mathrm{aenc}(m, k_2), k_1)$.

The bounded sessions protocol insecurity problem with public key encryption having the mentioned property was considered in [CKRT04]. Since the modular exponentiation models this property and it was considered in addition to the encryption in [CKRT03b], the authors concluded (by following the proof presented in their mentioned work) that the bounded sessions protocol insecurity problem (and, consequently, a satisfiability of well-formed constraint systems) modulo commuting encryption is NP-complete.

Combination of theories

An approach that allows one to combine different equational theories with disjoint signatures was presented in [CR10b]. It was shown, and proposed the corresponding algorithm, that if a problem of *ordered satisfiability* of well-formed constraints are decidable in each intruder theory (intruder composition rules and equational theory), then the satisfiability of well-formed constraints is decidable in the joint intruder theory. The ordered satisfiability problem is a modification of satisfiability one which takes in input an arbitrary linear ordering on variables and atoms, and a solution must preserve this order[15].

As an example the authors proved the decidability of the ordered satisfiability for standard Dolev-Yao theory (only with symmetric encryption), XOR theory and Abelian Group theory and consequently showed that its combination is also decidable. Note that this work considers explicit destructors in order to move all decomposition rules to equational theory.

1.1.2 General constraints

As was mentioned above, we are aware of only few works that relax well-formedness property. Here we give its short overview.

Dolev-Yao deduction system

An attempt to swerve from well-formed constraints considering Dolev-Yao deduction system was done by L.Mazaré first in [Maz05]. He *partially* relaxed knowledge monotonicity and called this kind of constraints *quasi well-formed* and claimed the NP-completeness of satisfiability of such constraints. If we express the condition of a constraint system $\{E_i \triangleright t_i\}_{i=1,\dots,n}$ to be quasi well-formed in our notation, then it would look like $x \in \mathrm{Vars}(E_i) \implies \exists j < i : x \in \mathrm{Vars}(t_j) \wedge E_j \not\subseteq E_i$.

Later, in his thesis [Maz06], he succeeded to avoid usage of well-formed properties and considered *general* constraint systems. He presented a decision procedure for satisfiability of general constraint systems, but with one restriction: keys used for encryption are atomic. We will relax the former restriction and show that satisfiability of general Dolev-Yao constraints is decidable and NP-hard.

DY and Associative commutative idempotent symbol

This equational theory is considered in the present document, as well as in our previous publications [ACRT10], [ACRT11] and [ACRTar]. The satisfiability of general deducibility constraint systems is NP-complete problem also for this case.

[15] A solution is represented by a substitution that sends variables to ground terms, and it is valid iff it satisfies the corresponding satisfiability problem and the value of any variable contains only those atoms as its subterm which are less then the variable itself

Stand-alone AC constraints

In [BCLD07] the authors considered deducibility constraints with the only symbol having associative and commutative properties, so-called *AC-constraints*. The intruder can only compose messages, but no decomposition rules allowed. By reducing a special form of Diophantine equations known undecidable to satisfiability of a system of deducibility constraints, they concluded undecidability of the AC-constraints. They also showed that a subclass of well-formed AC-constraints called *simple* to be decidable.

1.2 Terms and notions

We present here some general definitions of *term*, *substitution*, *equational theory*, *deduction rules*, *derivation*, *constraint system*, etc. Later some definitions may be precised since we will work on some instances of them, e.g. *DY deduction rules*, *ACI equational theory*.

1.2.1 Term algebra

We give some definitions related to term algebra (in the spirit of, e.g., [Ohl02]).

Definition 1.2.1 A *signature* is an at most countable set $\mathcal{F}$ of *functional symbols*, where every $f \in \mathcal{F}$ associated with a positive natural number called *arity* (the number of arguments it is supposed to have), i.e. $arity : \mathcal{F} \mapsto \mathbb{N}^+$.

Definition 1.2.2 Let $\mathcal{F}$ be a signature, $\mathcal{X}$ be an at most countable set of *variables* and $\mathcal{A}$ be an at most countable set of *atoms*[16]. The set of *terms* $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ is defined to be the smallest set such that

- $\mathcal{A} \cup \mathcal{X} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$.
- $\forall f \in \mathcal{F}$ if $n = arity(f)$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ then $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$.

For simplicity we will write $\mathcal{T}$ instead of $\mathcal{T}(\mathcal{F},\mathcal{A},\mathcal{X})$.

Definition 1.2.3 For $t \in \mathcal{T}$, $\mathrm{root}(t)$ denotes the root symbol of t and is defined by:

$$\mathrm{root}(t) = \begin{cases} f, & \text{if } t = f(t_1, \ldots, t_n), \text{ where } n = arity(f), \\ t, & \text{if } t \in \mathcal{X} \cup \mathcal{A}. \end{cases}$$

Definition 1.2.4 Let t be a term. We define a set of *subterms* $\mathrm{Sub}(t)$ as follows:

$$\mathrm{Sub}(t) = \begin{cases} \{t\}, & \text{if } t \in \mathcal{X} \cup \mathcal{A}; \\ \{t\} \bigcup\limits_{i=1,\ldots,n} \mathrm{Sub}(t_i), & \text{if } t = f(t_1, \ldots, t_n), \text{ where } n = arity(f). \end{cases}$$

Definition 1.2.5 Let t be a term. We define a set of its variables $\mathrm{Vars}(t)$ by $\mathrm{Vars}(t) = \mathcal{X} \cap \mathrm{Sub}(t)$. A term m that does not contain variables, i.e. such that $\mathrm{Vars}(m) = \varnothing$, is called *ground*. The set of all ground terms we denote as $\mathcal{T}_g$.

[16] Sets $\mathcal{F}$, $\mathcal{X}$ and $\mathcal{A}$ are considered to be pairwise disjoint

Definition 1.2.6 A *substitution* σ is a mapping from $\mathcal{X}$ to $\mathcal{T}$with the *domain* $\mathrm{dom}(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$. We present a substitution σ as $\{x \mapsto \sigma(x) \mid x \in \mathrm{dom}(\sigma)\}$. Every substitution σ extends uniquely to a function $\sigma : \mathcal{T} \mapsto \mathcal{T}$, where

$$\sigma(t) = \begin{cases} t, & \text{if } t \in \mathcal{A}; \\ \sigma(x), & \text{if } t = x \in \mathcal{X}; \\ f(\sigma(t_1), \ldots, \sigma(t_n)), & \text{if } t = f(t_1, \ldots, t_n), \text{ where } n = arity(f). \end{cases}$$

From now we consider only such substitutions σ that for any $x \in \mathrm{dom}(\sigma)$, $x \notin \mathrm{Vars}(\sigma(x))$. We often use postfix notation $t\sigma$ instead of $\sigma(t)$. We call a substitution σ *ground* if for every $x \in \mathrm{dom}(\sigma)$, the term $x\sigma$ is ground.

Example 11 Suppose we have three functional symbols: f, g and h with corresponding arities $1, 2$ and 2, i.e. $\mathcal{F} = \{f, g, h\}$ and $arity(f) = 1$, $arity(g) = arity(h) = 2$. Assume a set of atoms $\mathcal{A} = \{a, b, c\}$ and a set of variables $\mathcal{X} = \{x, y, z\}$. Then we give an example of a term we denote t, an element of $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$:

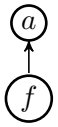
$$t = g(f(a), h(g(x, b), f(x))).$$

A term can be seen as a tree whose non-leaf nodes are labeled with functional symbols and leafs are labeled with variables or atoms. For example, a term presented above may be represented as shown in Figure 1.1a.

Let us consider a substitution $\sigma = \{x \mapsto f(a)\}$. Then the term obtained by application of substitution σ on term t is

$$t\sigma = g(f(a), h(g(f(a), b), f(f(a)))).$$

The tree representation of this term is given in Figure 1.1b.

Informally, every node in such tree representation corresponds to a subterm. That is, if we take any subtree of a tree representing term t, then this subtree represent some subterm of t. For example, a subtree  represent term $f(a)$ which is a subterm of t.

The converse is also true: every subterm of a term t corresponds to a subtree of a tree representation of t.

We note that there is a more "compact" representation of terms in form of Directed Acyclic Graph (DAG), where, informally, there is no two different nodes that represent the same (sub)term. The DAG representations of terms t and $t\sigma$ can be found in Figures 1.1c and 1.1d correspondingly.

1.2.2 Equational theories

Since we work with symbolic model of real world messages, we have to consider an equivalence between two terms that are syntactically different but represent the same bit-strings. For example, a bit-string representing a XOR of two values $1011 \oplus 0010$ is the same as one representing $0010 \oplus 1011$. Thus we want to express that $a \oplus b$ is equivalent to $b \oplus a$, while being, as you can see, syntactically different. For this, a notion of equational theory is introduced.

Definition 1.2.7 An *equational theory* $\mathcal{E}$ over set of terms $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ is a set of *identities* $\mathcal{E} = \{u_i = v_i\}_{i \in \mathbb{I}}$ where u_i and v_i are terms from $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{Y})$, $\mathcal{Y} \cap \mathcal{X} = \varnothing$ and $\mathbb{I}$ is an at most

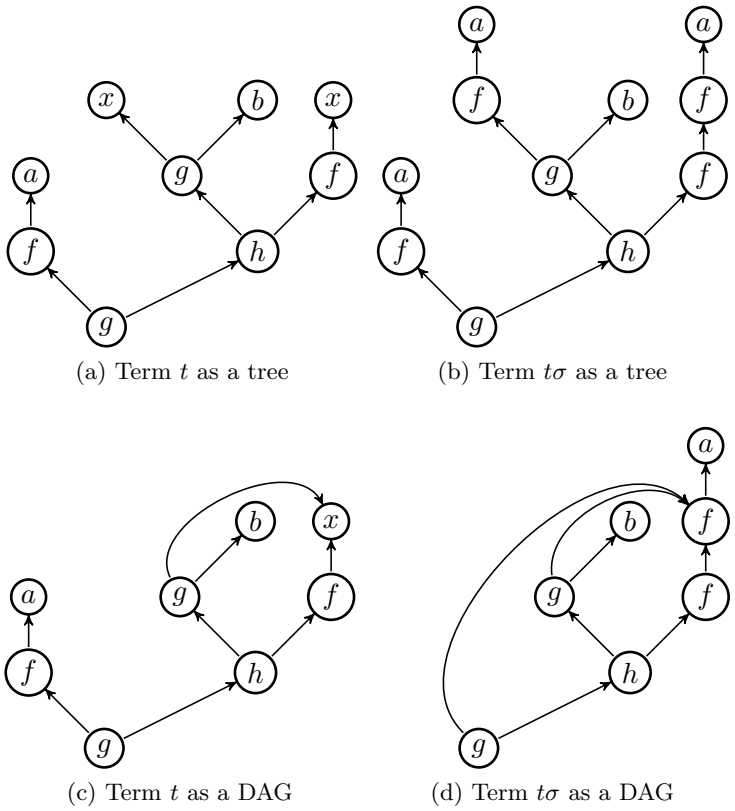


Figure 1.1: Tree and DAG representations of terms

countable set. An equational theory $\mathcal{E}$ generates some minimal binary relation of *congruence* $\equiv_{\mathcal{E}} \subseteq \mathcal{T} \times \mathcal{T}$ on the term algebra that contains all pairs of terms $p \equiv_{\mathcal{E}} q$ whenever there exist $(u = v) \in \mathcal{E}$ and substitution $\sigma : \mathcal{Y} \mapsto \mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ such that $p = u\sigma$ and $q = v\sigma$ ($\equiv_{\mathcal{E}}$ is our notation for $\overset{*}{\leftrightarrow}_E$ used in in [BN98], where you can find all missing here details). The relation of congruence is a binary relation which is

- reflective ($p \equiv_{\mathcal{E}} p$)
- symmetric ($p \equiv_{\mathcal{E}} q$ implies $q \equiv_{\mathcal{E}} p$)
- and transitive ($p \equiv_{\mathcal{E}} r$, $r \equiv_{\mathcal{E}} q$ implies $p \equiv_{\mathcal{E}} q$)

and is compatible with the term algebra, i.e. $t_i \equiv_{\mathcal{E}} s_i$ for $i = 1, \ldots, n$ implies $f(t_1, \ldots, t_n) \equiv_{\mathcal{E}} f(s_1, \ldots, t_n)$ for any n-ary functional symbol of $\mathcal{F}$. We write $\equiv$ instead of $\equiv_{\mathcal{E}}$ if from the context is clear what equational theory is used.

Definition 1.2.8 An equational theory $\mathcal{E}$ partitions the set of terms $\mathcal{T}(\mathcal{F}, \mathcal{A}, \mathcal{X})$ into *equivalence classes*. An equivalence class $[t]_{\mathcal{E}}$ of a term t is a set of all equivalent to it elements of $\mathcal{T}$, i.e. $[t]_{\mathcal{E}} = \{p \in \mathcal{T} \mid p \equiv t\}$. If we can fix an element from every class of equivalence, we can call it a *normal form*. We denote a normal form of a class of equivalence $[t]_{\mathcal{E}}$ by $\ulcorner t \urcorner$.

1.2.3 Derivations and constraint systems

Here we present a formal definition of deduction rules, derivations, constraint systems and its models. The deduction rules are needed to specify the operations that can be performed on the

symbolic representation of real-world messages.

Definition 1.2.9 A (deduction) *rule* is a tuple of terms written as $s_1, \ldots, s_k \to s$, where $s_1, \ldots, s_k, s$ are terms. A *deduction system* $\mathcal{D}$ is a set of deduction rules.

We suppose that the considered further rules belong to some fixed deduction system $\mathcal{D}$.

Definition 1.2.10 A *ground instance* of a rule $d = s_1, \ldots, s_k \to s$ is a rule $l = l_1, \ldots, l_k \to r$ where $l_1, \ldots, l_k, r$ are ground terms and there exists σ — ground substitution, such that $l_i = s_i\sigma, \forall i = 1, \ldots, k$ and $r = s\sigma$. We also call a ground instance of a rule a *ground rule* (or sometimes even *rule* when there is no ambiguity).

Given two sets of ground terms E, F and a ground rule $l \to r$, we write $E \to_{l \to r} F$ iff $F = E \cup \{r\}$ and $l \subseteq E$, where l is a set of terms. We write $E \to F$ iff there exists ground rule $l \to r$ such that $E \to_{l \to r} F$.

Definition 1.2.11 A *derivation* D of length $n \geq 0$ is a sequence of finite sets of ground terms $E_0, E_1, \ldots, E_n$ such that $E_0 \to E_1 \to \cdots \to E_n$, where $E_i = E_{i-1} \cup \{t_i\}, \forall i = \{1, \ldots, n\}$. A term t is *derivable* from a set of terms E iff there exists a derivation $D = E_0, \ldots, E_n$ such that $E_0 = E$ and $t \in E_n$. A set of terms T is derivable from E, iff every $t \in T$ is derivable from E. We denote $\mathrm{Der}(E)$ set of terms derivable from E.

A derivation represents a possible step-by-step evolution of a set of terms; this evolution is feasible in the sense that the every next set is obtained by an application of some deduction rule on terms from the previous set.

Definition 1.2.12 Let E be a set of terms and t be a term, we define the couple (E, t) denoted $E \triangleright t$ to be a *constraint*. A *constraint system* is a set

$$\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$$

where n is an integer and $E_i \triangleright t_i$ is a constraint for all $i \in \{1, \ldots, n\}$.

We extend the definition of $\mathrm{Vars}(\cdot)$ to constraint system $\mathcal{S}$ in a natural way: $\mathrm{Vars}(\mathcal{S})$ is a set of variables used in $\mathcal{S}$. We will say that $\mathcal{S}$ is normalized, if every term occurring in $\mathcal{S}$ is normalized. As $\ulcorner \mathcal{S} \urcorner$ we will denote a constraint system $\{\ulcorner E_i \urcorner \triangleright \ulcorner t_i \urcorner\}_{i=1,\ldots,n}$.

Definition 1.2.13 A ground substitution σ is a *model* of constraint $E \triangleright t$ (or σ satisfies this constraint), if $\ulcorner t\sigma \urcorner \in \mathrm{Der}(\ulcorner E\sigma \urcorner)$. A ground substitution σ is a *model* of a constraint system $\mathcal{S}$, if it satisfies all the constraints of $\mathcal{S}$ and $\mathrm{dom}(\sigma) = \mathrm{Vars}(\mathcal{S})$.

Informally, the constraint express a possibility to derive a term from a set of terms; and its model gives such values to variables (of the constraint), that there exists a derivation starting from the set of terms in the LHS of the instantiated constraint and ending with a set of terms which contains instantiated RHS of the constraint. And every operation is done modulo considered equational theory, since the real-world messages from the one class of equivalence should be indistinguishable.

Chapter 2

Solving general deducibility constraint systems

Contents

2.1 Dolev-Yao constraints extended with ACI symbol 27
2.1.1 Formal introduction to the problem 28
2.1.2 Ground case of DY+ACI 40
2.1.3 Existence of conservative solutions 43
2.1.4 Bounds on conservative solutions 47
2.2 Pure Dolev-Yao constraints 50

2.1 Dolev-Yao constraints extended with ACI symbol

In this section we present a decision procedure for the problem of satisfiability of general constraint systems where Dolev-Yao deduction system is extended with some deduction rules for an associative-commutative-idempotent symbol (DY+ACI). We consider operators for pairing, symmetric and asymmetric encryptions, decryption, signature, hashing and an ACI operator that will be used as a set constructor.

As for the proof structure, we will generally follow the ideas presented in, e.g., [MS05, MS03] or [RT01]. After introducing the formal notations, the main steps to show the decidability are as follows:

1. We present an algorithm for solving a ground derivability in DY+ACI model (§ 2.1.2).
2. We show that the normalization does not affect the satisfiability of a constraint system by a substitution: either we normalize a substitution or a constraint system (Propositions 2 and 3).
3. We prove the existence of a conservative solution of satisfiable constraint system: a substitution σ that sends a variable to an ACI-set of quasi-subterms of the constraint system instantiated with σ and the private keys corresponding to the atomic values of the constraint system (§ 2.1.3).
4. We give a bound on the size of a conservative solution, and, as consequence, we obtain decidability (§ 2.1.4).

Term	Description
$\mathrm{enc}(t_1, t_2)$	t_1 encrypted with symmetric key t_2
$\mathrm{aenc}(t_1, t_2)$	t_1 encrypted with asymmetric public key t_2
$\mathrm{pair}(t_1, t_2)$	t_1 concatenated with t_2
$\mathrm{priv}(t_2)$	corresponding private key for public key t_2
$\mathrm{sig}(t_1, \mathrm{priv}(t_2))$	signature of message t_1 with private key $\mathrm{priv}(t_2)$
$\mathrm{apply}(t_1, t_2)$	apply hash function t_1 on message t_2
$\cdot(\{t_1, \ldots, t_n\})$	set of messages $t_1, \ldots, t_n$

Table 2.1: Functional symbols explanations

2.1.1 Formal introduction to the problem

Terms and notions

In definition 2.1.1 we instantiate the class of terms that we will consider for DY+ACI.

Definition 2.1.1 *Terms* are defined according to the following grammar:

$$\begin{aligned}
term &::= variable \mid atom \mid \mathrm{pair}(term, term) \mid \\
&\quad \mathrm{enc}(term, term) \mid \cdot(tlist) \mid \mathrm{priv}(Keys) \mid \\
&\quad \mathrm{aenc}(term, Keys) \mid \mathrm{sig}(term, \mathrm{priv}(Keys)) \mid \\
&\quad \mathrm{apply}(atom, term) \\
Keys &::= variable \mid atom \\
tlist &::= term \mid term, tlist
\end{aligned}$$

where $atom \in \mathcal{A}$, $variable \in \mathcal{X}$.

The short explanation of functional symbols is given in Table 2.1. Note that by $\mathrm{sig}(p, \mathrm{priv}(a))$ we mean only a signature of message p with private key $\mathrm{priv}(a)$ and do not assume that one can retrieve the message itself from the signature.

Remark that the given instantiation deviates from the classical definition of the term:

- First, we do not allow complex keys for asymmetric encryption. As a consequence, we have to introduce a restriction on substitution applications: substitution σ cannot be applied to the term t, if the result does not abide by the given grammar.

  Example 12 Substitution $\sigma = \{x \mapsto \mathrm{pair}(a, b)\}$ cannot be applied to the term $\mathrm{aenc}(a, x)$.

- Second, it seems like we use a symbol $\cdot$ without fixed arity. In fact here we specify $\cdot$ as a shortcut for an appropriate representative of the following functional symbols with fixed arities: $\bigcup_{i=1}^{\infty} \{\cdot_i\}$, where $arity(\cdot_i) = i$.

  Example 13 Writing $\cdot(\{t_1, t_2\})$ is a shortcut for $\cdot_2(\{t_1, t_2\})$, while $\cdot(\{t_1, t_2, t_3\})$ is a shortcut for $\cdot_3(\{t_1, t_2, t_3\})$.

Since we will deal with a list of terms (for $\cdot$), we have to define some notions on the lists. We denote a term on i-th position of a list L as $L[i]$. Then $t \in L$ is a shortcut for $\exists i : t = L[i]$. We also define two binary relations $\subseteq$ and $\approx$ on lists as follows: $L_1 \subseteq L_2$ if and only if $\forall t \in L_1 \implies t \in L_2$; $L_1 \approx L_2$ if and only if $L_1 \subseteq L_2$ and $L_2 \subseteq L_1$, and naturally extend them if L_1 or L_2 is a set.

In Definition 2.1.2 we present an equational theory, denoted as ACI, that is considered together with the Dolev-Yao deduction system in this chapter.

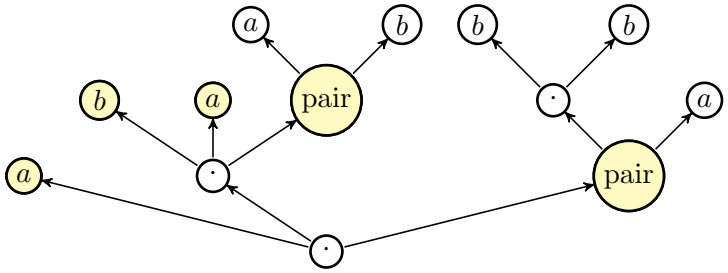


Figure 2.1: Illustration for Example 14: Elements of t (highlighted)

Definition 2.1.2 We consider symbol $\cdot$ to be

- associative: $\cdot(\{t_1,\ldots,t_k,\cdot(\{t_{k+1},\ldots,t_m\}),t_{m+1},\ldots,t_n\}) \equiv_{ACI} \cdot(\{t_1,\ldots,t_n\})$,
- commutative: $\cdot(\{t_1,t_2\}) \equiv_{ACI} \cdot(\{t_2,t_1\})$ and
- idempotent: $\cdot(\{t_1,t_1\}) \equiv_{ACI} \cdot(\{t_1\})$

Moreover, we suppose that $\cdot(\{t\}) \equiv_{ACI} t$. That is, the equational theory ACI is
$\{\cdot(\{y\}) = y, \quad \cdot(\{y_1,y_1\}) = \cdot(\{y_1\}), \quad \cdot(\{y_1,y_2\}) = \cdot(\{y_2,y_1\})\} \cup$
$\{\cdot(\{t_1,\ldots,t_k,\cdot(\{t_{k+1},\ldots,t_m\}),t_{m+1},\ldots,t_n\}) = \cdot(\{t_1,\ldots,t_n\})\}_{0\le k<m\le n}$.

Definition 2.1.3 For any term $t \in \mathcal{T}$ we define its *set of elements* by:

$$\mathrm{elems}(t) = \begin{cases} \bigcup_{p\in L} \mathrm{elems}(p) & \text{if } t = \cdot(L); \\ \{t\}, & \text{otherwise.} \end{cases}$$

We extend elems() to sets of terms or lists of terms T by $\mathrm{elems}(T) = \bigcup_{t\in T} \mathrm{elems}(t)$.

Example 14 Let us consider term $t = \cdot(\{a,\cdot(\{b,a,\mathrm{pair}(a,b)\}),\mathrm{pair}(\cdot(\{b,b\}),a)\})$. Set of its elements is $\mathrm{elems}(t) = \{a,b,\mathrm{pair}(\cdot(\{b,b\}),a),\mathrm{pair}(a,b)\}$. See Figure 2.1.

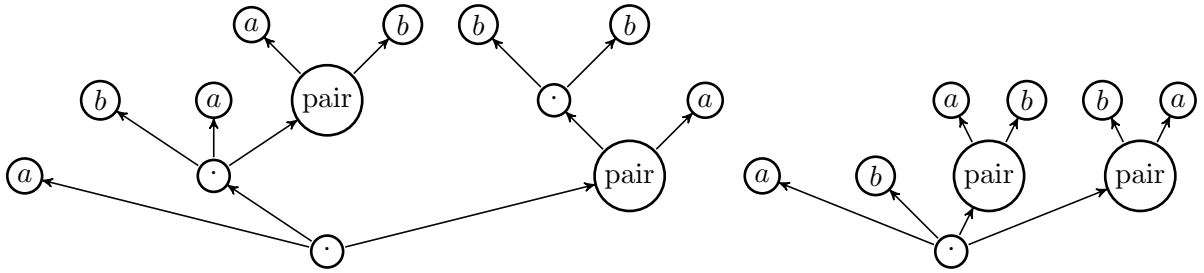
Definition 2.1.4 Let $\prec$ be a computable strict total order on $\mathcal{T}$, such that the question whether $p \prec q$ can be answered in polynomial time.

Definition 2.1.5 The cardinality of a set P is denoted by $|P|$.

We will use bin as a generalization of all binary operators: $\mathrm{bin} \in \{\mathrm{enc},\mathrm{aenc},\mathrm{pair},\mathrm{sig},\mathrm{apply}\}$.

Definition 2.1.6 The *normal form* of a term t (denoted by $\ulcorner t \urcorner$) is recursively defined by:

- $\ulcorner t \urcorner = t$, if $t \in \mathcal{X} \cup \mathcal{A}$
- $\ulcorner \mathrm{bin}(t_1,t_2) \urcorner = \mathrm{bin}(\ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner)$
- $\ulcorner \mathrm{priv}(t) \urcorner = \mathrm{priv}(\ulcorner t \urcorner)$
- $\ulcorner \cdot(L) \urcorner = \begin{cases} \cdot(L'), & \text{if } |\ulcorner \mathrm{elems}(L) \urcorner| > 1 \text{ and } L' \approx \ulcorner \mathrm{elems}(L) \urcorner \\ & \text{and for all } i<j,\ L'[i] \prec L'[j]; \\ t', & \text{if } \ulcorner \mathrm{elems}(L) \urcorner = \{t'\} \end{cases}$,



(a) Term t as a tree (b) Term $\ulcorner t \urcorner$ as a tree

Figure 2.2: Illustration for Example 15: Normal form of a term

where for set of terms T, $\ulcorner T \urcorner = \{\ulcorner t \urcorner : t \in T\}$.

We will call term t *normalized*, iff $t = \ulcorner t \urcorner$.

One can see that two terms are congruent modulo the ACI properties of $\cdot$ iff they have the same normal form.

Lemma 1. *For any two terms p and q, we have $p \equiv_{ACI} q$ if and only if $\ulcorner p \urcorner = \ulcorner q \urcorner$.*

Proof idea. For $p \equiv_{ACI} q \implies \ulcorner p \urcorner = \ulcorner q \urcorner$ it is enough to prove that if we apply any ACI-identity in any direction in any position of any term t, then the normal form after such application will not change. This is quite natural, since the normalization function deals with $\mathrm{elems}(\cdot)$ when meets an $\cdot$-node, and for any ACI-identity $u = v$ we have $\mathrm{elems}(u) = \mathrm{elems}(v)$.

For $\ulcorner p \urcorner = \ulcorner q \urcorner \implies p \equiv_{ACI} q$ it is enough to present a sequence of identities to apply on each position of any term t to obtain its normal form $\ulcorner t \urcorner$. We need to proceed bottom-up (from leafs to root) and once a node with root-symbol $\cdot$ is met (note that non-ACI nodes need no treatment) we flatten it using associativity identities from left to right, then using commutativity sort the children according to $\prec$ order and then using associativity from right to left we wrap equal terms with $\cdot$ node and then use identities of idempotency ($\cdot(\{y_1, y_1\}) = \cdot(\{y_1\})$ and $\cdot(\{y\}) = y$) to remove the duplicates. Thus, we may reduce p and q to the same term $\ulcorner p \urcorner$ (which is equal to $\ulcorner q \urcorner$) using ACI-identities. □

Other properties are stated in Lemma 4.

Example 15 For term $t = \cdot(\{a, \cdot(\{b, a, \mathrm{pair}(a, b)\}), \mathrm{pair}(\cdot(\{b, b\}), a)\})$ we have $\ulcorner t \urcorner = \cdot(\{a, b, \mathrm{pair}(a, b), \mathrm{pair}(b, a)\})$ (if $a \prec b \prec \mathrm{pair}(a, b) \prec \mathrm{pair}(b, a)$). See Figure 2.2.

Definition 2.1.7 Let t be a term. We define a set of *quasi-subterms* $\mathrm{QSub}(t)$ as follows:

$$\mathrm{QSub}(t) = \begin{cases} \{t\}, & \text{if } t \in \mathcal{X} \cup \mathcal{A}; \\ \{t\} \cup \mathrm{QSub}(t_1), & \text{if } t = \mathrm{priv}(t_1); \\ \{t\} \cup \mathrm{QSub}(t_1) \cup \mathrm{QSub}(t_2), & \text{if } t = \mathrm{bin}(t_1, t_2) \\ \{t\} \cup \bigcup_{p \in \mathrm{elems}(L)} \mathrm{QSub}(p), & \text{if } t = \cdot(L) \end{cases}$$

If T is a set of terms, then $\mathrm{QSub}(T) = \bigcup_{t \in T} \mathrm{QSub}(t)$. If $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$ is a constraint system, we define $\mathrm{QSub}(\mathcal{S}) = \bigcup_{t \in \bigcup_{i=1}^{n} E_i \cup \{t_i\}} \mathrm{QSub}(t)$.

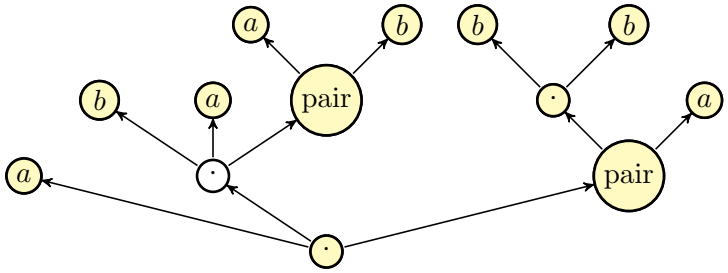


Figure 2.3: Illustration for Example 16: Quasi-subterms of t (highlighted)

Example 16 For term $t = \cdot(\{a, \cdot(\{b, a, \mathrm{pair}(a, b)\}), \mathrm{pair}(\cdot(\{b, b\}), a)\})$ we have

$$\mathrm{QSub}(t) = \{\cdot(\{a, \cdot(\{b, a, \mathrm{pair}(a, b)\}), \mathrm{pair}(\cdot(\{b, b\}), a)\}), \\ a, b, \mathrm{pair}(a, b), \mathrm{pair}(\cdot(\{b, b\}), a), \cdot(\{b, b\})\}.$$

See Figure 2.3.

We define $\mathrm{Sub}(t)$ as the set of subterms of t and the size of a term, as the number of its different subterms. This size will be often used as a "counter" for inductive proofs in this work.

Definition 2.1.8 Let t be a term. We give an instance of Definition 1.2.4 for our signature and define $\mathrm{Sub}(t)$ as follows:

$$\mathrm{Sub}(t) = \begin{cases} \{t\}, & \text{if } t \in \mathcal{X} \cup \mathcal{A}; \\ \{t\} \cup \mathrm{Sub}(t_1), & \text{if } t = \mathrm{priv}(t_1); \\ \{t\} \cup \mathrm{Sub}(t_1) \cup \mathrm{Sub}(t_2), & \text{if } t = \mathrm{bin}(t_1, t_2) \\ \{t\} \cup \bigcup_{p \in L} \mathrm{Sub}(p), & \text{if } t = \cdot(L). \end{cases}$$

If T is a set of terms, then $\mathrm{Sub}(T) = \bigcup_{t \in T} \mathrm{Sub}(t)$. If $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$ is a constraint system, we define $\mathrm{Sub}(\mathcal{S}) = \bigcup_{t \in \bigcup_{i=1}^{n} E_i \cup \{t_i\}} \mathrm{Sub}(t)$.

Example 17 For term $t = \cdot(\{a, \cdot(\{b, a, \mathrm{pair}(a, b)\}), \mathrm{pair}(\cdot(\{b, b\}), a)\})$ we have

$$\mathrm{Sub}(t) = \{\cdot(\{a, \cdot(\{b, a, \mathrm{pair}(a, b)\}), \mathrm{pair}(\cdot(\{b, b\}), a)\}), \\ \cdot(\{b, a, \mathrm{pair}(a, b)\}), \mathrm{pair}(\cdot(\{b, b\}), a), \\ a, b, \mathrm{pair}(a, b), \cdot(\{b, b\})\}.$$

Definition 2.1.9 We define a size of a term t as $\mathrm{size}(t) = |\mathrm{Sub}(t)|$, for set of terms T, $\mathrm{size}(T) = |\mathrm{Sub}(T)|$ and for constraint system $\mathcal{S}$ as $\mathrm{size}(\mathcal{S}) = |\mathrm{Sub}(\mathcal{S})|$.

Remark that such a definition of "size" does not polynomially approximate a number of bits needed to write the term or constraint system down (cf. Definition 3.1.2 in § 3.1, page 56). However, for normalized terms (but not constraint systems), it does.

We define a Dolev-Yao deduction system modulo ACI (denoted DY+ACI). It consists of composition rules and decomposition rules, depicted in Table 2.2 where $t_1, t_2, \ldots, t_m \in \mathcal{T}$.

We suppose, hereinafter, that the considered constraint system $\mathcal{S}$ contain at least one atom, i.e. $\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A} \neq \varnothing$. Otherwise, we can add one constraint $\{a\} \triangleright a$ to $\mathcal{S}$ which will be satisfied by any substitution. We denote $\{\mathrm{priv}(t) : t \in T\}$ for set of terms T as $\mathrm{priv}(T)$. We define $\mathrm{Vars}(\mathcal{S}) = \bigcup_{i=1}^{n} \mathrm{Vars}(E_i) \cup \mathrm{Vars}(t_i)$. We say that $\mathcal{S}$ is normalized, iff $\forall t \in \mathrm{QSub}(\mathcal{S})$, t is normalized.

Composition rules	Decomposition rules
$t_1, t_2 \to \ulcorner \mathrm{enc}(t_1, t_2) \urcorner$	$\mathrm{enc}(t_1, t_2), \ulcorner t_2 \urcorner \to \ulcorner t_1 \urcorner$
$t_1, t_2 \to \ulcorner \mathrm{aenc}(t_1, t_2) \urcorner$	$\mathrm{aenc}(t_1, t_2), \ulcorner \mathrm{priv}(t_2) \urcorner \to \ulcorner t_1 \urcorner$
$t_1, t_2 \to \ulcorner \mathrm{pair}(t_1, t_2) \urcorner$	$\mathrm{pair}(t_1, t_2) \to \ulcorner t_1 \urcorner$
$t_1, \mathrm{priv}(t_2) \to \ulcorner \mathrm{sig}(t_1, \mathrm{priv}(t_2)) \urcorner$	$\mathrm{pair}(t_1, t_2) \to \ulcorner t_2 \urcorner$
$t_1, \ldots, t_m \to \ulcorner \cdot(t_1, \ldots, t_m) \urcorner$	$\cdot(t_1, \ldots, t_m) \to \ulcorner t_i \urcorner$ for all i
$t_1, t_2 \to \ulcorner \mathrm{apply}(t_1, t_2) \urcorner$	

Table 2.2: DY+ACI deduction system rules

Example 18 We give an instance of general constraint system and its solution within DY+ACI deduction system.

$$\mathcal{S} = \left\{ \begin{array}{lcl} \mathrm{enc}(x, a), \mathrm{pair}(c, a) & \triangleright & b \\ \cdot(\{x, c\}) & \triangleright & a \end{array} \right\},$$

where $a, b, c \in \mathcal{A}$ and $x \in \mathcal{X}$.

One of the eventual models within DY+ACI is $\sigma = \{x \mapsto \mathrm{enc}(\mathrm{pair}(a, b), c)\}$.

Definition 2.1.10 Let $T = \{t_1, \ldots, t_k\}$ be a non-empty set of terms. Then we define $\pi(T)$ as follows:

$$\pi(T) = \ulcorner \cdot(t_1, \ldots, t_k) \urcorner$$

Remark: $\pi(\{t\}) = \ulcorner t \urcorner$.

Definition 2.1.11 We denote a set of non-variable quasi-subterms of a constraint system $\mathcal{S}$ as $\mathring{\mathrm{QSub}}(S, \mathcal{X})$, i.e. $\mathrm{QSub}(\mathcal{S}) \setminus \mathcal{X} = \mathring{\mathrm{QSub}}(S, \mathcal{X})$ or, for shorter notation, $\mathring{\mathrm{QSub}}(S)$.

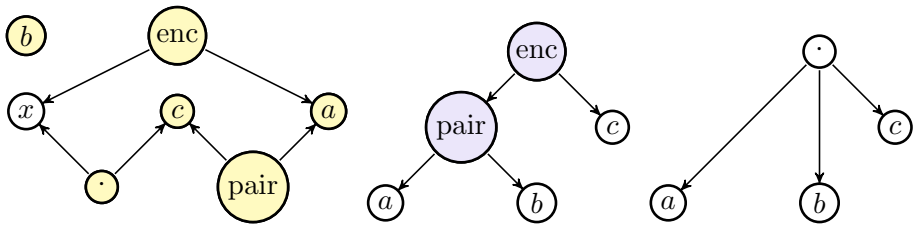
We introduce a transformation $\pi(H^{\mathcal{S},\sigma}(\cdot))$ on ground terms that replaces recursively all binary root symbols such that they are different from all the non-variable quasi-subterms of the constraint system instantiated with its model σ, with ACI symbol $\cdot$. Later we will show that $\pi(\mathrm{H}(\sigma))$ is also a model of $\mathcal{S}$.

Definition 2.1.12 Let us have a constraint system $\mathcal{S}$ which is satisfiable with model σ. Let us fix some $\alpha \in (\mathcal{A} \cap \mathrm{QSub}(\mathcal{S}))$. For given $\mathcal{S}$ and σ we define a function $H^{\mathcal{S},\sigma}(\cdot) : \mathcal{T}_g \to 2^{\mathcal{T}_g}$ as follows:

$$H^{\mathcal{S},\sigma}(t) = \left\{ \begin{array}{rl} \{\alpha\}, & \text{if } t \in (\mathcal{A} \setminus \mathrm{QSub}(\mathcal{S})); \\ \{a\}, & \text{if } t = a \in (\mathcal{A} \cap \mathrm{QSub}(\mathcal{S})); \\ \left\{\mathrm{priv}\left(\pi(H^{\mathcal{S},\sigma}(t_1))\right)\right\}, & \text{if } t = \mathrm{priv}(t_1); \\ \left\{\mathrm{bin}\left(\pi(H^{\mathcal{S},\sigma}(t_1)), \pi(H^{\mathcal{S},\sigma}(t_2))\right)\right\}, & \text{if } t = \mathrm{bin}(t_1, t_2) \\ & \ulcorner t \urcorner \in \ulcorner \mathring{\mathrm{QSub}}(S)\,\sigma \urcorner \\ H^{\mathcal{S},\sigma}(t_1) \cup H^{\mathcal{S},\sigma}(t_2), & \text{if } t = \mathrm{bin}(t_1, t_2) \\ & \wedge\ \ulcorner t \urcorner \notin \ulcorner \mathring{\mathrm{QSub}}(S)\,\sigma \urcorner \\ \bigcup_{p \in L} H^{\mathcal{S},\sigma}(p), & \text{if } t = \cdot(L). \end{array} \right.$$

Henceforward, we will omit parameters and write $H(\cdot)$ instead of $H^{\mathcal{S},\sigma}(\cdot)$ for shorter notation.

Definition 2.1.13 We define the superposition of $\pi(\cdot)$ and $H(\cdot)$ on a set of terms $T = \{t_1, \ldots, t_k\}$ as follows: $\pi(H(T)) = \{\pi(H(t)) \mid t \in T\}$.



(a) $\mathrm{QSub}(\mathcal{S})$ as a DAG; $\mathring{\mathrm{QSub}}(\mathcal{S})$ are highlighted

(b) $x\sigma$ as a DAG Subterms that have no pattern in $\mathring{\mathrm{QSub}}(\mathcal{S})$ are highlighted

(c) $\pi(H(x\sigma))$ as a DAG

Figure 2.4: Illustration for Example 19: Working of $H^{\mathcal{S},\sigma}(\cdot)$ function

Definition 2.1.14 Let $\theta = \{x_1 \mapsto t_1, \ldots, x_k \mapsto t_k\}$ be a substitution. We define $\pi(H(\theta))$ the substitution $\{x_1 \mapsto \pi(H(t_1)), \ldots, x_k \mapsto \pi(H(t_k))\}$.

Note that since for every $x \in \mathrm{dom}(\theta)$ we have $x \notin \mathrm{Vars}(x\theta)$, we also have $\mathrm{dom}(\pi(H(\theta))) = \mathrm{dom}(\theta)$.

Example 19 Let us consider $\mathcal{S}$ and its model σ from Example 18 and show that $\pi(H(\sigma))$ is also a model of $\mathcal{S}$. $\pi(H(\mathrm{enc}(\mathrm{pair}(a,b),c))) = \pi(H(\mathrm{pair}(a,b)) \cup \{c\}) = \pi(\{a\} \cup \{b\} \cup \{c\}) = \cdot(\{a,b,c\})$ (we suppose that $a \prec b \prec c$). One can see that $\pi(\mathrm{H}(\sigma)) = \{x \mapsto \cdot(\{a,b,c\})\}$ is also a model of $\mathcal{S}$ within DY+ACI.

In Figure 2.4 one may see how mighty $H(\cdot)$ function breaks everything that does not match any pattern from the constraint system. This is done because for the solution all these constructions are not principal and may be replaced by a "set" (an ACI set) of its sub-terms which have some pre-image in non-variable quasi-subterms of the constraint system. As was mentioned, the rest of this chapter is devoted to prove this property.

General properties used in proof

The two following lemmas state simple properties of derivability.

Lemma 2. *Let $A, B, C \subseteq \mathcal{T}_g$. Then if $A \subseteq \mathrm{Der}(B)$ and $B \subseteq \mathrm{Der}(C)$ then $A \subseteq \mathrm{Der}(C)$.*

Lemma 3. *Let $A, B, C, D \subseteq \mathcal{T}_g$. Then if $A \subseteq \mathrm{Der}(B)$ and $C \subseteq \mathrm{Der}(D)$ then $A \cup C \subseteq \mathrm{Der}(B \cup D)$.*

In Lemma 4 we list some auxiliary properties that will be used in main proof.

Lemma 4. *The following statements are true:*

1. $\ulcorner\cdot(t,t)\urcorner = \ulcorner t\urcorner$, $\ulcorner\cdot(t_1,t_2)\urcorner = \ulcorner\cdot(t_2,t_1)\urcorner$, $\ulcorner\cdot(\cdot(t_1,t_2),t_3)\urcorner = \ulcorner\cdot(t_1,\cdot(t_2,t_3))\urcorner = \ulcorner\cdot(t_1,t_2,t_3)\urcorner$
2. *if t and $t\sigma$ are terms, then* $\ulcorner t\sigma\urcorner = \ulcorner\ulcorner t\sigma\urcorner\urcorner = \ulcorner\ulcorner t\urcorner\sigma\urcorner = \ulcorner t\ulcorner\sigma\urcorner\urcorner = \ulcorner\ulcorner t\urcorner\ulcorner\sigma\urcorner\urcorner$
3. $s \in \mathrm{QSub}(\ulcorner t\urcorner) \implies s = \ulcorner s\urcorner$
4. $\forall s \in \mathrm{Sub}(\ulcorner t\urcorner)\ \exists s' \in \mathrm{Sub}(t) : s = \ulcorner s'\urcorner$
5. $\ulcorner\mathrm{elems}(t)\urcorner = \mathrm{elems}(\ulcorner t\urcorner)$

6. $\ulcorner\cdot(\ulcorner t_1\urcorner,\ldots,\ulcorner t_m\urcorner)\urcorner = \ulcorner\cdot(t_1,\ldots,t_m)\urcorner$; $\pi(T) = \pi(\ulcorner T\urcorner)$
7. $\mathrm{elems}(\ulcorner\cdot(\ulcorner t_1\urcorner,\ldots,\ulcorner t_m\urcorner)\urcorner) = \mathrm{elems}(\cdot(\ulcorner t_1\urcorner,\ldots,\ulcorner t_m\urcorner)) = \bigcup_{i=1,\ldots,m}\mathrm{elems}(\ulcorner t_i\urcorner)$
8. $H(t) = \bigcup_{p\in\mathrm{elems}(t)} H(p)$,
9. $H(t) = H(\ulcorner t\urcorner)$
10. $\pi(H(t)) = \pi(H(\ulcorner t\urcorner)) = \ulcorner\pi(H(t))\urcorner = \ulcorner\pi(H(\ulcorner t\urcorner))\urcorner$
11. $\pi(T_1\cup\cdots\cup T_m) = \pi(\{\pi(T_1),\ldots,\pi(T_m)\})$
12. $\mathrm{QSub}(\mathrm{QSub}(t)) = \mathrm{QSub}(t)$
13. $\mathrm{QSub}(\ulcorner t\urcorner) \subseteq \ulcorner\mathrm{QSub}(t)\urcorner$
14. $\mathrm{QSub}(t\sigma) \subseteq \mathrm{QSub}(t)\,\sigma \cup \mathrm{QSub}(\mathrm{Vars}(t)\,\sigma)$
15. $\mathrm{Sub}(t\sigma) = \mathrm{Sub}(t)\,\sigma \cup \mathrm{Sub}(\mathrm{Vars}(t)\,\sigma)$
16. $|\ulcorner T\urcorner| \le |T|$, $|T\sigma| \le |T|$
17. $\mathrm{elems}(t) \subseteq \mathrm{QSub}(t) \subseteq \mathrm{Sub}(t)$
18. *For term t,* $\mathrm{size}(\ulcorner t\urcorner) \le \mathrm{size}(t)$*;*
    for set of terms T, $\mathrm{size}(\ulcorner T\urcorner) \le \mathrm{size}(T)$*;*
    for constraint system $\mathcal{S}$, $\mathrm{size}(\ulcorner\mathcal{S}\urcorner) \le \mathrm{size}(\mathcal{S})$
19. $\mathrm{QSub}(\cdot(\{t_1,\ldots,t_l\})) \subseteq \{\cdot(\{t_1,\ldots,t_l\})\} \cup \mathrm{QSub}(t_1)\cdots\cup\mathrm{QSub}(t_l)$
20. $\forall s \in \mathrm{Sub}(t)\ \ \mathrm{size}(\ulcorner t\sigma\urcorner) \ge \mathrm{size}(\ulcorner s\sigma\urcorner)$.

Proof. We will give proofs of several statements. To skip it, go to p. 39.

Statement 1: Follows from the definition of the normalization function and Definition 2.1.3.

Statement 2: Here we will prove only $\ulcorner t\sigma\urcorner = \ulcorner\ulcorner t\sigma\urcorner\urcorner$ by induction on $\mathrm{size}(\ulcorner t\sigma\urcorner)$.

- If $\mathrm{size}(\ulcorner t\sigma\urcorner) = 1$, then $\ulcorner t\sigma\urcorner \in \mathcal{X}\cup\mathcal{A}$ and the statement is trivial.
- Suppose that for some k, for any $\ulcorner t\sigma\urcorner$, such that $\mathrm{size}(\ulcorner t\sigma\urcorner) < k$ $(k > 1)$, $\ulcorner t\sigma\urcorner = \ulcorner\ulcorner t\sigma\urcorner\urcorner$.
- Consider $\ulcorner t\sigma\urcorner$ such that $\mathrm{size}(\ulcorner t\sigma\urcorner) = k$. By the definition of normal form $\ulcorner t\sigma\urcorner$ may be:
  - $t\sigma$, if $t\sigma \in \mathcal{X}\cup\mathcal{A}$. In this case $\ulcorner\ulcorner t\sigma\urcorner\urcorner = \ulcorner t\sigma\urcorner$.
  - $\mathrm{bin}(\ulcorner t_1\urcorner,\ulcorner t_2\urcorner)$. then $\ulcorner\mathrm{bin}(\ulcorner t_1\urcorner,\ulcorner t_2\urcorner)\urcorner = \mathrm{bin}(\ulcorner\ulcorner t_1\urcorner\urcorner,\ulcorner\ulcorner t_2\urcorner\urcorner) =$ (by induction supposition, since $\mathrm{size}(\ulcorner t_i\urcorner) < k$ for $i = 1,2$) $= \mathrm{bin}(\ulcorner t_1\urcorner,\ulcorner t_2\urcorner) = \ulcorner t\sigma\urcorner$.
  - $\mathrm{priv}(\ulcorner t\urcorner)$. Can be proved by analogy with the previous case.
  - t', if $\ulcorner\mathrm{elems}(t\sigma)\urcorner = \{t'\}$. Then t' is normalized, therefore $\ulcorner t'\urcorner = t'$. Thus $\ulcorner\ulcorner t\sigma\urcorner\urcorner = \ulcorner t\sigma\urcorner$.
  - $\cdot(L')$, if $|\ulcorner\mathrm{elems}(t\sigma)\urcorner| > 1$ and $L' \approx \ulcorner\mathrm{elems}(t\sigma)\urcorner$ and for all $i < j$, $L'[i] < L'[j]$. By definition of elems, the list L' does not contain terms with ACI root symbol. Moreover, all elements of L' are normalized. This means that $\ulcorner\mathrm{elems}(L')\urcorner \approx L'$. Finally, since all elements in list L' are non-repeating and sorted, we can conclude that $\ulcorner\cdot(L')\urcorner = \cdot(L')$ (note also that number of elements in L' is greater than 1). Thus, $\cdot(L') = \ulcorner\cdot(L')\urcorner$ and $\ulcorner\ulcorner t\sigma\urcorner\urcorner = \ulcorner t\sigma\urcorner$.

In [BN98] it is stated that the smallest equivalence relation (in our case $\equiv$) generated by an equational theory (in our case ACI) specified as a set of identities is *closed under substitution*, that is, $p \equiv q$ implies $p\sigma \equiv q\sigma$. Thus using Lemma 1 and $\ulcorner\ulcorner t\urcorner\urcorner = t$ we obtain $t \equiv \ulcorner t\urcorner$ and $t\sigma \equiv \ulcorner t\urcorner\sigma$, and then $\ulcorner t\sigma\urcorner = \ulcorner\ulcorner t\urcorner\sigma\urcorner$.

Note that $\ulcorner t\sigma\urcorner = \ulcorner\ulcorner t\urcorner\ulcorner\sigma\urcorner\urcorner$ follows from $\ulcorner t\sigma\urcorner = \ulcorner\ulcorner t\urcorner\sigma\urcorner$ and $\ulcorner t\sigma\urcorner = \ulcorner t\ulcorner\sigma\urcorner\urcorner$.

Statement 3: By induction on size(s). Let us fix t.

- size(s) = size$(\ulcorner t\urcorner)$. Then $s = \ulcorner t\urcorner$ and $\ulcorner s\urcorner = \ulcorner\ulcorner t\urcorner\urcorner$, and from Statement 2 (by taking empty σ) we have $\ulcorner t\urcorner = \ulcorner\ulcorner t\urcorner\urcorner$, and thus $\ulcorner s\urcorner = s$.
- Suppose that for some k, for any $s \in \mathrm{QSub}(\ulcorner t\urcorner)$, such that size$(s) > k$, $s = \ulcorner s\urcorner$.
- Consider case, where $s \in \mathrm{QSub}(\ulcorner t\urcorner)$ and size$(s) = k$. Then, by definition of QSub$(\cdot)$, s is in
  - priv$(s) \in \mathrm{QSub}(\ulcorner t\urcorner)$. By induction supposition we have $\ulcorner \mathrm{priv}(s)\urcorner = \mathrm{priv}(s)$, and as $\ulcorner \mathrm{priv}(s)\urcorner = \mathrm{priv}(\ulcorner s\urcorner)$, we have $s = \ulcorner s\urcorner$.
  - bin$(s,p) \in \mathrm{QSub}(\ulcorner t\urcorner)$. By induction we have $\ulcorner \mathrm{bin}(s,p)\urcorner = \mathrm{bin}(s,p)$, and as $\ulcorner \mathrm{bin}(s,p)\urcorner = \mathrm{bin}(\ulcorner s\urcorner, \ulcorner p\urcorner)$, we have $s = \ulcorner s\urcorner$.
  - bin$(p,s) \in \mathrm{QSub}(\ulcorner t\urcorner)$. The similar case.
  - $s \in \mathrm{elems}(\cdot(L)), \cdot(L) \in \mathrm{QSub}(\ulcorner t\urcorner)$. As size$(\cdot(L)) > k$, we have $\cdot(L) = \ulcorner\cdot(L)\urcorner$, that means (from Definition 2.1.6) that L is a list of normalized non-ACI-set terms, and as elems$(L) \approx L$, we have that s is normalized.

Statement 4: Suppose the opposite and let us take $s \in \mathrm{Sub}(\ulcorner t\urcorner)$ with maximal size(s) that does not satisfy the desired property. Note that the "biggest" term in Sub$(\ulcorner t\urcorner)$, i.e. $\ulcorner t\urcorner$, does satisfy the property, as we can choose $s' = t \in \mathrm{Sub}(t)$. By definition of Sub$(\cdot)$ if $s \in \mathrm{Sub}(\ulcorner t\urcorner)$ and $s \neq \ulcorner t\urcorner$ then $\exists r \in \mathrm{Sub}(\ulcorner t\urcorner)$ such that

- $r = \mathrm{bin}(p,s)$ or $r = \mathrm{bin}(s,p)$ or $r = \mathrm{priv}(s)$. Without loss of generality we consider only the first case ($r = \mathrm{bin}(p,s)$) as other ones are similar. As size$(r) >$ size(s), there exists $r' \in \mathrm{Sub}(t)$ such that $r = \ulcorner r'\urcorner$. By definition of $\ulcorner\cdot\urcorner$:
  - either $r' = \mathrm{bin}(p',s')$ and $\ulcorner p'\urcorner = p$ and $\ulcorner s'\urcorner = s$. As $s' \in \mathrm{Sub}(r') \subseteq \mathrm{Sub}(t)$ the property is proved.
  - or $r' = \cdot(L)$ and $\ulcorner \mathrm{elems}(L)\urcorner = \{r\}$. Since $\forall q \in \mathrm{elems}(L)$, root$(q) \neq \cdot$, then $\exists q \in \mathrm{elems}(L) : q = \mathrm{bin}(p',s')$ and $\ulcorner p'\urcorner = p$ and $\ulcorner s'\urcorner = s$. Using Statement 17 we have $s' \in \mathrm{Sub}(t)$.
- $r = \cdot(L)$ and $s \in L$. Then (since size$(r) >$ size(s)) we have $\exists r' \in \mathrm{Sub}(t) : \ulcorner r'\urcorner = r$. Using Lemma 6 and Statements 3 we obtain r — normalized, and thus, root$(s) \neq \cdot$. Then by definition of $\ulcorner\cdot\urcorner$ we have $r' = \cdot(L')$ and $L \approx \ulcorner \mathrm{elems}(L')\urcorner$, and thus, $s \in \ulcorner \mathrm{elems}(L')\urcorner$, that is $\exists s' \in \mathrm{elems}(L') : s = \ulcorner s'\urcorner$. Using again Statement 17 we have $s' \in \mathrm{Sub}(t)$.

Statement 5: This statement is trivial, if $t \neq \cdot(L)$. Otherwise, let $t = \cdot(t_1, \ldots, t_n)$.

- if $\ulcorner \mathrm{elems}(t)\urcorner = \{p\}$, where $p \neq \cdot(L_p)$. Then $\ulcorner t\urcorner = p$ and then elems$(\ulcorner t\urcorner) = \mathrm{elems}(p) = \{p\} = \ulcorner \mathrm{elems}(t)\urcorner$.
- if $\ulcorner \mathrm{elems}(t)\urcorner = \{p_1, \ldots, p_k\}$, $k > 1$, where $p_i \neq \cdot(L_i)$ for all i. Then $\ulcorner t\urcorner = \cdot(L)$, where $L \approx \{p_1, \ldots, p_k\}$. That means that elems$(\ulcorner t\urcorner) = \bigcup_{p \in \{p_1,\ldots,p_k\}} \mathrm{elems}(p) = \{p_1, \ldots, p_k\}$.

Statement 6: The first part follows from the definition of normal form and Statement 5. The second one directly follows from the first.

Statement 7: We get the first part of equality if apply Statement 5: $\mathrm{elems}\,(\ulcorner\cdot(\ulcorner t_1\urcorner,\ldots,\ulcorner t_m\urcorner)\urcorner) = \ulcorner\mathrm{elems}\,(\cdot(\ulcorner t_1\urcorner,\ldots,\ulcorner t_m\urcorner))\urcorner$ and then from Definition 2.1.3 and Statement 5 we have that $\mathrm{elems}\,(\cdot(\ulcorner t_1\urcorner,\ldots,\ulcorner t_m\urcorner))$ is a set of normalized terms. The second part directly follows from Definition 2.1.3.

Statement 8: By induction on $\mathrm{size}\,(t)$.

- $\mathrm{size}\,(t) = 1$, implies $t = a \in \mathcal{A}$ and then $\mathrm{elems}\,(a) = \{a\}$, i.e. the equality becomes trivial.
- Suppose that for any $t : \mathrm{size}\,(t) < k$ $(k > 1)$, $H\,(t) = \bigcup_{p\in\mathrm{elems}(t)} H\,(p)$ holds.
- Given a term $t : \mathrm{size}\,(t) = k$, $k > 1$. We should prove $H\,(t) = \bigcup_{p\in\mathrm{elems}(t)} H\,(p)$.
  - $t = \mathrm{priv}\,(t_1)$ or $t = \mathrm{bin}\,(p, q)$. In both cases, $\mathrm{elems}\,(t) = \{t\}$, and thus, he equality is trivial.
  - $t = \cdot(L)$. Note that $\forall s \in L$, $\mathrm{size}\,(s) < k$. Then, on one hand, $H\,(\cdot(L)) = \bigcup_{p\in L} H\,(p) =$ (by induction supposition) $= \bigcup_{p\in L}\bigcup_{p'\in\mathrm{elems}(p)} H\,(p')$. On the other hand, $\bigcup_{p\in\mathrm{elems}(\cdot(L))} H\,(p) = \bigcup_{p\in\bigcup_{p'\in L}\mathrm{elems}(p')} H\,(p) = \bigcup_{p'\in L}\bigcup_{p\in\mathrm{elems}(p')} H\,(p)$. Thus, $H\,(t) = \bigcup_{p\in\mathrm{elems}(t)} H\,(p)$.

Statement 9: By induction on $\mathrm{size}\,(t)$:

- $\mathrm{size}\,(t) = 1$ is possible in the only case: $t = a \in \mathcal{A}$ and as $a = \ulcorner a\urcorner$, the equality is trivial.
- Suppose that for any $t : \mathrm{size}\,(t) < k$ $(k > 1)$, $H\,(t) = H\,(\ulcorner t\urcorner)$ holds.
- Given a term $t : \mathrm{size}\,(t) = k$, $k > 1$. We need to prove that $H\,(t) = H\,(\ulcorner t\urcorner)$.
  - if $t = \mathrm{priv}\,(t_1)$, then $H\,(t) = \{\mathrm{priv}\,(\pi(H\,(t_1)))\} =$ (by induction supposition) $= \{\mathrm{priv}\,(\pi(H\,(\ulcorner t_1\urcorner)))\} = H\,(\mathrm{priv}\,(\ulcorner t_1\urcorner)) = H\,(\ulcorner t\urcorner)$.
  - if $t = \mathrm{bin}\,(p, q)$ and $\ulcorner t\urcorner \in \ulcorner\mathrm{QSub}\,(\mathcal{S})\,\sigma\urcorner$. Then $H\,(\ulcorner t\urcorner) = H\,(\mathrm{bin}\,(\ulcorner p\urcorner,\ulcorner q\urcorner)) = \{\mathrm{bin}\,(\pi(H\,(\ulcorner p\urcorner)),\pi(H\,(\ulcorner q\urcorner)))\} =$ (by induction) $= \{\mathrm{bin}\,(\pi(\ulcorner H\,(p)\urcorner),\pi(\ulcorner H\,(q)\urcorner))\} =$ (by Statement 6) $= \{\mathrm{bin}\,(\pi(H\,(p)),\pi(H\,(q)))\} = H\,(\mathrm{bin}\,(p, q))$.
  - if $t = \mathrm{bin}\,(p, q)$ and $\ulcorner t\urcorner \notin \ulcorner\mathrm{QSub}\,(\mathcal{S})\,\sigma\urcorner$. Then $H\,(t) = H\,(p) \cup H\,(q) =$ (by induction supposition) $= H\,(\ulcorner p\urcorner) \cup H\,(\ulcorner q\urcorner) =$ (as $\ulcorner\mathrm{bin}\,(\ulcorner p\urcorner,\ulcorner q\urcorner)\urcorner = \ulcorner t\urcorner \notin \ulcorner\mathrm{QSub}\,(\mathcal{S})\,\sigma\urcorner$) $= H\,(\mathrm{bin}\,(\ulcorner p\urcorner,\ulcorner q\urcorner)) = H\,(\ulcorner t\urcorner)$
  - if $t = \cdot(L)$, where $L = \{t_1,\ldots,t_m\}$. Note first that as $t = \cdot(L)$, $\forall s \in \mathrm{elems}\,(t)$, $\mathrm{size}\,(s) < \mathrm{size}\,(t)$. Then, by Statement 8, $H\,(t) = \bigcup_{p\in\mathrm{elems}(t)} H\,(p) =$ (by induction supposition) $= \bigcup_{p\in\mathrm{elems}(t)} H\,(\ulcorner p\urcorner)$. On the other part, $H\,(\ulcorner t\urcorner) = \bigcup_{p\in\mathrm{elems}(\ulcorner t\urcorner)} H\,(p) =$ (by Statement 5) $= \bigcup_{p\in\ulcorner\mathrm{elems}(t)\urcorner} H\,(p) = \bigcup_{p\in\mathrm{elems}(t)} H\,(\ulcorner p\urcorner) = H\,(t)$.

Statement 10: This follows from Statements 9, 6, Definition 2.1.10 and from equality $\ulcorner\ulcorner t\urcorner\urcorner = \ulcorner t\urcorner$ (Statement 2).

Statement 11: From definition of π and Statement 5, we have $\mathrm{elems}\,(\pi(T_i)) = \ulcorner\mathrm{elems}\,(T_i)\urcorner$. Next $\pi(\{\pi(T_1),\ldots,\pi(T_m)\}) = \ulcorner\cdot(L)\urcorner$ (here we use $\ulcorner\cdot(L)\urcorner$ to capture two cases from definition of normalization at once), where $L \approx \ulcorner\mathrm{elems}\,(\{\pi(T_1),\ldots,\pi(T_m)\})\urcorner$ $= \ulcorner\bigcup_{i=1,\ldots,m}\ulcorner\mathrm{elems}\,(T_i)\urcorner\urcorner = \ulcorner\bigcup_{i=1,\ldots,m}\mathrm{elems}\,(T_i)\urcorner$, while $\pi(T_1 \cup\cdots\cup T_m) = \ulcorner\cdot(L')\urcorner$, where $L' \approx \ulcorner\bigcup_{i=1,\ldots,m}\mathrm{elems}\,(T_i)\urcorner$.

Statement 12: $\mathrm{QSub}(t) \subseteq \mathrm{QSub}(\mathrm{QSub}(t))$ is trivial as $t \in \mathrm{QSub}(t)$. Now we prove by induction on $\mathrm{size}(t)$ that $\mathrm{QSub}(\mathrm{QSub}(t)) \subseteq \mathrm{QSub}(t)$

- $\mathrm{size}(t) = 1$. Then $t \in \mathcal{A} \cup \mathcal{X}$. As $\mathrm{QSub}(t) = \{t\}$ the statement is trivial.
- Suppose that for any $t : \mathrm{size}(t) < k$ $(k \geq 1)$, the statement is true.
- Given a term $t : \mathrm{size}(t) = k$, $k > 1$. Let us consider all possible cases:
  - $t = \mathrm{bin}(t_1, t_2)$.By definition $\mathrm{QSub}(t) = \{t\} \cup \mathrm{QSub}(t_1) \cup \mathrm{QSub}(t_2)$. Then, $\mathrm{QSub}(\mathrm{QSub}(t)) = \mathrm{QSub}(t) \cup \mathrm{QSub}(\mathrm{QSub}(t_1)) \cup \mathrm{QSub}(\mathrm{QSub}(t_2))$ and, as $\mathrm{size}(t_i) < k$ for $i = 1, 2$, by using induction supposition we obtain the wanted property.
  - $t = \mathrm{priv}(t_1)$. Proof is similar to one for the case above.
  - $t = \cdot(L)$. We have $\mathrm{QSub}(t) = \{t\} \cup \bigcup_{p \in \mathrm{elems}(L)} \mathrm{QSub}(p)$. Then $\mathrm{QSub}(\mathrm{QSub}(t)) = \mathrm{QSub}(t) \cup \bigcup_{p \in \mathrm{elems}(L)} \mathrm{QSub}(\mathrm{QSub}(p))$, but as $\mathrm{size}(p) < k$ for every such p we can apply the induction supposition and get $\bigcup_{p \in \mathrm{elems}(L)} \mathrm{QSub}(\mathrm{QSub}(p)) = \bigcup_{p \in \mathrm{elems}(L)} \mathrm{QSub}(p) = \mathrm{QSub}(t) \setminus \{t\}$. Then $\mathrm{QSub}(\mathrm{QSub}(t)) = \mathrm{QSub}(t) \cup (\mathrm{QSub}(t) \setminus \{t\}) = \mathrm{QSub}(t)$.

Statement 13: By induction on $\mathrm{size}(t)$.

- $\mathrm{size}(t) = 1$. Then $t \in \mathcal{A} \cup \mathcal{X}$. As $\mathrm{QSub}(t) = \{t\}$ and $t = \ulcorner t \urcorner$, the statement holds.
- Suppose that for any $t : \mathrm{size}(t) < k$ $(k > 1)$, the statement is true.
- Given a term $t : \mathrm{size}(t) = k$, $k \geq 1$. Let us consider all possible cases:
  - $t = \mathrm{bin}(t_1, t_2)$.On the one hand, $\mathrm{QSub}(t) = \{t\} \cup \mathrm{QSub}(t_1) \cup \mathrm{QSub}(t_2)$. On the other hand, $\ulcorner t \urcorner = \mathrm{bin}(\ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner)$ and then, $\mathrm{QSub}(\ulcorner t \urcorner) = \{\ulcorner t \urcorner\} \cup \mathrm{QSub}(\ulcorner t_1 \urcorner) \cup \mathrm{QSub}(\ulcorner t_2 \urcorner)$. Then, as $\mathrm{QSub}(\ulcorner t_i \urcorner) \subseteq \ulcorner \mathrm{QSub}(t_i) \urcorner$, we have that $\mathrm{QSub}(\ulcorner t \urcorner) \subseteq \ulcorner \mathrm{QSub}(t) \urcorner$.
  - $t = \mathrm{priv}(t_1)$. Proof is similar to one for the case above.
  - $t = \cdot(L)$. We have $\mathrm{QSub}(t) = \{t\} \cup \bigcup_{p \in \mathrm{elems}(L)} \mathrm{QSub}(p)$. From Statement 5 we have $\mathrm{elems}(\ulcorner \cdot(L) \urcorner) = \ulcorner \mathrm{elems}(\cdot(L)) \urcorner$, and then, $\mathrm{QSub}(\ulcorner \cdot(L) \urcorner) = \{\ulcorner \cdot(L) \urcorner\} \cup \bigcup_{p \in \mathrm{elems}(\ulcorner \cdot(L) \urcorner)} \mathrm{QSub}(p) =$
    $\ulcorner \{\cdot(L)\} \urcorner \cup \bigcup_{p \in \mathrm{elems}(\cdot(L))} \mathrm{QSub}(\ulcorner p \urcorner) \subseteq$ (by supposition)
    $\subseteq \ulcorner \{\cdot(L)\} \urcorner \cup \bigcup_{p \in \mathrm{elems}(\cdot(L))} \ulcorner \mathrm{QSub}(p) \urcorner = \ulcorner \{\cdot(L)\} \cup \bigcup_{p \in \mathrm{elems}(\cdot(L))} \mathrm{QSub}(p) \urcorner = \ulcorner \mathrm{QSub}(t) \urcorner$.

Statement 14: By induction on $\mathrm{size}(t)$

- $\mathrm{size}(t) = 1$.
  - $t \in \mathcal{A}$. As $t\sigma = t$ and $\mathrm{Vars}(t) = \varnothing$, the statement becomes trivial.
  - $t \in \mathcal{X}$. Then $\mathrm{QSub}(t)\sigma = t\sigma$, $\mathrm{Vars}(t) = \{t\}$; We have $\mathrm{QSub}(t\sigma) \subseteq \{t\sigma\} \cup \mathrm{QSub}(t\sigma)$.
- Suppose that for any $t : \mathrm{size}(t) < k$ $(k \geq 1)$, the statement is true.
- Given a term $t : \mathrm{size}(t) = k$, $k > 1$. Let us consider all possible cases:
  - $t = \mathrm{bin}(t_1, t_2)$.Then $t\sigma = \mathrm{bin}(t_1\sigma, t_2\sigma)$ and $\mathrm{Vars}(t) = \mathrm{Vars}(t_1) \cup \mathrm{Vars}(t_2)$. $\mathrm{QSub}(t\sigma) = \{t\sigma\} \cup \mathrm{QSub}(t_1\sigma) \cup \mathrm{QSub}(t_2\sigma) \subseteq$ (as $\mathrm{size}(t_i) < k$) $\subseteq \{t\sigma\} \cup \mathrm{QSub}(t_1)\,\sigma \cup \mathrm{QSub}(\mathrm{Vars}(t_1)\,\sigma) \cup \mathrm{QSub}(t_2)\,\sigma \cup \mathrm{QSub}(\mathrm{Vars}(t_2)\,\sigma) = \{t\sigma\} \cup \mathrm{QSub}(t_1)\,\sigma \cup \mathrm{QSub}(t_2)\,\sigma \cup \mathrm{QSub}((\mathrm{Vars}(t_1) \cup \mathrm{Vars}(t_2$
    $\mathrm{QSub}(t)\,\sigma \cup \mathrm{QSub}(\mathrm{Vars}(t)\,\sigma)$.
  - $t = \mathrm{priv}(t_1)$. Proof is similar to one for the case above.

– $t = \cdot(\{t_1, \ldots, t_m\})$. Then $t\sigma = \cdot(\{t_1\sigma, \ldots, t_m\sigma\})$ and $\mathrm{Vars}(t) = \bigcup_{i=1,\ldots,m} \mathrm{Vars}(t_i)$. Then we have $\mathrm{QSub}(t\sigma) = \{t\sigma\} \cup \bigcup_{p \in \mathrm{elems}(\{t_1\sigma,\ldots,t_m\sigma\})} \mathrm{QSub}(p) \subseteq$ (using Statement 17) $\subseteq \{t\sigma\} \cup \bigcup_{p \in \bigcup_{i=1}^{m} \mathrm{QSub}(t_i\sigma)} \mathrm{QSub}(p) =$ (as $\mathrm{QSub}(\mathrm{QSub}(p)) = \mathrm{QSub}(p)$) $= \{t\sigma\} \cup \bigcup_{i=1,\ldots,m} \mathrm{QSub}(t_i\sigma) \subseteq$ (as $\mathrm{size}(t_i) < k$) $\subseteq \{t\sigma\} \cup$ $\cup \bigcup_{i=1,\ldots,m} (\mathrm{QSub}(t_i)\,\sigma \cup \mathrm{QSub}(\mathrm{Vars}(t_i)\,\sigma))$ $= \{t\sigma\} \cup \bigcup_{i=1,\ldots,m} \mathrm{QSub}(t_i)\,\sigma \cup \mathrm{QSub}\left(\left(\bigcup_{i=1,\ldots,m} \mathrm{Vars}(t_i)\right)\sigma\right)$ $= \mathrm{QSub}(t)\,\sigma \cup \mathrm{Sub}(\mathrm{Vars}(t)\,\sigma)$.

Statement 15: By induction on $\mathrm{size}(t)$

- $\mathrm{size}(t) = 1$.
  - $t \in \mathcal{A}$. As $t\sigma = t$ and $\mathrm{Vars}(t) = \varnothing$, the statement becomes trivial.
  - $t \in \mathcal{X}$. Then $\mathrm{Sub}(t)\,\sigma = t\sigma$, $\mathrm{Vars}(t) = \{t\}$; and as for any term p, $p \in \mathrm{Sub}(p)$, we have $\mathrm{Sub}(t\sigma) = \{t\sigma\} \cup \mathrm{Sub}(t\sigma)$.
- Suppose that for any $t : \mathrm{size}(t) < k$ $(k \geq 1)$, the statement is true.
- Given a term $t : \mathrm{size}(t) = k$, $k > 1$. Let us consider all possible cases:
  - $t = \mathrm{bin}(t_1, t_2)$.Then $t\sigma = \mathrm{bin}(t_1\sigma, t_2\sigma)$ and $\mathrm{Vars}(t) = \mathrm{Vars}(t_1) \cup \mathrm{Vars}(t_2)$. $\mathrm{Sub}(t\sigma) = \{t\sigma\} \cup \mathrm{Sub}(t_1\sigma) \cup \mathrm{Sub}(t_2\sigma) =$ (as $\mathrm{size}(t_i) < k$) $= \{t\sigma\} \cup \mathrm{Sub}(t_1)\,\sigma \cup \mathrm{Sub}(\mathrm{Vars}(t_1)\,\sigma) \cup$ $\mathrm{Sub}(t_2)\,\sigma \cup \mathrm{Sub}(\mathrm{Vars}(t_2)\,\sigma) = \{t\sigma\} \cup \mathrm{Sub}(t_1)\,\sigma \cup \mathrm{Sub}(t_2)\,\sigma \cup \mathrm{Sub}((\mathrm{Vars}(t_1) \cup \mathrm{Vars}(t_2))\sigma) =$ $\mathrm{Sub}(t)\,\sigma \cup \mathrm{Sub}(\mathrm{Vars}(t)\,\sigma)$.
  - $t = \mathrm{priv}(t_1)$. Proof is similar to one for the case above.
  - $t = \cdot(\{t_1, \ldots, t_m\})$. Then $t\sigma = \cdot(\{t_1\sigma, \ldots, t_m\sigma\})$ and $\mathrm{Vars}(t) = \bigcup_{i=1,\ldots,m} \mathrm{Vars}(t_i)$. Then we have $\mathrm{Sub}(t\sigma) = \{t\sigma\} \cup \bigcup_{i=1,\ldots,m} \mathrm{Sub}(t_i\sigma) =$ (as $\mathrm{size}(t_i) < k$) $= \{t\sigma\} \cup$ $\bigcup_{i=1,\ldots,m} (\mathrm{Sub}(t_i)\,\sigma \cup \mathrm{Sub}(\mathrm{Vars}(t_i)\,\sigma)) = \{t\sigma\} \cup \bigcup_{i=1,\ldots,m} \mathrm{Sub}(t_i)\,\sigma \cup$ $\cup\, \mathrm{Sub}\left(\left(\bigcup_{i=1,\ldots,m} \mathrm{Vars}(t_i)\right)\sigma\right) = \mathrm{Sub}(t)\,\sigma \cup \mathrm{Sub}(\mathrm{Vars}(t)\,\sigma)$.

Statement 16: It follows from the fact that $f(t) = \ulcorner t \urcorner$ and $g_\sigma(t) = t\sigma$ are deterministic functions, and thus return at most one value for one given argument.

Statement 17: First we prove that $\mathrm{elems}(t) \subseteq \mathrm{QSub}(t)$. We use induction on $\mathrm{size}(t)$.

- If $\mathrm{root}(t) \neq \cdot$, then $\mathrm{elems}(t) = \{t\} \subseteq \mathrm{QSub}(t)$. This case includes all t such that $\mathrm{size}(t) = 1$. Thus we need to consider only $t = \cdot(L)$.
- Suppose that for any $t : \mathrm{size}(t) < k$ $(k \geq 1)$, the statement holds.
- If for some t we have $\mathrm{size}(t) = k$, $k > 1$, then $\mathrm{elems}(t) = \bigcup_{p \in L} \mathrm{elems}(p)$ and $\mathrm{QSub}(t) = \{t\} \bigcup_{p \in L} \mathrm{QSub}(p)$. And since $\mathrm{size}(p) < k$ using the induction supposition we obtain the wanted statement.

Now we show that $\mathrm{QSub}(t) \subseteq \mathrm{Sub}(t)$. Again, applying proof by induction on on $\mathrm{size}(t)$ we have:

- If $\mathrm{size}(t) = 1$, then $\mathrm{QSub}(t) = \mathrm{Sub}(t) = \{t\}$.
- Suppose that for any $t : \mathrm{size}(t) < k$ $(k \geq 1)$, the statement holds.
- If for some t we have $\mathrm{size}(t) = k$, $k > 1$, then
  - $t = \mathrm{bin}(t_1, t_2)$.Then $\mathrm{QSub}(t) = \{t\} \cup \mathrm{QSub}(t_1) \cup \mathrm{QSub}(t_2)$ and $\mathrm{Sub}(t) = \{t\} \cup \mathrm{Sub}(t_1) \cup \mathrm{Sub}(t_2)$, where $\max\{\mathrm{size}(t_1), \mathrm{size}(t_2)\} < k$. And then using induction supposition we can conclude for this case.

– $t = \mathrm{priv}(t_1)$. Proof is similar to one for the case above.
– $t = \cdot(\{t_1, \dots, t_m\})$. Then we have $\mathrm{QSub}(t) = \{t\} \cup \bigcup_{p \in \mathrm{elems}(\{t_1,\dots,t_m\})} \mathrm{QSub}(p) \subseteq$ (using the already proved part of the property) $\subseteq \{t\} \cup \bigcup_{p \in \mathrm{QSub}(\{t_1,\dots,t_m\})} \mathrm{QSub}(p) =$ (as $\mathrm{QSub}(\mathrm{QSub}(t)) = \mathrm{QSub}(t)$) $= \{t\} \cup \bigcup_{p \in \{t_1,\dots,t_m\}} \mathrm{QSub}(p) \subseteq$ (using induction supposition, as $\forall i\ \mathrm{size}(t_i) < k$) $\subseteq \{t\} \cup \bigcup_{p \in \{t_1,\dots,t_m\}} \mathrm{Sub}(p) = \mathrm{Sub}(t)$. □

Statement 18: Using Statement 4 and the fact that $\ulcorner\cdot\urcorner$ is a deterministic function we obtain $\forall p, q \in \mathrm{Sub}(\ulcorner t \urcorner)\, p \neq q\, \exists p', q' \in \mathrm{Sub}(t) : p = \ulcorner p' \urcorner \wedge q = \ulcorner q' \urcorner \wedge p \neq q$. And thus, $|\mathrm{Sub}(\ulcorner t \urcorner)| \leq |\mathrm{Sub}(t)|$.

Statement 19: We have $\mathrm{QSub}(\cdot(\{t_1, \dots, t_l\})) = \{\cdot(\{t_1, \dots, t_l\})\} \cup \bigcup_{i=1}^{l} \mathrm{QSub}(\mathrm{elems}(t_i))$. Using Statement 17 and 12 we have $\mathrm{QSub}(\mathrm{elems}(t_i)) \subseteq \mathrm{QSub}(\mathrm{QSub}(t_i)) = \mathrm{QSub}(t_i)$. Thus, $\mathrm{QSub}(\cdot(\{t_1, \dots, t_l\})) \subseteq \{\cdot(\{t_1, \dots, t_l\})\} \cup \mathrm{QSub}(t_1) \cdots \cup \mathrm{QSub}(t_l)$.

Statement 20: From Statement 15 we can easily obtain $\mathrm{Sub}(s\sigma) \subseteq \mathrm{Sub}(t\sigma)$. Then we need to prove that prove that $\mathrm{size}(\ulcorner p \urcorner) \leq \mathrm{size}(\ulcorner q \urcorner)$, if $\mathrm{Sub}(p) \subseteq \mathrm{Sub}(q)$. The proof is mainly based on the fact that if $\mathrm{Sub}(p) \subseteq \mathrm{Sub}(q)$ then $\mathrm{Sub}(\ulcorner p \urcorner) \setminus \{\ulcorner p \urcorner\} \subseteq \mathrm{Sub}(\ulcorner q \urcorner)$. Let us consider several cases.

- If $p = q$ then the statement is trivial.
- If there exists $v \in \mathrm{Sub}(q)$ such that $v = \mathrm{bin}(p, p')$ or $v = \mathrm{bin}(p', p)$ or $v = \mathrm{priv}(p)$. Then note that by definition of normalization, since $\mathrm{root}(v) \neq \cdot$, we have $\ulcorner v \urcorner \in \mathrm{Sub}(\ulcorner q \urcorner)$. Then (without loss of generality we consider only case $v = \mathrm{bin}(p, p')$) $\ulcorner v \urcorner = \mathrm{bin}(\ulcorner p \urcorner, \ulcorner p' \urcorner)$ and thus $\ulcorner p \urcorner \in \mathrm{Sub}(\ulcorner v \urcorner)$. Therefore, (since $\ulcorner v \urcorner \in \mathrm{Sub}(\ulcorner q \urcorner)$) $\ulcorner p \urcorner \in \mathrm{Sub}(\ulcorner q \urcorner)$ and then $\mathrm{Sub}(\ulcorner p \urcorner) \subseteq \mathrm{Sub}(\ulcorner q \urcorner)$. From this follows $\mathrm{size}(\ulcorner p \urcorner) \leq \mathrm{size}(\ulcorner q \urcorner)$.
- Otherwise, there exists $v \in \mathrm{QSub}(q)$ such that $v = \cdot(L)$ and $\mathrm{elems}(p) \subseteq \mathrm{elems}(v)$ (note that $v \neq p$, otherwise we are in the one of the two cases above). From Statement 5 we have $\mathrm{elems}(\ulcorner p \urcorner) \subseteq \mathrm{elems}(\ulcorner v \urcorner)$. Note that by definition of normalization and since $v \in \mathrm{QSub}(q)$ (and thus $v = q$ or there exists $v' \in \mathrm{Sub}(q)$ such that $v' = \mathrm{bin}(v, p')$ or $v' = \mathrm{bin}(p', v)$ or $v' = \mathrm{priv}(v)$) we have that $\ulcorner v \urcorner \in \mathrm{Sub}(\ulcorner q \urcorner)$, moreover, $\mathrm{Sub}(\ulcorner v \urcorner) \subseteq \mathrm{Sub}(\ulcorner q \urcorner)$. Then from Statement 17 we have $\mathrm{elems}(\ulcorner v \urcorner) \subseteq \mathrm{Sub}(\ulcorner q \urcorner)$. Thus, $\mathrm{elems}(\ulcorner p \urcorner) \subseteq \mathrm{Sub}(\ulcorner q \urcorner)$, consequently, $\mathrm{Sub}(\mathrm{elems}(\ulcorner p \urcorner)) \subseteq \mathrm{Sub}(\ulcorner q \urcorner)$. Now, if $\mathrm{elems}(\ulcorner p \urcorner) = \{\ulcorner p \urcorner\}$, then the statement becomes trivial. Otherwise, $\ulcorner p \urcorner = \cdot(\mathrm{elems}(\ulcorner p \urcorner))$, and then $\mathrm{Sub}(\ulcorner p \urcorner) = \ulcorner p \urcorner \cup \mathrm{Sub}(\mathrm{elems}(\ulcorner p \urcorner))$. Since we have already shown that $\mathrm{Sub}(\mathrm{elems}(\ulcorner p \urcorner)) \subseteq \mathrm{Sub}(\ulcorner q \urcorner)$, to prove the statement it is enough to show that there exists $p' \in \mathrm{Sub}(\ulcorner q \urcorner)$ such that $p' \notin \mathrm{Sub}(\mathrm{elems}(\ulcorner p \urcorner))$. For such value we can choose $p' = \ulcorner v \urcorner$ ($\ulcorner v \urcorner$ cannot be in $\mathrm{Sub}(\mathrm{elems}(\ulcorner p \urcorner))$, since $\mathrm{elems}(\ulcorner p \urcorner) \subseteq \mathrm{elems}(\ulcorner v \urcorner) \subseteq \mathrm{Sub}(\ulcorner v \urcorner)$ and in the current case $\mathrm{root}(\ulcorner v \urcorner) = \cdot$).

Lemma 5. *Given a constraint system $\mathcal{S}$ and its model σ. Then substitution $\pi(\mathrm{H}(\sigma))$ is normalized.*

Proof. For any $x \in \mathrm{dom}(\pi(\mathrm{H}(\sigma)))$, $x\,\pi(\mathrm{H}(\sigma)) = \pi(H(x\sigma)) = \ulcorner \pi(H(x\sigma)) \urcorner$ (by Lemma 4). □

Lemma 6. *For any normalized term t, $\mathrm{QSub}(t) = \mathrm{Sub}(t)$.*

Proof. By induction on $\mathrm{size}(t)$.

- $\mathrm{size}(t) = 1$. Then $t \in \mathcal{X} \cup \mathcal{A}$, and thus, $\mathrm{QSub}(t) = \mathrm{Sub}(t) = \{t\}$.

- Suppose that for any $t : \text{size}(t) < k$ $(k > 1)$, $\text{QSub}(t) = \text{Sub}(t)$.
- Given a term $t : \text{size}(t) = k$, $k > 1$. We need to show that $\text{QSub}(t) = \text{Sub}(t)$.
  - $t = \text{bin}(t_1, t_2)$. Then $\text{QSub}(\text{bin}(t_1, t_2)) = \{t\} \cup \text{QSub}(t_1) \cup \text{QSub}(t_2) = (\text{as size}(t_i) < k) = \{t\} \cup \text{Sub}(t_1) \cup \text{Sub}(t_2) = \text{Sub}(t)$
  - $t = \text{priv}(t_1)$. Then $\text{QSub}(\text{priv}(t_1)) = \{t\} \cup \text{QSub}(t_1) = \{t\} \cup \text{Sub}(t_1) = \text{Sub}(t)$
  - $t = \cdot(L)$. As t is normalized, $\forall p \in L$, $p \neq \cdot(L_p)$. Then $\text{elems}(L) \approx L$. Thus, we have $\text{QSub}(t) = \{t\} \cup \bigcup_{p \in \text{elems}(L)} \text{QSub}(p) = \{t\} \cup \bigcup_{p \in L} \text{QSub}(p) = \{t\} \cup \bigcup_{p \in L} \text{Sub}(p) = \text{Sub}(t)$. □

In Proposition 1 we remark that ACI-set of normalized terms has the same deductive expressiveness as that set of normalized terms itself.

Proposition 1. *Let T be a set of terms $T = \{t_1, \ldots, t_k\}$. Then $\pi(T) \in \text{Der}(\ulcorner T \urcorner)$ and $\ulcorner T \urcorner \subseteq \text{Der}(\{\pi(T)\})$.*

In Proposition 2 we state that a constraint system and its normal form have the same models. In Proposition 3 we show the equivalence, for a constraint system, between the existence of a model and the existence of a normalized model. As a consequence we will need only to consider normalized constraints and models in the sequel.

Proposition 2. *The substitution σ is a model of constraint system $\mathcal{S}$ if and only if σ is a model of $\ulcorner \mathcal{S} \urcorner$.*

Proof. By definition, σ is a model of $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$, iff $\forall i \in \{1, \ldots, n\}, \ulcorner t_i \sigma \urcorner \in \text{Der}(\ulcorner E_i \sigma \urcorner)$. But by Lemma 4 we have that $\ulcorner t_i \sigma \urcorner = \ulcorner \ulcorner t_i \urcorner \sigma \urcorner$ and $\ulcorner E_i \sigma \urcorner = \ulcorner \ulcorner E_i \urcorner \sigma \urcorner$. Thus, σ is a model of $\mathcal{S}$ if and only if σ is a model of $\ulcorner \mathcal{S} \urcorner$. □

Proposition 3. *The substitution σ is a model of constraint system $\mathcal{S}$ if and only if $\ulcorner \sigma \urcorner$ is a model of $\mathcal{S}$.*

Proof. Proof is similar to one of Proposition 2. □

2.1.2 Ground case of DY+ACI

In this subsection we consider a question of ground derivability within DY+ACI: given a normalized ground term t and a set of normalized ground terms E is t derivable from E using operations from Table 2.2? In this subsection we present an algorithm that is able to answer this question in polynomial time. This result will be used in Algorithm 4, where we need to check whether a ground substitution σ satisfies a constraint system $\mathcal{S}$. On the other hand, the polynomial complexity of such check is essential for proving that the satisfiability of DY+ACI constraint system is in NP.

First, for the ground case we consider an equivalent to DY+ACI deduction system DY+ACI' (Table 2.3) obtained from the first by replacing a set of rules $\cdot(t_1, \ldots, t_m) \to \ulcorner t_i \urcorner$ for all i with a new set of rules $t \to \ulcorner s \urcorner$ for all $s \in \text{elems}(t)$, where $t = \cdot(L)$.

Now, we show an equivalence of the two deduction systems.

Lemma 7. $t \in \text{Der}_{DY+ACI}(E) \iff t \in \text{Der}_{DY+ACI'}(E)$

Composition rules	Decomposition rules
$t_1, t_2 \to \ulcorner \mathrm{enc}(t_1, t_2) \urcorner$	$\mathrm{enc}(t_1, t_2), \ulcorner t_2 \urcorner \to \ulcorner t_1 \urcorner$
$t_1, t_2 \to \ulcorner \mathrm{aenc}(t_1, t_2) \urcorner$	$\mathrm{aenc}(t_1, t_2), \ulcorner \mathrm{priv}(t_2) \urcorner \to \ulcorner t_1 \urcorner$
$t_1, t_2 \to \ulcorner \mathrm{pair}(t_1, t_2) \urcorner$	$\mathrm{pair}(t_1, t_2) \to \ulcorner t_1 \urcorner$
$t_1, \mathrm{priv}(t_2) \to \ulcorner \mathrm{sig}(t_1, \mathrm{priv}(t_2)) \urcorner$	$\mathrm{pair}(t_1, t_2) \to \ulcorner t_2 \urcorner$
$t_1, \ldots, t_m \to \ulcorner \cdot(t_1, \ldots, t_m) \urcorner$	$t \to \ulcorner s \urcorner$ for all $s \in \mathrm{elems}(t)$, where $t = \cdot(L)$
$t_1, t_2 \to \ulcorner \mathrm{apply}(t_1, t_2) \urcorner$	

Table 2.3: DY+ACI' rules

Proof sketch. We have to show that every rule of one deduction system can be simulated by a combination of rules from the other. Since for the rules that are common for the both deduction systems such simulation is trivial, we consider here only rules from symmetric difference of two deduction systems.

The DY+ACI' rules $\forall s \in \mathrm{elems}(t)\ t \to \ulcorner s \urcorner$, if $t = \cdot(L)$ are modeled by successive application of rules $\forall i\ \cdot(t_1, \ldots, t_m) \to \ulcorner t_i \urcorner$. The converse simulation of $\cdot(t_1, \ldots, t_m) \to \ulcorner t_i \urcorner$ by DY+ACI' is based on getting all the normalized elements of t_i and, if $|\ulcorner \mathrm{elems}(t_i) \urcorner| \geq 2$ then reconstructing $\ulcorner t_i \urcorner$ by rule $p_1, \ldots, p_l \to \ulcorner \cdot(p_1, \ldots, p_l) \urcorner$, where $p_1, \ldots, p_l$ are $\ulcorner \mathrm{elems}(t_i) \urcorner$. □

Algorithm 3: Verifying derivability of term

Input: A normalized ground constraint $E \triangleright t$
Output: $t \in \mathrm{Der}_{DY+ACI}(E)$

```
1  Let S := QSub(E) ∪ QSub(t) ∖ E;
2  Let D := E;
3  while true do
4      if exists DY rule l → r, such that l ⊆ D and r ∈ S then
5          S := S ∖ {r};
6          D := D ∪ {r};
7      else
8          if exists s ∈ S : elems(s) ⊆ D then
9              S := S ∖ {s};
10             D := D ∪ {s};
11         else
12             if exists s ∈ D : elems(s) ⊈ D then
13                 S := S ∖ elems(s);
14                 D := D ∪ elems(s);
15             else
16                 return t ∈ D;
```

Lemma 8. *For Algorithm 3 the following statements are true:*

- *for any step*[17]*,* $D \cup S = \mathrm{QSub}(E \cup \{t\})$ *and* $D \cap S = \varnothing$*;*
- *it terminates;*

[17]Consider two sequential assignments as one step

- *for any step,* $D \subseteq \mathrm{Der}_{DY+ACI}(E)$.

The following lemmas will be used to prove correctness of the algorithm.

Lemma 9.

- *For any decomposition rule* $l \to r$ *of DY+ACI', if* l *is normalized, then* r *is a quasi-subterm of* l.
- *For any composition rule* $l \to r$ *of DY+ACI' except* $\{t_1, \ldots, t_m\} \to \ulcorner\cdot(t_1, \ldots, t_m)\urcorner$, *if* l *is normalized, then* $l \subseteq \mathrm{QSub}(r)$.

Lemma 10. *After the execution of Step 16 of Algorithm 3, if* $l \to r$ *is a DY+ACI' rule, such that* $l \subseteq D$ *and* $r \notin D$, *then* $l \to r$ *is a composition rule and* $r \notin \mathrm{QSub}(E \cup \{t\})$.

Proof. Suppose, $l \to r$ is a decomposition. By Lemma 9 we have that $r \in \mathrm{QSub}(l)$ and thus, $r \in \mathrm{QSub}(D) \subseteq D \cup S$. Then $r \notin D$ implies $r \in S$, and then, Step 16 must be skipped, as branch 4 or 12 should have been visited.

Thus, $l \to r$ is a composition. As algorithm reached Step 16, that means $r \notin S$ (otherwise one of three branches must be visited and this step would be skipped). As $r \notin S$ and $r \notin D$, we have $r \notin S \cup D = \mathrm{QSub}(E \cup \{t\})$. □

Lemma 11. *Given a set of normalized terms* S *such that for any* $s \in S$, $\mathrm{elems}(s) \subseteq S$. *Then for any DY+ACI' composition rule* $l \to r$ *such that* $l \subseteq S$ *we have* $\mathrm{elems}(r) \subseteq S \cup \{r\}$.

Proof. All cases of composition rules except $t_1, \ldots, t_m \to \ulcorner\cdot(t_1, \ldots, t_m)\urcorner$ are trivial, as for them $\mathrm{elems}(r) = \{r\}$. For this case, as $\mathrm{elems}(t_i) \subseteq S$ for all i, then (by Lemma 4, statement 7) $\mathrm{elems}(\ulcorner\cdot(t_1, \ldots, t_m)\urcorner) = \mathrm{elems}(\cdot(t_1, \ldots, t_m)) = \bigcup_{i=1}^{m} \mathrm{elems}(t_i) \subseteq S$. □

Proposition 4. *Algorithm 3 is correct.*

Proof. If algorithm returns true, then by Lemma 8 we have $t \in \mathrm{Der}_{DY+ACI'}(E)$.

Show that output is correct, if algorithm returns false. Note that we consider values of D and S that they have after finishing the algorithm. Suppose that output is false ($t \notin D$), but $t \in \mathrm{Der}_{DY+ACI'}(E)$. Then there exists minimal by length derivation $\{E_i\}_{i=0,\ldots,n}$ where $n \geq 1$, $D = E_0$ (as $D \subseteq \mathrm{Der}_{DY+ACI'}(E)$ and $t \notin D$) and $t \in E_n$ and $E_{i+1} \setminus E_i \neq \varnothing$ and $E_i \to_{l_i \to r_i} E_{i+1}$ for all $i = 0, \ldots, n-1$. Then, applying Lemma 10 we have $l_0 \to r_0$ is a composition, and $r_0 \notin \mathrm{QSub}(E \cup \{t\})$.

Let m be the smallest index such that there exists $s \in S = \mathrm{QSub}(E \cup \{t\}) \setminus D$ and $s \in E_m$.

Let k be the minimal integer, such that $l_k \to r_k$ is a decomposition.

Show, $k \leq m$. Suppose the opposite, then s is built by a chain of composition rules from D. If $l_{m-1} \to r_{m-1}$ (where $r_{m-1} = s$) is

- a rule in form of $\{t_1, \ldots, t_c\} \to \ulcorner\cdot(t_1, \ldots, t_c)\urcorner$, then $\mathrm{elems}(s) \neq \{s\}$ (otherwise it contradicts to minimality of the derivation) and by Lemma 11, $\mathrm{elems}(s) \subseteq E_{m-1}$ ($m \neq 1$, otherwise this step would be executed in the algorithm). As $s \in S$, then $\mathrm{elems}(s) \subseteq \mathrm{QSub}(s) \subseteq \mathrm{QSub}(E \cup \{t\})$. If $\mathrm{elems}(s) \subseteq D$ then we got contradiction with the fact that this step would be executed in the algorithm. If there exists $e \in \mathrm{elems}(s)$ and $e \notin D$ (that means, $e \in S$), then we get a contradiction with the minimality of m, as $e \in S$ was deduced before.
- any other composition rule, then by Lemma 9, $l_{m-1} \subseteq \mathrm{QSub}(s)$, and thus, $l_{m-1} \subseteq D \cup S$. Similarly to the previous case, $m \neq 1$ and we get a contradiction with either minimality of m, or with the fact that the algorithm would have to add s into D.

Note that this also shows that decomposition rule is present in derivation.

Show, $l_k \not\subseteq D$. Suppose the opposite. Then by Lemma 9, we have $r_k \subseteq D$ what contradicts to $E_{k+1} \setminus E_k \neq \varnothing$. Thus, at least one element from l_k is not from D. Let us consider all possible decomposition rules $l_k \to r_k$:

- $\{\mathrm{pair}(t_1, t_2)\} \to \ulcorner t_1 \urcorner$. We know that $\mathrm{pair}(t_1, t_2)$ is not in D, thus, it was built by composition. As E_i are normalized, the only possible way to build by composition $\mathrm{pair}(t_1, t_2)$ from normalized terms is $\{t_1, t_2\} \to \mathrm{pair}(t_1, t_2)$ (other ways, like $\mathrm{pair}(t_1, t_2), \mathrm{pair}(t_1, t_2) \to \ulcorner \cdot (\{\mathrm{pair}(t_1, t_2), \mathrm{pair}(t_1, t_2)\}) \urcorner$ would contradict the minimality of the derivation). Thus, t_1 was derived before (or was in D), i.e. $t_1 \in E_k$. That contradicts to $E_{k+1} \setminus E_k \neq \varnothing$.
- $\{\mathrm{pair}(t_1, t_2)\} \to \ulcorner t_2 \urcorner$. Similar case.
- $\{\mathrm{enc}(t_1, t_2), \ulcorner t_2 \urcorner\} \to \ulcorner t_1 \urcorner$. The case where $\mathrm{enc}(t_1, t_2) \notin D$ has similar explanations as two cases above. Thus, $\mathrm{enc}(t_1, t_2) \in D$. That means, $t_2 \in \mathrm{QSub}(E \cup \{t\})$ and $t_2 \notin D$, i.e. $t_2 \in S$. This means, t_2 was derived before and $t_2 \in S$, what contradicts to $k \leq m$.
- $\{\mathrm{aenc}(t_1, t_2), \ulcorner \mathrm{priv}(t_2) \urcorner\} \to \ulcorner t_1 \urcorner$ is a similar case to previous one. Note that if $\mathrm{priv}(t_2)$ is not in D, then it must be obtained by decomposition.
- $t \to \ulcorner s \urcorner$, where $s \in \mathrm{elems}(t)$ and $t = \cdot(L)$. By Lemma 11, $\mathrm{elems}(t) \subseteq E_k$, that contradicts minimality of derivation ($E_{k+1} \setminus E_k \neq \varnothing$). □

2.1.3 Existence of conservative solutions

In this subsection we will show that for any satisfiable constraint system, there exist a model in special form (so called *conservative solution*). Roughly speaking, a model in this form can be uniquely defined for each variable by set of quasi-subterms of the constraint system and set of atoms (also from the constraint system) that must be "priv"ed. This will bound a search space for the model (see § 2.1.4).

First, we show that for the quasi-subterms of constraint system instantiated with its model, the application of transformation $\pi(H(\cdot))$ on such terms is equivalent modulo ACI to the result obtained if the transformation was applied only on the model.

Proposition 5. *Given a normalized constraint system $\mathcal{S}$ and its normalized model σ. For all $t \in \mathrm{QSub}(\mathcal{S})$, $\ulcorner t\,\pi(\mathrm{H}(\sigma)) \urcorner = \ulcorner \pi(H(t\sigma)) \urcorner$.*

Proof. We will prove it by induction on $|\mathrm{Sub}(t)|$, where t is normalized.

- Let $|\mathrm{Sub}(t)| = 1$. Then:
  - either $t \in \mathcal{A}$. In this case $t \in (\mathcal{A} \cap \mathrm{QSub}(\mathcal{S}))$, and as $t\mu = t$ for any substitution μ, then $\pi(H(t\sigma)) = \pi(H(t)) = \pi(\{t\}) = t$ and $t\,\pi(\mathrm{H}(\sigma)) = t$. Thus, $t\,\pi(\mathrm{H}(\sigma)) = \pi(H(t\sigma))$.
  - or $t \in \mathcal{X}$. As σ is a model and $t \in \mathrm{QSub}(\mathcal{S})$, we have $t \in \mathrm{dom}(\sigma)$, and, by definition, $t \in \mathrm{dom}(\pi(\mathrm{H}(\sigma)))$. Then, by definition of $\pi(\mathrm{H}(\sigma))$, $t\,\pi(\mathrm{H}(\sigma)) = \pi(H(t\sigma))$. □
- Assume that for some $k \geq 1$ if $|\mathrm{Sub}(t)| \leq k$, then $\ulcorner t\,\pi(\mathrm{H}(\sigma)) \urcorner = \ulcorner \pi(H(t\sigma)) \urcorner$.
- Show that for any t such that $|\mathrm{Sub}(t)| \geq k+1$, where $t = \mathrm{bin}(p, q)$ or $t = \mathrm{priv}(q)$ or $t = \cdot(t_1, \ldots, t_m)$, but $|\mathrm{Sub}(p)| \leq k$, $|\mathrm{Sub}(q)| \leq k$ and $|\mathrm{Sub}(t_i)| \leq k$, for all $i \in \{1, \ldots, m\}$, statement $\ulcorner t\,\pi(\mathrm{H}(\sigma)) \urcorner = \ulcorner \pi(H(t\sigma)) \urcorner$ is still true. We have:

– either $t = \mathrm{bin}(p,q)$. As $t = \mathrm{bin}(p,q) \in \mathrm{QSub}(\mathcal{S}) \Rightarrow p \in \mathrm{QSub}(\mathcal{S})$ and $q \in \mathrm{QSub}(\mathcal{S})$. As $|\mathrm{Sub}(p)| < |\mathrm{Sub}(t)|$ and from the induction assumption, we have $\ulcorner p\,\pi(\mathrm{H}(\sigma))\urcorner = \ulcorner\pi(H(p\sigma))\urcorner$. The same holds for q.

Again, as $\mathrm{bin}(p,q)\,\sigma \in \mathring{\mathrm{QSub}}(S)\,\sigma$ (as $\mathrm{bin}(p,q) \notin \mathcal{X}$ and $t \in \mathrm{QSub}(\mathcal{S})$) we have that $\ulcorner\pi(H(\mathrm{bin}(p,q)\,\sigma))\urcorner = \ulcorner\pi(H(\mathrm{bin}(p\sigma,q\sigma)))\urcorner = \ulcorner\pi(H(\ulcorner\mathrm{bin}(p\sigma,q\sigma)\urcorner))\urcorner = \ulcorner\pi(H(\mathrm{bin}(\ulcorner p\sigma\urcorner,\ulcorner q\sigma\urcorner)))\urcorner = \ulcorner\pi(\{\mathrm{bin}(\pi(H(\ulcorner p\sigma\urcorner)),\pi(H(\ulcorner q\sigma\urcorner)))\})\urcorner = \ulcorner\pi(\{\mathrm{bin}(\ulcorner\pi(H(p\sigma))\urcorner,\ulcorner\pi(H(q\sigma))\urcorner)\})\urcorner = \ulcorner\mathrm{bin}(\ulcorner\pi(H(p\sigma))\urcorner,\ulcorner\pi(H(q\sigma))\urcorner)\urcorner = \ulcorner\mathrm{bin}(\ulcorner p\,\pi(\mathrm{H}(\sigma))\urcorner,\ulcorner q\,\pi(\mathrm{H}(\sigma))\urcorner)\urcorner = \ulcorner\mathrm{bin}(p,q)\,\pi(\mathrm{H}(\sigma))\urcorner = \ulcorner t\,\pi(\mathrm{H}(\sigma))\urcorner$.

– or $t = \cdot(t_1,\ldots,t_m)$. As t is normalized, it implies that for all $i \in \{1,\ldots,m\}$, t_i are not in form of $\cdot(L_i)$ and then $t_i \in \mathrm{QSub}(\mathcal{S})$, and thus, we have $t_i \in \mathrm{QSub}(\mathcal{S}) \wedge \ulcorner\pi(H(t_i\sigma))\urcorner = \ulcorner t_i\,\pi(\mathrm{H}(\sigma))\urcorner$. $\pi(H(t\sigma)) = \pi(H(\cdot(t_1\sigma,\ldots,t_m\sigma))) = \pi(H(t_1\sigma)\cup\cdots\cup H(t_m\sigma))$ = (by Statement 11 of Lemma 4) $= \pi(\{\pi(H(t_1\sigma)),\ldots,\pi(H(t_m\sigma))\}) = \pi(\{\ulcorner t_1\,\pi(\mathrm{H}(\sigma))\urcorner,\ldots,\ulcorner t_m\,\pi(\mathrm{H}(\sigma))\urcorner\}) = \ulcorner\cdot(\ulcorner t_1\,\pi(\mathrm{H}(\sigma))\urcorner,\ldots,\ulcorner t_m\,\pi(\mathrm{H}(\sigma))\urcorner)\urcorner = \ulcorner\cdot(t_1\,\pi(\mathrm{H}(\sigma)),\ldots,t_m\,\pi(\mathrm{H}(\sigma)))\urcorner = \ulcorner(\cdot(t_1,\ldots,t_m))\,\pi(\mathrm{H}(\sigma))\urcorner = \ulcorner t\,\pi(\mathrm{H}(\sigma))\urcorner$

– or $t = \mathrm{priv}(q)$. Then $q \in \mathrm{QSub}(\mathcal{S})$.

$\pi(H(t\sigma)) = \pi(\{\mathrm{priv}(\pi(H(q\sigma)))\}) = \ulcorner\mathrm{priv}(\pi(H(q\sigma)))\urcorner = \ulcorner\mathrm{priv}(q\,\pi(\mathrm{H}(\sigma)))\urcorner = \ulcorner\mathrm{priv}(q)\,\pi(\mathrm{H}(\sigma))\urcorner = \ulcorner t\,\pi(\mathrm{H}(\sigma))\urcorner$.

Thus, the proposition is proven.

Now we show that relation of derivability between a term and a set of terms is stable with regard to transformation $\pi(H(\cdot))$.

Lemma 12. *Given a normalized constraint system $\mathcal{S}$ and its normalized model σ. For any DY+ACI rule $l_1,\ldots,l_k \to r$, $\pi(H(r)) \in \mathrm{Der}(\{\pi(H(l_1)),\ldots,\pi(H(l_k))\})$.*

Proof. Let us consider all the cases of DY+ACI rules:

- $t_1, t_2 \to \ulcorner\mathrm{pair}(t_1,t_2)\urcorner$ We have two cases:
  - $\exists u \in \mathring{\mathrm{QSub}}(S)$ such that $\ulcorner\mathrm{pair}(t_1,t_2)\urcorner = \ulcorner u\sigma\urcorner$. Then $\pi(H(\ulcorner\mathrm{pair}(t_1,t_2)\urcorner)) = \pi(H(\mathrm{pair}(\ulcorner t_1\urcorner,\ulcorner t_2\urcorner))) = \pi(\{\mathrm{pair}(\pi(H(\ulcorner t_1\urcorner)),\pi(H(\ulcorner t_2\urcorner)))\}) = \ulcorner\mathrm{pair}(\pi(H(t_1)),\pi(H(t_2)))\urcorner$ and then $\pi(H(\ulcorner\mathrm{pair}(t_1,t_2)\urcorner)) \in \mathrm{Der}(\{\pi(H(t_1)),\pi(H(t_2))\})$.
  - $\nexists u \in \mathring{\mathrm{QSub}}(S)$ such that $\ulcorner\mathrm{pair}(t_1,t_2)\urcorner = \ulcorner u\sigma\urcorner$. Then (by definition, Lemma 4 and Proposition 1) $\pi(H(\ulcorner\mathrm{pair}(t_1,t_2)\urcorner)) = \pi(H(\ulcorner t_1\urcorner)\cup H(\ulcorner t_2\urcorner)) \in \mathrm{Der}(\ulcorner H(t_1)\cup H(t_2)\urcorner)$. By Proposition 1, $\ulcorner H(t_1)\urcorner \subseteq \mathrm{Der}(\{\pi(H(t_1))\})$ and $\ulcorner H(t_2)\urcorner \subseteq \mathrm{Der}(\{\pi(H(t_2))\})$, then by Lemma 3, $\ulcorner H(t_1)\urcorner\cup\ulcorner H(t_2)\urcorner \subseteq \mathrm{Der}(\{\pi(H(t_1))\}\cup\{\pi(H(t_2))\})$. Now, by applying Lemma 2, we have that $\pi(H(\ulcorner\mathrm{pair}(t_1,t_2)\urcorner)) \in \mathrm{Der}(\{\pi(H(t_1))\}\cup\{\pi(H(t_2))\})$.

  So, in this case $\pi(H(r)) \in \mathrm{Der}(\{\pi(H(l_1)),\pi(H(l_2))\})$.
- $t_1, t_2 \to \ulcorner\mathrm{enc}(t_1,t_2)\urcorner$. Proof of this case can be done by analogy of previous one.
- $\{t_1, t_2\} \to \ulcorner\mathrm{aenc}(t_1,t_2)\urcorner$. The same.
- $\{t_1, t_2\} \to \ulcorner\mathrm{apply}(t_1,t_2)\urcorner$. Idem.
- $t_1, \mathrm{priv}(t_2) \to \ulcorner\mathrm{sig}(t_1,\mathrm{priv}(t_2))\urcorner$.

  - $\exists u \in \mathring{\mathrm{QSub}}(S)$ such that $\ulcorner \mathrm{sig}(t_1, \mathrm{priv}(t_2)) \urcorner = \ulcorner u\sigma \urcorner$. Then $\pi(H(\ulcorner \mathrm{sig}(t_1, \mathrm{priv}(t_2)) \urcorner)) = \pi(H(\mathrm{sig}(t_1, \mathrm{priv}(t_2)))) = \pi(\{\mathrm{sig}(\pi(H(\ulcorner t_1 \urcorner)), \pi(H(\ulcorner \mathrm{priv}(t_2) \urcorner)))\}) = \ulcorner \mathrm{sig}(\pi(H(t_1)), \mathrm{priv}(\pi(H(t_2)))) \urcorner$ and then $\pi(H(\ulcorner \mathrm{sig}(t_1, \mathrm{priv}(t_2)) \urcorner)) \in \mathrm{Der}(\{\pi(H(t_1)), \pi(H(\mathrm{priv}(t_2)))\})$ (as $\pi(H(\mathrm{priv}(t_2))) = \mathrm{priv}(\pi(H(t_2)))$).
  - $\nexists u \in \mathring{\mathrm{QSub}}(S)$ such that $\ulcorner \mathrm{sig}(t_1, \mathrm{priv}(t_2)) \urcorner = \ulcorner u\sigma \urcorner$. This case can be proved in similar way as done for $\{t_1, t_2\} \to \ulcorner \mathrm{pair}(t_1, t_2) \urcorner$.

- $t_1, \ldots, t_m \to \ulcorner \cdot(t_1, \ldots, t_m) \urcorner$. On one hand, $\pi(H(\ulcorner \cdot(t_1, \ldots, t_m) \urcorner)) = \pi(H(\cdot(t_1, \ldots, t_m))) = \pi(H(t_1) \cup \cdots \cup H(t_m)) \in \mathrm{Der}(\ulcorner H(t_1) \cup \cdots \cup H(t_m) \urcorner)$. On the other hand, $\ulcorner H(t_i) \urcorner \subseteq \mathrm{Der}(\{\pi(H(t_i))\})$ (by Proposition 1).
  And thus, by Lemma 3, $\pi(H(\ulcorner \cdot(t_1, \ldots, t_m) \urcorner)) \in \mathrm{Der}(\{\pi(H(t_1)), \ldots, \pi(H(t_m))\})$.

- $\mathrm{enc}(t_1, t_2), \ulcorner t_2 \urcorner \to \ulcorner t_1 \urcorner$.
  Here we have to show that $\pi(H(\ulcorner t_1 \urcorner))$ is derivable from $\{\pi(H(\mathrm{enc}(t_1, t_2))), \pi(H(\ulcorner t_2 \urcorner))\}$. Consider two cases:

  - $\exists u \in \mathring{\mathrm{QSub}}(S)$ such that $\ulcorner \mathrm{enc}(t_1, t_2) \urcorner = \ulcorner u\sigma \urcorner$.
    Then $\pi(H(\mathrm{enc}(t_1, t_2))) = \mathrm{enc}(\pi(H(t_1)), \pi(H(t_2)))$ and then $\pi(H(\ulcorner t_1 \urcorner)) = \pi(H(t_1)) \in \mathrm{Der}(\{\mathrm{enc}(\pi(H(t_1)), \pi(H(t_2))), \ulcorner \pi(H(\ulcorner t_2 \urcorner)) \urcorner\})$.
  - $\nexists u \in \mathring{\mathrm{QSub}}(S)$ such that $\ulcorner \mathrm{enc}(t_1, t_2) \urcorner = \ulcorner u\sigma \urcorner$. Then $\pi(H(\mathrm{enc}(t_1, t_2))) = \pi(H(t_1) \cup H(t_2))$. Using Proposition 1, we have $\ulcorner H(t_1) \cup H(t_2) \urcorner \subseteq \mathrm{Der}(\{\pi(H(\mathrm{enc}(t_1, t_2)))\})$, thus (by Lemma 4) $\ulcorner H(t_1) \urcorner \subseteq \mathrm{Der}(\{\pi(H(\mathrm{enc}(t_1, t_2)))\})$. And then, by Proposition 1 we have that $\pi(H(t_1)) \in \mathrm{Der}(\ulcorner H(t_1) \urcorner)$. Therefore, by Lemma 2, we have $\pi(H(\ulcorner t_1 \urcorner)) = \pi(H(t_1)) \in \mathrm{Der}(\pi(H(\mathrm{enc}(t_1, t_2))))$.

- $\mathrm{aenc}(t_1, t_2), \ulcorner \mathrm{priv}(t_2) \urcorner \to \ulcorner t_1 \urcorner$. Here we have to show that $\pi(H(\ulcorner t_1 \urcorner))$ is derivable from $\{\pi(H(\mathrm{aenc}(t_1, t_2))), \pi(H(\ulcorner \mathrm{priv}(t_2) \urcorner))\}$. Consider two cases:

  - $\exists u \in \mathring{\mathrm{QSub}}(S)$ such that $\ulcorner \mathrm{aenc}(t_1, t_2) \urcorner = \ulcorner u\sigma \urcorner$.
    Then $\pi(H(\mathrm{aenc}(t_1, t_2))) = \mathrm{aenc}(\pi(H(t_1)), \pi(H(t_2)))$ and then $\pi(H(\ulcorner t_1 \urcorner)) = \pi(H(t_1)) \in \mathrm{Der}(\{\mathrm{aenc}(\pi(H(t_1)), \pi(H(t_2))), \ulcorner \mathrm{priv}(\pi(H(t_2))) \urcorner\})$. On the other hand, $\pi(H(\ulcorner \mathrm{priv}(t_2) \urcorner)) = \pi(H(\mathrm{priv}(t_2))) = \pi(\{\mathrm{priv}(\pi(H(t_2)))\}) = \ulcorner \mathrm{priv}(\pi(H(t_2))) \urcorner$.
  - $\nexists u \in \mathrm{QSub}(S)$ such that $\ulcorner \mathrm{aenc}(t_1, t_2) \urcorner = \ulcorner u\sigma \urcorner$. Then $\pi(H(\mathrm{aenc}(t_1, t_2))) = \pi(H(t_1) \cup H(t_2))$. Using Proposition 1, we have $\ulcorner H(t_1) \cup H(t_2) \urcorner \subseteq \mathrm{Der}(\{\pi(H(\mathrm{aenc}(t_1, t_2)))\})$, thus (by Lemma 4) $\ulcorner H(t_1) \urcorner \subseteq \mathrm{Der}(\{\pi(H(\mathrm{aenc}(t_1, t_2)))\})$. And then, by Proposition 1 we have that $\pi(H(t_1)) \in \mathrm{Der}(\ulcorner H(t_1) \urcorner)$. Therefore, by Lemma 2, we have $\pi(H(\ulcorner t_1 \urcorner)) = \pi(H(t_1)) \in \mathrm{Der}(\pi(H(\mathrm{aenc}(t_1, t_2))))$.

- $\mathrm{pair}(t_1, t_2) \to \ulcorner t_1 \urcorner$. Here, as usual, we consider two cases:

  - $\exists u \in \mathring{\mathrm{QSub}}(S)$ such that $\ulcorner \mathrm{pair}(t_1, t_2) \urcorner = \ulcorner u\sigma \urcorner$.
    Then $\pi(H(\mathrm{pair}(t_1, t_2))) = \mathrm{pair}(\pi(H(t_1)), \pi(H(t_2)))$ and then $\pi(H(\ulcorner t_1 \urcorner)) = \ulcorner \pi(H(t_1)) \urcorner \in \mathrm{Der}(\{\pi(H(\mathrm{pair}(t_1, t_2)))\})$.
  - $\nexists u \in \mathring{\mathrm{QSub}}(S)$ such that $\ulcorner \mathrm{pair}(t_1, t_2) \urcorner = \ulcorner u\sigma \urcorner$. Then $\pi(H(\mathrm{pair}(t_1, t_2))) = \pi(H(t_1) \cup H(t_2))$. Then by Proposition 1, we have $\ulcorner H(t_1) \cup H(t_2) \urcorner \subseteq \mathrm{Der}(\{\pi(H(\mathrm{pair}(t_1, t_2)))\})$, thus $\ulcorner H(t_1) \urcorner \subseteq \mathrm{Der}(\{\pi(H(\mathrm{pair}(t_1, t_2)))\})$. And then, by Proposition 1 we have that $\pi(H(t_1)) \in \mathrm{Der}(\ulcorner H(t_1) \urcorner)$. Therefore, by Lemma 2, we have $\pi(H(\ulcorner t_1 \urcorner)) = \pi(H(t_1)) \in \mathrm{Der}(\pi(H(\mathrm{pair}(t_1, t_2))))$.

- $\mathrm{pair}(t_1, t_2) \to \ulcorner t_2 \urcorner$. Proof as above.

- $\cdot(t_1,\ldots,t_m) \to \ulcorner t_i \urcorner$. We have $\pi(H(\cdot(t_1,\ldots,t_2))) = \pi(H(t_1) \cup \cdots \cup H(t_m))$. Then by Proposition 1, $\ulcorner H(t_1) \cup \cdots \cup H(t_m) \urcorner \subseteq \mathrm{Der}(\pi(H(\cdot(t_1,\ldots,t_m))))$; and then $\ulcorner H(t_i) \urcorner \subseteq \mathrm{Der}(\pi(H(\cdot(t_1,\ldots,t_m))))$. As $\pi(H(t_i)) \in \mathrm{Der}(\ulcorner H(t_i) \urcorner)$, by Lemma 2 we have $\pi(H(\ulcorner t_i \urcorner)) = \pi(H(t_i)) \in \mathrm{Der}(\pi(H(\cdot(t_1,\ldots,t_2))))$.

As all possible cases satisfy lemma conditions, we proved the lemma. □

Using Proposition 5 and Lemma 12 we will show that transformation $\pi(H(\cdot))$ preserves the property of substitution to be a model.

Theorem 1. *Given a normalized constraint system $\mathcal{S}$ and its normalized model σ. Then substitution $\pi(\mathrm{H}(\sigma))$ also satisfies $\mathcal{S}$.*

Proof. Suppose, without loss of generality, $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$. Let us take any constraint $(E \triangleright t) \in \mathcal{S}$. As σ is a model of $\mathcal{S}$, there exists a derivation $D = A_0,\ldots,A_k$ such that $A_0 = \ulcorner E\sigma \urcorner$ and $\ulcorner t\sigma \urcorner \in A_k$.

By Lemma 12 and Lemma 3 we can easily prove that if $k > 0$, $\pi(H(A_j)) \subseteq \mathrm{Der}(\pi(H(A_{j-1})))$, $j = 1,\ldots,k$. Then, applying transitivity of $\mathrm{Der}(\cdot)$ (Lemma 2) k times, we have that $\pi(H(A_k)) \subseteq \mathrm{Der}(\pi(H(A_0)))$. In the case where $k = 0$, the statement $\pi(H(A_k)) \subseteq \mathrm{Der}(\pi(H(A_0)))$ is also true.

Using Proposition 5 we have that $\pi(H(A_0)) = \pi(H(E\sigma)) = \ulcorner E\,\pi(\mathrm{H}(\sigma)) \urcorner$, as $E \subseteq \mathrm{QSub}(\mathcal{S})$. The same for t: $\pi(H(t\sigma)) = \ulcorner t\,\pi(\mathrm{H}(\sigma)) \urcorner$, and as $\ulcorner t\sigma \urcorner \in A_k$, we have $\ulcorner t\,\pi(\mathrm{H}(\sigma)) \urcorner \in \pi(H(A_k))$. Thus, we have that $\ulcorner t\,\pi(\mathrm{H}(\sigma)) \urcorner \in \pi(H(A_k)) \subseteq \mathrm{Der}(\pi(H(A_0))) = \mathrm{Der}(\ulcorner E\,\pi(\mathrm{H}(\sigma)) \urcorner)$, that means $\pi(\mathrm{H}(\sigma))$ satisfies any constraint of $\mathcal{S}$.

From now till the end of subsection we will study a very useful property of $\pi(\mathrm{H}(\sigma))$. Proposition 6 and its corollary show, that if constraint system has a normalized model σ which sends different variables to different values, then there exists another normalized model $\pi(\mathrm{H}(\sigma))$ that sends any variable of its domain to an ACI-set of some non-variable quasi-subterms of constraint system instantiated by itself and some private keys built with atoms of the constraint system.

Lemma 13. *Given a normalized substitution σ and normalized term u. If $\ulcorner u\sigma \urcorner = \mathrm{bin}(p,q)$ and $u \notin \mathcal{X}$ and $x\sigma \neq y\sigma, x \neq y$ then $\exists s \in \mathrm{QSub}(u)$ such that $s = \mathrm{bin}(p',q')$ and $\ulcorner s\sigma \urcorner = \mathrm{bin}(p,q)$. The similar is true in the case of $\ulcorner u\sigma \urcorner = \mathrm{priv}(p)$.*

Proof. As $u = \ulcorner u \urcorner$ and $\ulcorner u\sigma \urcorner = \mathrm{bin}(p,q)$, we have:

- u not in form of $\cdot(L)$. Then, as $u \notin \mathcal{X}$ and $\ulcorner u\sigma \urcorner = \mathrm{bin}(p,q)$, we have $u = \mathrm{bin}(p',q')$ (where $\ulcorner p'\sigma \urcorner = p$ and $\ulcorner q'\sigma \urcorner = q$). Then we can choose $s = \mathrm{bin}(p',q') = u \in \mathrm{QSub}(u)$.
- $u = \cdot(t_1,\ldots,t_m)$, $m > 1$, as $u = \ulcorner u \urcorner$. Then, for all i, t_i is either a variable, or $\mathrm{bin}(p'_i,q'_i)$. But, as $x\sigma \neq y\sigma, x \neq y$ and as σ is normalized, we can claim that $\{t_1,\ldots,t_m\}$ contains at most one variable. Since $\ulcorner u\sigma \urcorner = \mathrm{bin}(p,q)$ we have $\forall i,j\ \ulcorner t_i\sigma \urcorner = \ulcorner t_j\sigma \urcorner$, and then, as $m > 1$, there exists i such that $t_i = \mathrm{bin}(p'_i,q'_i)$. Then by definition of normalization function, and from $\ulcorner u\sigma \urcorner = \mathrm{bin}(p,q)$ we have that $\ulcorner \mathrm{elems}(u\sigma) \urcorner = \{\mathrm{bin}(p,q)\}$ and as $t_i\sigma$ is an element of $u\sigma$, we have $\ulcorner \mathrm{bin}(p'_i,q'_i)\,\sigma \urcorner = \mathrm{bin}(p,q)$. Thus, we can choose $s = t_i$, as $t_i \in \mathrm{QSub}(u)$ and $t_i = \mathrm{bin}(p'_i,q'_i)$. □

The other case (priv) can be proved similarly.

Proposition 6. *Given a normalized constraint system $\mathcal{S}$ and its normalized model σ such that $\forall x, y \in \mathrm{dom}(\sigma), x \neq y \implies x\sigma \neq y\sigma$. Then for any $x \in \mathrm{dom}(\pi(\mathrm{H}(\sigma)))$ there exist $k \in \mathbb{N}$ and $s_1, \ldots, s_k \in \mathring{\mathrm{QSub}}(S) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$ such that $\mathrm{root}(s_i) \neq \cdot$ and $x\,\pi(\mathrm{H}(\sigma)) = \pi(\{s_1\,\pi(\mathrm{H}(\sigma)), \ldots, s_k\,\pi(\mathrm{H}(\sigma))\})$.*

Proof. By definition, $x\,\pi(\mathrm{H}(\sigma)) = \pi(H(x\sigma))$. Let us take any $s \in H(x\sigma)$ (note that s is a ground term). Then, by definition of $H(\cdot)$ we have:

- either $s \in \mathcal{A}$. Then, by definition of $H(\cdot)$, $s \in (\mathcal{A} \cap \mathrm{QSub}(\mathcal{S}))$. Thus, $s\,\pi(\mathrm{H}(\sigma)) = s$, $s \in \mathring{\mathrm{QSub}}(S)$, $s \neq \cdot(L)$;
- or $s = \mathrm{bin}(\pi(H(t_1)), \pi(H(t_2)))$ and $\exists u \in \mathring{\mathrm{QSub}}(S)$ such that $\ulcorner u\sigma \urcorner = \ulcorner \mathrm{bin}(t_1, t_2) \urcorner = \mathrm{bin}(\ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner)$. As all conditions of Lemma 13 are satisfied, $\exists v \in \mathrm{QSub}(u)$ such that $\ulcorner v\sigma \urcorner = \mathrm{bin}(\ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner)$ and $v = \mathrm{bin}(p, q)$ and as $u \in \mathring{\mathrm{QSub}}(S)$ then $v \in \mathring{\mathrm{QSub}}(S)$. By Proposition 5, $\ulcorner v\,\pi(\mathrm{H}(\sigma)) \urcorner = \pi(H(v\sigma)) = \pi(H(\ulcorner v\sigma \urcorner)) = \pi(H(\mathrm{bin}(t_1, t_2))) = \pi(\{\mathrm{bin}(\pi(H(t_1)), \pi(H(t_2)))\}) = \mathrm{bin}(\pi(H(t_1)), \pi(H(t_2))) = s$. That means, $\exists v \in \mathring{\mathrm{QSub}}(S), v \neq \cdot(L)$ such that $s = \ulcorner v\,\pi(\mathrm{H}(\sigma)) \urcorner$.
- or $s = \mathrm{priv}(\pi(H(t_1)))$. In this case, as s is ground, $\pi(H(t_1))$ must be an atom, moreover, by definition of $H(\cdot)$, this atom is from $(\mathcal{A} \cap \mathrm{QSub}(\mathcal{S}))$. Therefore, $s = \mathrm{priv}(a)$, where $a \in \mathcal{A} \cap \mathrm{QSub}(\mathcal{S})$ (and of course, $s \neq \cdot(L)$).

Thus, $\forall s \in H(x\sigma)$, $\exists v \in (\mathrm{QSub}(\mathcal{S})) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A}) \setminus \mathcal{X} \mid s = \ulcorner v\,\pi(\mathrm{H}(\sigma)) \urcorner$. Therefore, as $x\,\pi(\mathrm{H}(\sigma)) = \pi(H(x\sigma))$, we have that $x\,\pi(\mathrm{H}(\sigma)) = \pi(\{\ulcorner s_1\,\pi(\mathrm{H}(\sigma)) \urcorner, \ldots, \ulcorner s_k\,\pi(\mathrm{H}(\sigma)) \urcorner\}) = \pi(\{s_1\,\pi(\mathrm{H}(\sigma)), \ldots, s_k\,\pi(\mathrm{H}(\sigma))\})$, where $s_1, \ldots, s_k \in \mathring{\mathrm{QSub}}(\mathcal{S}) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$ and $s_i \neq \cdot(L), \forall 1 \leq i \leq k$. That proves the proposition. □

Corollary 1. *Given normalized constraint system $\mathcal{S}$ and its normalized model σ' such that $x \neq y \implies x\sigma' \neq y\sigma'$. Then there exists a normalized model σ of $\mathcal{S}$ such that for any $x \in \mathrm{dom}(\sigma)$ there exist $k \in \mathbb{N}$ and $s_1, \ldots, s_k \in \mathring{\mathrm{QSub}}(S) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$ such that $x\sigma = \pi(\{s_1\sigma, \ldots, s_k\sigma\})$ and $s_i \neq s_j$, if $i \neq j$ and $s_i \neq \cdot(L)$ for all i.*

Any normalized model with property shown in Corollary 1 we will call *conservative*:

Definition 2.1.15 A substitution σ is a *conservative model* of constraint system $\mathcal{S}$, iff

1. σ is normalized;
2. σ is a model of $\mathcal{S}$;
3. For any $x \in \mathrm{dom}(\sigma)$ there exist $k \in \mathbb{N}$ and $s_1, \ldots, s_k \in \mathring{\mathrm{QSub}}(S) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$ such that $x\sigma = \pi(\{s_1\sigma, \ldots, s_k\sigma\})$ and $s_i \neq s_j$, if $i \neq j$ and $s_i \neq \cdot(L)$ for all i.

2.1.4 Bounds on conservative solutions

To get a decidability result, we first show an upper bound on size of conservative model. Then we reduce any satisfiable constraint system to another satisfiable one that admits a conservative model (this reduction is, in fact, using one name for the variables on which some preliminary fixed model returns equal values). Moreover the considered conservative model of the obtained constraint system can be easily extended to a conservative model (of the same size!) of the initial constraint system. We also show that the reduced constraint system is not bigger (by size) than the original one. This means that the original constraint system has a model which is bounded with regard to the size of the constraint system. Thus, we obtain an existence of a model with bounded size for any satisfiable constraint system.

Lemma 14. *Given a normalized constraint system $\mathcal{S}$ and its conservative model σ. Then* $\forall x \in \mathrm{Vars}(\mathcal{S}), \mathrm{QSub}(x\sigma) \subseteq \ulcorner \mathrm{QSub}(\mathcal{S})\sigma \urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$.

Proof. Given a ground substitution σ, let us define a strict total order on variables: $x \sqsubset y \iff (\mathrm{size}(x\sigma) < \mathrm{size}(y\sigma)) \vee (\mathrm{size}(x\sigma) = \mathrm{size}(y\sigma) \wedge x \prec y)$.

By Proposition 6 $\forall x\, x\sigma = \pi(\{s_1^x\sigma, \ldots, s_{k^x}^x\sigma\})$, where $s_i^x \in (\mathrm{QSub}(\mathcal{S}) \setminus \mathcal{X}) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$ and $s_i^x \neq \cdot(L)$.

Let us show that if $y \in \mathrm{Vars}(s_i^x)$ for some i, then $y \sqsubset x$. Suppose that $y \in \mathrm{Vars}(s_i^x)$ and $x \sqsubset y$. Then $\mathrm{size}(x\sigma) = \mathrm{size}(\pi(\{s_1^x\sigma, \ldots, s_{k^x}^x\sigma\})) = \mathrm{size}(\ulcorner \cdot(s_1^x\sigma, \ldots, s_{k^x}^x\sigma) \urcorner) \geq$ (by Lemma 4) $\geq \mathrm{size}(\ulcorner s_i^x\sigma \urcorner) > \mathrm{size}(\ulcorner y\sigma \urcorner)$, because we know that $s_i^x = \mathrm{bin}(p, q)$ or $s_i^x = \mathrm{priv}(p)$ and $y \in \mathrm{Vars}(s_i^x)$ (for example, in the first case, $\mathrm{size}(\ulcorner s_i^x\sigma \urcorner) = \mathrm{size}(\mathrm{bin}(\ulcorner p\sigma \urcorner, \ulcorner q\sigma \urcorner)) = 1 + \mathrm{size}(\{\ulcorner p\sigma \urcorner, \ulcorner q\sigma \urcorner\})$ and as $y \in \mathrm{Vars}(\{p, q\})$, using Statement 20 of Lemma 4, we get $\mathrm{size}(\ulcorner s_i^x\sigma \urcorner) \geq 1 + \mathrm{size}(\ulcorner y\sigma \urcorner)$) And as $\mathrm{size}(\ulcorner y\sigma \urcorner) = \mathrm{size}(y\sigma)$ we have $y \sqsubset x$. Contradiction.

Now we show by induction main property of this lemma.

- let $x = \min_{\sqsubset}(\mathrm{Vars}(\mathcal{S}))$. Then $x\sigma = \pi(\{s_1^x\sigma, \ldots, s_{k^x}^x\sigma\}) = \ulcorner \cdot(s_1^x\sigma, \ldots, s_{k^x}^x\sigma) \urcorner$ and all s_i^x are ground (as $\nexists y \sqsubset x$). Then $x\sigma = \ulcorner \cdot(s_1^x, \ldots, s_{k^x}^x) \urcorner$. We have that $\mathrm{QSub}(x\sigma) = \{\ulcorner \cdot(s_1^x, \ldots, s_{k^x}^x) \urcorner\} \cup \mathrm{QSub}(s_1^x) \cup \cdots \cup \mathrm{QSub}(s_{k^x}^x) \subseteq \ulcorner \mathrm{QSub}(\mathcal{S})\sigma \urcorner \cup \mathrm{priv}(\mathcal{A} \cap \mathrm{QSub}(\mathcal{S}))$, as $\forall s \in \mathrm{QSub}(s_i^x)$, $s \in \mathcal{T}_g$ and $s \in \mathrm{QSub}(\mathcal{S})$ or $s = \mathrm{priv}(a)$ or $s = a$, where $a \in \mathrm{QSub}(\mathcal{S}) \cap \mathcal{A}$, therefore $s = \ulcorner s \urcorner = s\sigma \in \ulcorner \mathrm{QSub}(\mathcal{S})\sigma \urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$ and $\ulcorner \cdot(s_1^x, \ldots, s_{k^x}^x) \urcorner = x\sigma \in \ulcorner \mathrm{QSub}(\mathcal{S})\sigma \urcorner$.

- Suppose, for all $z \sqsubset y$, $\mathrm{QSub}(z\sigma) \subseteq \ulcorner \mathrm{QSub}(\mathcal{S}\sigma) \urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$.

- Show that $\mathrm{QSub}(y\sigma) \subseteq \mathrm{QSub}(\mathcal{S}\sigma) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$.
  We know that $y\sigma = \pi(\{s_1^y\sigma, \ldots, s_{k^y}^y\sigma\}) = \ulcorner \cdot(s_1^y\sigma, \ldots, s_{k^y}^y\sigma) \urcorner$ and $\forall z \in \mathrm{Vars}(s_i^y), z \sqsubset y$. Then we have $\mathrm{QSub}(y\sigma) = \{y\sigma\} \cup \mathrm{QSub}(\ulcorner s_1^y\sigma \urcorner) \cup \cdots \cup \mathrm{QSub}(\ulcorner s_{k^y}^y\sigma \urcorner)$. We know that $y\sigma \in \ulcorner \mathrm{QSub}(\mathcal{S})\sigma \urcorner$. Let us show that $\mathrm{QSub}(\ulcorner s_i^y\sigma \urcorner) \subseteq \ulcorner \mathrm{QSub}(\mathcal{S})\sigma \urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$. By Lemma 4 we have $\mathrm{QSub}(\ulcorner s_i^y\sigma \urcorner) \subseteq \ulcorner \mathrm{QSub}(s_i^y\sigma) \urcorner \subseteq \ulcorner \mathrm{QSub}(s_i^y)\sigma \cup \mathrm{QSub}(\mathrm{Vars}(s_i^y)\sigma) \urcorner = \ulcorner \mathrm{QSub}(s_i^y)\sigma \urcorner \cup \mathrm{QSub}(\mathrm{Vars}(s_i^y)\sigma)$. We can see that $\ulcorner \mathrm{QSub}(s_i^y)\sigma \urcorner \subseteq \ulcorner \mathrm{QSub}(\mathcal{S})\sigma \urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$ (as $s_i^y \in \mathrm{QSub}(\mathcal{S}) \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$); and by induction supposition and by statement proved above we have $\mathrm{QSub}(\mathrm{Vars}(s_i^y)\sigma) \subseteq \ulcorner \mathrm{QSub}(\mathcal{S})\sigma \urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$. Thus, $\mathrm{QSub}(y\sigma) \subseteq \ulcorner \mathrm{QSub}(\mathcal{S})\sigma \urcorner \cup \mathrm{priv}(\mathrm{QSub}(\mathcal{S}) \cap \mathcal{A})$.

Proposition 7. *For normalized constraint system $\mathcal{S}$ that have conservative model σ,* $\forall x \in \mathrm{Vars}(\mathcal{S}), \mathrm{size}(x\sigma) \leq 2 \times \mathrm{size}(\mathcal{S})$.

Proof. As $|\ulcorner \mathrm{Sub}(\mathcal{S})\sigma \urcorner| \leq |\mathrm{Sub}(\mathcal{S})\sigma| \leq |\mathrm{Sub}(\mathcal{S})| = \mathrm{size}(\mathcal{S})$, we have (using the fact that σ is normalized and Lemma 14) that $|\mathrm{Sub}(x\sigma)| = |\mathrm{QSub}(x\sigma)| \leq |\ulcorner \mathrm{QSub}(\mathcal{S})\sigma \urcorner \cup \mathrm{priv}(\mathcal{A} \cap \mathrm{QSub}(\mathcal{S}))| \leq |\ulcorner \mathrm{QSub}(\mathcal{S})\sigma \urcorner| + |\mathrm{priv}(\mathcal{A} \cap \mathrm{QSub}(\mathcal{S}))| \leq \mathrm{size}(\mathcal{S}) + |\mathcal{A} \cap \mathrm{QSub}(\mathcal{S})| \leq 2 \times \mathrm{size}(\mathcal{S})$; thus, $\mathrm{size}(x\sigma) \leq 2 \times \mathrm{size}(\mathcal{S})$. □

From this proposition and Corollary 1 we obtain an existence of bounded model for a normalized constraint system that have a model sending different variables to different values. We will reduce an arbitrary constraint system to already studied case. The target properties are stated in Proposition 8 and Corollary 2.

Lemma 15. *Given any constraint system $\mathcal{S}$ and any substitution θ such that* $\mathrm{dom}(\theta) = \mathrm{Vars}(\mathcal{S})$ *and* $\mathrm{dom}(\theta)\,\theta \subseteq \mathrm{dom}(\theta)$. *Then* $\mathrm{size}(\mathcal{S}\theta) \leq \mathrm{size}(\mathcal{S})$.

Proof. By Lemma 4, $\text{size}(\mathcal{S}\theta) = |\text{Sub}(\mathcal{S}\theta)| = |\text{Sub}(\mathcal{S})\,\theta \cup \text{Sub}(\text{Vars}(\mathcal{S})\,\theta)|$, but $\text{Vars}(\mathcal{S})\,\theta \subseteq \text{dom}(\theta) = \text{Vars}(\mathcal{S})$ ($\text{Vars}(\mathcal{S}\sigma)$ consists only of variables), and then $\text{Sub}(\text{Vars}(\mathcal{S})\,\theta) = \text{Vars}(\mathcal{S})\,\theta$. As $\text{Vars}(\mathcal{S}) \subseteq \text{Sub}(\mathcal{S})$, we have $\text{Sub}(\mathcal{S})\,\theta \cup \text{Sub}(\text{Vars}(\mathcal{S})\,\theta) = \text{Sub}(\mathcal{S})\,\theta$. Thus, $\text{size}(\mathcal{S}\theta) = |\text{Sub}(\mathcal{S})\,\theta| \leq |\text{Sub}(\mathcal{S})| = \text{size}(\mathcal{S})$. □

Definition 2.1.16 Let σ and δ be substitutions. Then we define $\sigma[\delta]$ as a substitution, such that $\text{dom}(\sigma[\delta]) = \text{dom}(\delta)$ and $\forall x \in \text{dom}(\sigma[\delta]),\ x\sigma[\delta] = (x\delta)\sigma$.

Lemma 16. *Let θ and σ be substitutions such that $\text{dom}(\theta)\,\theta = \text{dom}(\sigma)$, $\text{dom}(\sigma) \subseteq \text{dom}(\theta)$ and σ is ground. Then, for any term t, $(t\theta)\sigma = t\sigma[\theta]$.*

Proof idea. When apply θ to t, every variable x of t such that $x \in \text{dom}(\theta)$ is replaced by $x\theta$; then we apply σ to $t\theta$: every variable y of $t\theta$ is replaced by $y\sigma$, thus, every variable x from $\text{dom}(\theta)$ will be replaced to $(x\theta)\sigma$ (as $\text{dom}(\theta)\,\theta = \text{dom}(\sigma)$); and no other variables will be replaced (as $\text{dom}(\sigma) \subseteq \text{dom}(\theta)$). Thus, we can see that it is the same as in definition of $\sigma[\theta]$. □

Proposition 8. *Given any satisfiable constraint system $\mathcal{S}$. Then there exists a model σ of $\mathcal{S}$ such that $\forall x \in \text{dom}(\sigma),\ \text{size}(x\sigma) \leq 2 \times \text{size}(\ulcorner\mathcal{S}\urcorner)$*

Proof idea. Given a normalized model σ' of $\mathcal{S}$ we build such substitution θ that maps different variables whose σ'-values are equal to one (variable). In this way we obtain a new constraint system $\ulcorner\mathcal{S}\theta\urcorner$ and its normalized model. Then we may apply Corollary 1 and get a conservative model σ'' of $\ulcorner\mathcal{S}\theta\urcorner$. By applying Proposition 7 we get a bound on size for this model. On the other part, we use Lemma 16 to show that $\sigma''[\theta]$ is a model of $\ulcorner\mathcal{S}\urcorner$. And then, using the obtained bound and Lemma 15, we show the existence of a model with the desired property. □

Proof. From proposition 2 and 3 we know that if σ' is a model of $\mathcal{S}$ then $\ulcorner\sigma'\urcorner$ is a model of $\mathcal{S}$ and $\ulcorner\sigma'\urcorner$ is a model of $\ulcorner\mathcal{S}\urcorner$. Then, there exists a substitution $\theta : \text{dom}(\theta) = \text{dom}(\ulcorner\sigma'\urcorner),\ \text{dom}(\theta)\,\theta \subseteq \text{dom}(\theta),\ \sigma'' = \ulcorner\sigma'\urcorner|_{\text{dom}(\theta)\theta}$ and σ'' is a model of $\ulcorner\mathcal{S}\urcorner\theta$ such that $x\sigma'' \neq y\sigma''$, if $x \neq y$ (this is true because we can show how to build θ : given the $\ulcorner\sigma'\urcorner$ — simply split $\text{dom}(\ulcorner\sigma'\urcorner)$ into the classes of equivalence modulo $\ulcorner\sigma'\urcorner$, i.e. $x \equiv y \iff x\ulcorner\sigma'\urcorner = y\ulcorner\sigma'\urcorner$; for every class choose one representative $[x]_{\equiv}$, and then $x\theta = [x]_{\equiv}$). Note that $\theta\sigma'' = \sigma'$, that's why σ'' is a model of $\ulcorner\mathcal{S}\urcorner\theta$.

Then, as σ'' is a model of $\ulcorner\mathcal{S}\urcorner\theta$, using Proposition 2, we can say that σ'' is a model of $\ulcorner\ulcorner\mathcal{S}\urcorner\theta\urcorner$. Moreover, $x\sigma'' \neq y\sigma''$, if $\forall x, y \in \text{dom}(\sigma'')\ x \neq y$ and σ'' is normalized. Then, we can apply Corollary 1, which gives us existence of conservative model δ of $\ulcorner\ulcorner\mathcal{S}\urcorner\theta\urcorner$. That is why we can apply Proposition 7: $\forall x \in \text{Vars}(\ulcorner\ulcorner\mathcal{S}\urcorner\theta\urcorner), \text{size}(x\delta) \leq 2 \times \text{size}(\ulcorner\ulcorner\mathcal{S}\urcorner\theta\urcorner)$.

Note that using Proposition 2, Lemma 16 and definition of "model", we can easily show that $\delta[\theta]$ is a model of $\ulcorner S\urcorner$. Moreover, $\delta[\theta]$ is normalized. By definition of $\delta[\theta]$ we can say that $\forall x \in \text{dom}(\delta[\theta])\ \exists y \in \text{dom}(\theta)\,\theta : x\delta[\theta] = y\delta$ and as $y \in \mathcal{X}$ (by definition of θ), then $\text{size}(x\delta[\theta]) = \text{size}(y\delta) \leq 2 \times \text{size}(\ulcorner\ulcorner\mathcal{S}\urcorner\theta\urcorner) \leq 2 \times \text{size}(\ulcorner\mathcal{S}\urcorner\theta)$. Applying Lemma 15, we have $\text{size}(x\delta[\theta]) \leq 2 \times \text{size}(\ulcorner\mathcal{S}\urcorner)$.

Summing up, we have a normalized model $\sigma = \delta[\theta]$ of $\ulcorner\mathcal{S}\urcorner$ such that for any $x \in \text{dom}(\sigma)$ we have $\text{size}(x\sigma) \leq 2 \times \text{size}(\ulcorner\mathcal{S}\urcorner)$. □

Corollary 2. *Constraint system $\mathcal{S}$ is satisfiable if and only if there exists a normalized model of $\mathcal{S}$ defined on $\text{Vars}(\mathcal{S})$ which maps a variable to a ground term in $\mathcal{T}(\mathcal{A} \cap \text{QSub}(\ulcorner\mathcal{S}\urcorner), \varnothing)$ with size not greater than double $\text{size}(\mathcal{S})$.*

Using this result, we propose an algorithm of satisfiability of constraint system (Algorithm 4).

Algorithm 4: Solving constraint system

Input: A constraint system $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\dots,n}$
Output: Model σ, if exists; otherwise $\bot$

```
1 Guess for every variable of S a value of normalized ground substitution σ with size not
  greater than 2 × size(S);
2 if σ satisfies E_i ▷ t_i for all i = 1,...,n then
3 |  return σ
4 else
5 |  return ⊥
```

Proposition 9. *Algorithm 4 is correct.*

Proof. Let σ be an output of Algorithm 4. Then σ is a ground substitution and σ satisfies all constraints from $\mathcal{S}'$ and therefore, satisfies all constraints from $\mathcal{S}$. This means, σ is a model of $\mathcal{S}$. □

Proposition 10. *Algorithm 4 is complete.*

Proof. Suppose, $\mathcal{S}$ is satisfiable. Then, by Corollary 2, there exists a guess of value of ground substitution on every element of $\mathrm{Vars}(\mathcal{S})$ with size not greater than $2 \times \mathrm{size}(\mathcal{S})$ which represents a model σ of $\mathcal{S}$. Thus, algorithm 4 will return this σ.

2.2 Pure Dolev-Yao constraints

The previous result on constraint solving for DY+ACI theory can be projected to the classical DY case. We cannot apply it directly, since in the resulting model we may have an ACI symbol which is not in DY signature. Thus, we need to prove the decidability of DY case. The scheme we follow to solve a constraint system within DY deduction system is shown in Figure 2.5.

First, we can show that if a constraint system is satisfiable within DY, then it is satisfiable within DY+ACI (Proposition 11).

Second, as we know from the previous chapter, we can find a model of a given constraint system within DY+ACI.

Third, we will transform the model obtained from previous step (which is in DY+ACI) in such a way, that the resulting substitution will be a model of initial constraint system within DY (Theorem 2). The idea of satisfactory transformation δ is simple: we replace any ACI list of terms with nested pairs: $\cdot(\{t_1, \dots, t_n\})$ we replace with $\mathrm{pair}(t_1, \mathrm{pair}(..., t_n))$. Note that this transformation will have a linear complexity and the transformed model will have the size not more than twice bigger than initial (since we consider normalized solution in DY+ACI). This gives us a class of complexity, which is NP (see Chapter 3), for the problem of satisfiability of general constraint system within DY model.

Definition 2.2.1 We define a replacement $\delta(t) : \mathcal{T}_g \mapsto \mathcal{T}_g$ in the following way:

$$\delta(t) = \begin{cases} t, & \text{if } t \in \mathcal{X} \cup \mathcal{A}; \\ \mathrm{bin}(\delta(p), \delta(q)), & \text{if } t = \mathrm{bin}(p, q), \\ \mathrm{priv}(\delta(p)), & \text{if } t = \mathrm{priv}(p); \\ \delta(t_1), & \text{if } t = \cdot(t_1); \\ \mathrm{pair}(\delta(t_1), \delta(\cdot(t_2, \dots, t_m))), & \text{if } t = \cdot(t_1, \dots, t_m), m > 1; \end{cases}$$

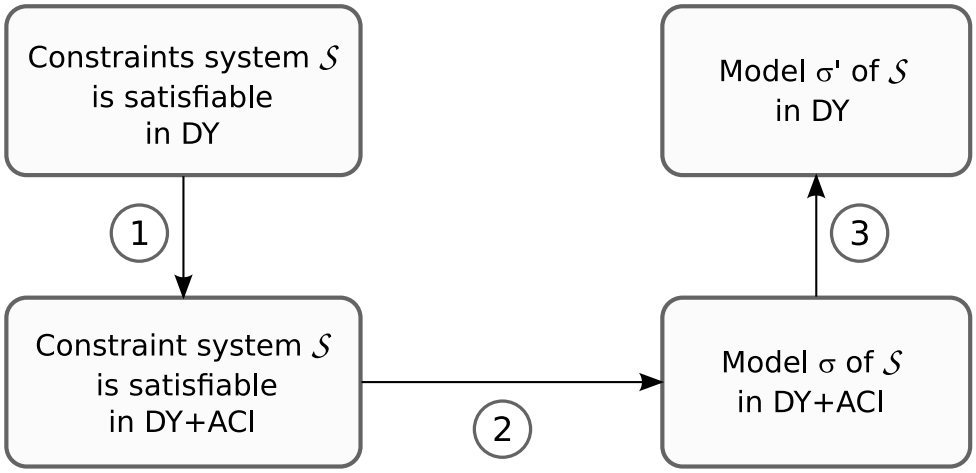


Figure 2.5: Proof plan

Definition 2.2.2 Given a substitution σ and a set of ground terms $T \subseteq \mathcal{T}_g$. We extend a definition of δ in the following way: $\delta(\sigma) = \{x \to \delta(x\sigma)\}_{x \in \mathrm{dom}(\sigma)}$; and $\delta(T) = \{\delta(t) : t \in T\}$.

Let us recall a variant of Dolev-Yao deduction system (DY) in Table 2.4.

Composition rules	Decomposition rules
$t_1, t_2 \to \mathrm{enc}(t_1, t_2)$	$\mathrm{enc}(t_1, t_2), t_2 \to t_1$
$t_1, t_2 \to \mathrm{aenc}(t_1, t_2)$	$\mathrm{aenc}(t_1, t_2), \mathrm{priv}(t_2) \to t_1$
$t_1, t_2 \to \mathrm{pair}(t_1, t_2)$	$\mathrm{pair}(t_1, t_2) \to t_1$
$t_1, \mathrm{priv}(t_2) \to \mathrm{sig}(t_1, \mathrm{priv}(t_2))$	$\mathrm{pair}(t_1, t_2) \to t_2$
$t_1, t_2 \to \mathrm{apply}(t_1, t_2)$	

Table 2.4: DY deduction system rules

Definition 2.2.3 Constraint system $\mathcal{S}$ is *standard*, iff $\forall s \in \mathrm{Sub}(\mathcal{S})\ \mathrm{root}(s) \neq \cdot$. The definition is extended in natural way to terms, sets of terms and substitutions.

We can instantiate the notion of derivation for Dolev-Yao deduction system in a natural way, and denote it as Der_{DY}.

Lemma 17. *Any standard constraint system is normalized.*

Lemma 18. *Let t be a standard term, σ be a normalized substitution. Then $t\sigma$ is normalized.*

Proposition 11. *If a standard constraint system $\mathcal{S}$ has a model σ within DY deduction system, then $\mathcal{S}$ has a model within DY+ACI deduction system.*

Proof. It is enough to consider the same model σ in DY+ACI. As $\mathcal{S}\sigma$ is normalized and as DY+ACI includes all the rules from DY, it is easy to show using the same derivation that proves σ to be a model in DY, that σ stays a model of $\mathcal{S}$ in DY+ACI. □

The goal of the following reasoning is to show that we can build a model of a constraint system within DY from a model of this constraint system within DY+ACI.

Lemma 19. *For any DY+ACI rule $l_1, \ldots, l_k \to r$, if l_i are normalized for all $i = 1, \ldots, k$ then $\delta(r) \in \mathrm{Der}_{DY}(\{\delta(l_1), \ldots, \delta(l_k)\})$.*

Proof. Let us consider all possible rules:

- $t_1, t_2 \to \ulcorner \mathrm{pair}(t_1, t_2) \urcorner$

  As t_1 and t_2 are normalized, then $\ulcorner \mathrm{pair}(t_1, t_2) \urcorner = \mathrm{pair}(t_1, t_2)$.
  We can see that $\delta(\mathrm{pair}(t_1, t_2)) = \mathrm{pair}(\delta(t_1), \delta(t_2)) \in \mathrm{Der}_{DY}(\{\delta(t_1), \delta(t_2)\})$.

- $t_1, t_2 \to \ulcorner \mathrm{enc}(t_1, t_2) \urcorner$. Proof of this case can be done by analogy of previous one.

- $t_1, t_2 \to \ulcorner \mathrm{aenc}(t_1, t_2) \urcorner$. Proof of this case can be done by analogy of previous one.

- $t_1, t_2 \to \ulcorner \mathrm{apply}(t_1, t_2) \urcorner$. Proof of this case can be done by analogy of previous one.

- $t_1, \mathrm{priv}(t_2) \to \ulcorner \mathrm{sig}(t_1, \mathrm{priv}(t_2)) \urcorner$.

  As t_1 and $\mathrm{priv}(t_2)$ are normalized, then $\ulcorner \mathrm{sig}(t_1, \mathrm{priv}(t_2)) \urcorner = \mathrm{sig}(t_1, \mathrm{priv}(t_2))$.
  We can see that $\delta(\mathrm{sig}(t_1, \mathrm{priv}(t_2))) = \mathrm{sig}(\delta(t_1), \delta(\mathrm{priv}(t_2))) = \mathrm{sig}(\delta(t_1), \mathrm{priv}(\delta(t_2))) \in \mathrm{Der}_{DY}(\{\delta(t_1), \mathrm{priv}(\delta(t_2))\})$, but $\mathrm{priv}(\delta(t_2)) = \delta(\mathrm{priv}(t_2))$.

- $t_1, \ldots, t_m \to \ulcorner \cdot(t_1, \ldots, t_m) \urcorner$.

  The fact that $\mathrm{elems}(\ulcorner \cdot(t_1, \ldots, t_m) \urcorner) = \bigcup_{i=1,\ldots,m} \mathrm{elems}(t_i)$ follows from $t_i = \ulcorner t_i \urcorner$ (for all i) and Lemma 4.

  We can (DY)-derive from $\{\delta(t_i)\}$ any term in $\delta(\mathrm{elems}(t_i))$, trivially, if $t_i \neq \cdot(L)$ and by applying rules $\mathrm{pair}(s_1, s_2) \xrightarrow{DY} s_1$ and $\mathrm{pair}(s_1, s_2) \xrightarrow{DY} s_2$ otherwise (proof by induction on size of t_i).

  One can observe that $\delta(t)$ is a pairing (composition of $\mathrm{pair}(\cdot, \cdot)$ operator with itself) of $\delta(\mathrm{elems}(t))$ (by definition of $\delta(\cdot)$ and $\mathrm{elems}\cdot$). And then, as $\delta(t)$ is limited in size, we can (DY)-derive $\delta(t)$ from $\delta(\mathrm{elems}(t))$ by iterative use of rule $s_1, s_2 \xrightarrow{DY} \mathrm{pair}(s_1, s_2)$, if needed.

  Thus, first we can derive $\delta(\mathrm{elems}(t_i))$ for all i, and then rebuild (derive with composition rules) $\delta(\ulcorner \cdot(t_1, \ldots, t_m) \urcorner)$.

- $\mathrm{enc}(t_1, t_2), \ulcorner t_2 \urcorner \to \ulcorner t_1 \urcorner$.

  As $\mathrm{enc}(t_1, t_2)$ is normalized, then $t_1 = \ulcorner t_1 \urcorner$ and $t_2 = \ulcorner t_2 \urcorner$. Thus,
  $\delta(t_1) \in \mathrm{Der}_{DY}(\{\mathrm{enc}(\delta(t_1), \delta(t_2)), \delta(t_2)\})$ and this is what we need, as $\delta(\mathrm{enc}(t_1, t_2)) = \mathrm{enc}(\delta(t_1), \delta(t_2))$.

- $\mathrm{pair}(t_1, t_2) \to \ulcorner t_1 \urcorner$. Similar case.

- $\mathrm{pair}(t_1, t_2) \to \ulcorner t_2 \urcorner$. Similar case.

- $\mathrm{aenc}(t_1, t_2), \ulcorner \mathrm{priv}(t_2) \urcorner \to \ulcorner t_1 \urcorner$. Similar case. Note that $\delta(\mathrm{priv}(t_2)) = \mathrm{priv}(\delta(t_2))$

- $\cdot(t_1, \ldots, t_m) \to \ulcorner t_i \urcorner$.

  As we said above, $\delta(\mathrm{elems}(\cdot(t_1, \ldots, t_m))) \subseteq \mathrm{Der}_{DY}(\{\delta(\cdot(t_1, \ldots, t_m))\})$; and since $\delta(\mathrm{elems}(t_i)) \subseteq \delta(\mathrm{elems}(\cdot(t_1, \ldots, t_m)))$, we can (DY)-derive (by composition rules) $\delta(t_i)$ from $\delta(\mathrm{elems}(t_i))$. □

Proposition 12. *Given a standard constraint system $\mathcal{S}$ and its normalized model σ in DY+ACI. Then, for any subterm of the system $t \in \mathrm{Sub}(\mathcal{S})$, we have $\delta(t\sigma) = t\delta(\sigma)$.*

Proof. The proof is done by induction as in Proposition 5.

- Let $\mathrm{size}(t) = 1$. Then either $t \in \mathcal{A}$ or $t \in \mathcal{X}$. Both are trivial cases.
- Assume that for some $k \geq 1$ if $\mathrm{size}(t) \leq k$, then $\delta(t\sigma) = t\delta(\sigma)$.
- Show that for t such that $\mathrm{size}(t) \geq k+1$, where $t = \mathrm{bin}(p,q)$ or $t = \mathrm{priv}(p)$ and $\mathrm{size}(p) \leq k$ and $\mathrm{size}(q) \leq k$, statement $\delta(t\sigma) = t\delta(\sigma)$ is still true. We have:
  - either $t = \mathrm{bin}(p,q)$. As $\delta(\mathrm{bin}(p,q)\sigma) = \delta(\mathrm{bin}(p\sigma, q\sigma)) = \mathrm{bin}(\delta(p\sigma), \delta(q\sigma)) = \mathrm{bin}(p\delta(\sigma), q\delta(\sigma)) = \mathrm{bin}(p,q)\delta(\sigma)$.
  - or $t = \mathrm{priv}(p)$. In this case the proof can be done by analogy with previous one.

  Remark: as $\mathcal{S}$ is standard, $t \neq \cdot(L)$. □

Theorem 2. *Given a standard constraint system $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$ and its normalized model σ in DY+ACI. Then $\delta(\sigma)$ is a model in DY of $\mathcal{S}$.*

Proof. Let $E \triangleright t$ be any element of $\mathcal{S}$. As σ is a model of $\mathcal{S}$, then $\ulcorner t\sigma \urcorner \in \mathrm{Der}(\ulcorner E\sigma \urcorner)$. As σ is normalized and $\mathcal{S}$ is standard, using Lemma 18 we have $\ulcorner t\sigma \urcorner = t\sigma$ and $\ulcorner E\sigma \urcorner = E\sigma$. Then, $t\sigma \in \mathrm{Der}(E\sigma)$. That means, there exists a DY+ACI derivation $D = \{A_0, \ldots, A_k\}$ such that $A_0 = E\sigma$ and $t\sigma \in A_k$.

By Lemma 19 and Lemma 3 (which also works for DY case) we can easily prove that if $k > 0$, $\delta(A_j) \subseteq \mathrm{Der}_{DY}(\delta(A_{j-1}))$, $j = 1, \ldots, k$. Note that $\delta(A)$ is a set of standard terms (and thus, normalized) for any set of terms A. Then, applying transitivity of $\mathrm{Der}_{DY}(\cdot)$ (Lemma 2 for DY) k times, we have that $\delta(A_k) \subseteq \mathrm{Der}_{DY}(\delta(A_0))$. In the case where $k = 0$, the statement $\delta(A_k) \subseteq \mathrm{Der}_{DY}(\delta(A_0))$ is also true.

Using Proposition 12 we have that $\delta(A_0) = \delta(E\sigma) = E\delta(\sigma)$, as $E \subseteq \mathrm{QSub}(\mathcal{S})$. The same for t: $\delta(t\sigma) = t\delta(\sigma)$, and as $t\sigma \in A_k$, we have $t\delta(\sigma) \in \delta(A_k)$.

Thus, we have that $t\delta(\sigma) \in \delta(A_k) \subseteq \mathrm{Der}_{DY}(\delta(A_0)) = \mathrm{Der}_{DY}(E\delta(\sigma))$, that means $\delta(\sigma)$ DY-satisfies any constraint of $\mathcal{S}$.

We present an example illustrating the theorem.

Example 20 Let us consider a standard constraint system similar to one in Example 18.

$$\mathcal{S} = \left\{ \begin{array}{lcl} \mathrm{enc}(x,a), \mathrm{pair}(c,a) & \triangleright & b \\ \mathrm{pair}(x,c) & \triangleright & a \end{array} \right\},$$

Using Algorithm 4, we can get a model of $\mathcal{S}$ within DY+ACI, let's say, as in Example 19, $\sigma = \{x \mapsto \cdot(\{a,b,c\})\}$.

Then, by applying transformation $\delta(\cdot)$, we will get $\sigma' = \delta(\sigma) = \{x \mapsto \mathrm{pair}(a, \mathrm{pair}(b,c))\}$. We can see that σ' is also a model of $\mathcal{S}$ within DY (as it was proven in Theorem 2).

Corollary 3 (of Theorem 2 and Proposition 11). *A standard constraint system $\mathcal{S}$ is satisfiable within DY iff it is satisfiable within DY+ACI.*

Chapter 3

Complexity analysis

In this chapter we study the complexity of the algorithms proposed in Chapter 2. First, we expose a representation of constraint systems to justify the selected measure of algorithms inputs. Then, we notice that the normalization algorithm is polynomial in time. Then, we show the polynomial complexity of the ground derivability algorithm. And as a consequence of the results given before, we obtain that the proposed algorithm for solving general constraint system within DY+ACI model is in NP. By reusing result for the well-formed constraints, we can show that the problem is NP-hard, and thus, it is NP-complete. Moreover, we show that, in general, for subterm-convergent deduction systems (that cover DY case) the satisfiability problem of constraint systems is undecidable, while for the well-formed case, it was shown decidable.

3.1 Measure of the input

To reason about algorithm complexity, we have to define a measure of its input. For terms and set of terms, we will use $\mathrm{size}(\cdot) + |\mathbb{E}(\cdot)|$, where $\mathbb{E}(\cdot)$ is a set of edges of DAG-representation of its argument. In other words, we use a so-called *DAG-size*. For system of constraints $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\dots,n}$ we will use $n \times \mathrm{size}(\mathcal{S}) + |\mathbb{E}(\mathcal{S})|$. The justification is given below.

Definition 3.1.1 *DAG-representation* of a constraint system $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\dots,n}$ is a tagged graph with labeled edges $\mathbb{G} = \langle \mathbb{V}, \mathbb{E}, \mathrm{tag} \rangle$ ($\mathbb{V}$ is a set of vertices and $\mathbb{E}$ is a set of edges; tag is a tagging function defined on $\mathbb{V}$) such that:

- there exists a bijection $f : \mathbb{V} \mapsto \mathrm{Sub}(\mathcal{S})$;
- $\forall v \in \mathbb{V}\ \mathrm{tag}(v) = \langle s, m \rangle$, where
  - $s = \mathrm{root}(f(v))$;
  - m is $2n$-bit integer, where $m[2i-1] = 1$ if and only if $f(v) \in E_i$ and $m[2i] = 1$ if and only if $f(v) = t_i$.
- $v_1 \xrightarrow{1} v_2 \in \mathbb{E}$ if and only if $\exists p \in \mathcal{T} : (\exists\, \mathrm{bin} : f(v_1) = \mathrm{bin}(f(v_2), p)) \vee f(v_1) = \mathrm{priv}(f(v_2))$;
- $v_1 \xrightarrow{2} v_2 \in \mathbb{E}$ if and only if $\exists p \in \mathcal{T} : \exists\, \mathrm{bin} : f(v_1) = \mathrm{bin}(p, f(v_2))$;
- $v_1 \xrightarrow{i} v_2 \in \mathbb{E}$ if and only if $f(v_1) = \cdot(L) \wedge L[i] = f(v_2)$;

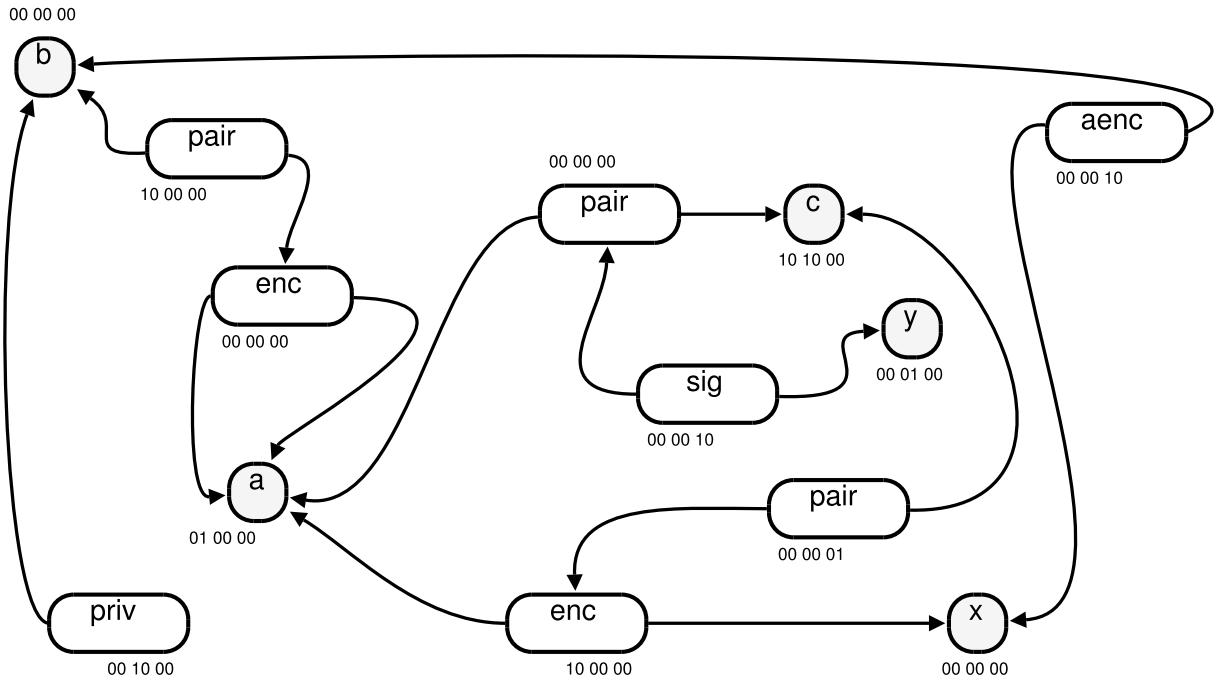


Figure 3.1: DAG-representation of constraint system $\mathcal{S}$

Example 21 A constraint system

$$\mathcal{S} = \begin{cases} \{\mathrm{enc}(a,x), \mathrm{pair}(b, \mathrm{enc}(a,a)), c\} & \triangleright\, a \\ \{\mathrm{priv}(b), c\} & \triangleright\, y \\ \{\mathrm{enc}(\mathrm{sig}(a, \mathrm{priv}(c)), y), \mathrm{aenc}(x,b)\} & \triangleright\, \mathrm{pair}(\mathrm{enc}(a,x), c) \end{cases}$$

will be represented as shown[18] in Figure 3.1. Nodes of this graph represent elements from $\mathrm{Sub}(\mathcal{S})$ by indicating their root symbols (first part of their tags) and pointers to the children.

Remark that this representation can be refined, as we know that RHS of a constraint is exactly one term. That is why we could tag a node not with $2n$ bits but with $n + \lceil \log(n+1) \rceil$ bits (concerning the second component of the tagging function).

The shown representation can be written in not more than $P(n \times |\mathbb{V}(\mathcal{S})| + |\mathbb{E}(\mathcal{S})|)$ bits of space, where n is a number of constraints, $\mathbb{V}(\cdot)$ is a set of nodes and $\mathbb{E}(\cdot)$ is a set of edges in the DAG-representation, and P is some polynomial with non-negative coefficients. As we have a bijection between $\mathbb{V}(\mathcal{S})$ and $\mathrm{Sub}(\mathcal{S})$, we obtain $|\mathbb{V}(\mathcal{S})| = \mathrm{size}(\mathcal{S})$. On the other hand, as we are not interested in rigorous estimation of complexity, but work in a polynomial class, we will estimate complexity of algorithms by taking $n \times \mathrm{size}(\mathcal{S}) + |\mathbb{E}(\mathcal{S})|$ as the measure of constraint system $\mathcal{S}$.

The DAG-representation of a term t has the similar structure as it was shown for constraint system except that it does not need the second part of a tagging function: we need only $\mathrm{root}(f(v))$ as a node's tag. The real size of this representation will be polynomially bounded by $\mathrm{size}(t) + |\mathbb{E}(t)|$. Thus we give the following definition:

Definition 3.1.2 The *measure* of term t is defined as: $\mathrm{measure}(t) = \mathrm{size}(t) + |\mathbb{E}(t)|$. For a constraint system $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$, its measure: $\mathrm{measure}(\mathcal{S}) = n \times \mathrm{size}(\mathcal{S}) + |\mathbb{E}(\mathcal{S})|$.

[18] Label "1" (resp."2") of an edge is represented by a left (resp. right) side of its source node

Note that for the normalized terms and constraint systems, number of edges in their DAG-representation are polynomially limited w.r.t. the number of vertices:

Lemma 20. *For any normalized term t, $|\mathbb{E}(t)| < (\text{size}(t))^2$. For any normalized constraint system $\mathcal{S}$, $|\mathbb{E}(\mathcal{S})| < (\text{size}(\mathcal{S}))^2$.*

Proof. Since the term (resp. constraint system) is normalized, we cannot have more than two edge between two nodes. It evidently holds for binary and unary nodes; for $\cdot$-nodes it holds because of normalization: if a $\cdot$-node has two edges to one child, the term is not normalized (one of these edges should have been removed, since such $\cdot$-node has two children, but by the definition of normal form, any two children on different positions of an $\cdot$-node must be in $\prec$ relation which does not allow equality). Therefore, as the graph is directed and acyclic, with as maximum two edges between two nodes, we have not more than $\text{size}(x) \times (\text{size}(x) - 1)$ edges (where x is a term t or constraint system $\mathcal{S}$). □

3.2 Ground derivability in DY+ACI is in P

We need to show that algorithm of checking whether a normalized ground term is derivable from a set of normalized ground terms can be done in polynomial time, that is, estimate the complexity of Algorithm 3. This property will be used in the next section to show the complexity of DY+ACI constraints satisfiability problem is in NP.

Proposition 13. *Algorithm 3 has a polynomial complexity on* $\text{size}(E \cup \{t\})$.

Proof. We will give a very coarse estimate.

First remark that in any step of algorithm, $|S|$ and $|D|$ don't exceed $|\text{QSub}(E \cup \{t\})|$.

Building of set $\text{QSub}(E) \cup \text{QSub}(t)$ takes linear time on $\text{size}(E \cup \{t\})$.

Building S will take not more than $O(|E| \times |\text{QSub}(E) \cup \text{QSub}(t)|)$, that is, not more than $O\left((\text{size}(E \cup \{t\}))^2\right)$.

The main loop has at most $|\text{QSub}(E \cup \{t\})| - |E|$ steps. Searching for DY rule with left-hand side in D and right-hand side in S is not greater than $O(|S| \times |D|^2)$ and thus, not greater than $O((\text{size}(E \cup \{t\}))^3)$. The next `if` can be performed in $O(|S| \times |D| \times (\text{size}(E \cup \{t\})))$ steps and the last `if` can be also done for cubic time. The check done in `return` statement is linear. And finally, thanks to the Statement 17 of Lemma 4, we can easily justify the claimed complexity.□

3.3 Satisfiability of general DY+ACI constraint systems is in NP

Lemma 21. *Given a term t. Normalization can be done in polynomial time on* $\text{measure}(t)$. *The same holds for a constraint system $\mathcal{S}$: normalization can be done in polynomial time on* $\text{measure}(\mathcal{S})$.

Proof idea (for the case of terms). The algorithm of term normalization works bottom-up by flattening nested ACI-sets, sorting children of ACI-set nodes, merging duplicated nodes while removing unnecessary duplicating edges and removing nodes without incoming edges (except the root-node of t). □

Proposition 14. *The general constraint system within DY+ACI satisfiability problem that Algorithm 4 solves is in NP.*

Proof idea. Algorithm 4 returns a proof for the decision problem if it exists. We have to show that the verification of this proof takes a polynomial time with regard to the input problem measure. To do this, we will normalize $\mathcal{S}\sigma$ and then apply algorithm of checking ground derivability. Using the fact that $\text{size}(x\sigma) \leq 2 \times \text{size}(\mathcal{S})$ and polynomial complexity of the normalization and the ground derivability, we can over-approximate the execution time with polynomial on $\text{measure}(\mathcal{S})$. □

Proof. As was stated before, the measure of the problem input is $\text{measure}(\mathcal{S}) = n \times \text{size}(\mathcal{S}) + |\mathbb{E}(\mathcal{S})|$, where $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\dots,n}$.

Algorithm 4 returns a normalized proof σ for decision problem if it exists. Moreover, $\text{size}(x\sigma) \leq 2 \times \text{size}(\mathcal{S})$ for any $x \in \text{Vars}(\mathcal{S})$.

First, we will normalize $\mathcal{S}\sigma$. From Lemma 21 follows, that we can do it for the time $T_{\lceil\cdot\rceil} \leq P_{\lceil\cdot\rceil}(\text{measure}(\mathcal{S}\sigma))$, where $P_{\lceil\cdot\rceil}$ is some polynomial with non-negative coefficients of some degree $m'' > 0$.

From Proposition 13 we know that check of derivability of a normalized ground term g from set of normalized ground terms G takes a polynomial time depending on $\text{size}(G \cup \{g\})$. That is, there exists a polynomial P_g with non-negative coefficients, such that number of operations (execution time) to verify the derivability (g from G) will be limited by $P_g(\text{size}(G \cup \{g\}))$. Then the execution time for checking a set of ground constraints $\{G_i \triangleright g_i\}_{i=1,\dots,n}$ will be limited by $\sum_{i=1}^{n} P_g(\text{size}(G_i \cup \{g_i\}))$.

To prove that the algorithm is in NP we need to show that execution time of check is polynomially limited by measure of algorithm's input, i.e. there exists a polynomial P, such that execution time does not exceed $O(P(n \times \text{size}(\mathcal{S}) + |\mathbb{E}(\mathcal{S})|))$ steps.

In our case, execution time T of a check will be $T = T_{\lceil\cdot\rceil} + T_g$, where T_g is a time needed for checking ground derivability of $\mathcal{S}\sigma$: $T_g \leq \sum_{i=1}^{n} P_g(\text{size}(\lceil(E_i \cup \{t_i\})\sigma\rceil))$. As P_g is a polynomial, let us say, of degree $m' > 0$, with non-negative coefficients, we can use the fact that for any positive integers $x_1, \dots, x_k$ we have $\sum_{i=1}^{k} P_g(x_i) \leq P_g(\sum_{i=1}^{k} x_i)$. Then we have $T_g \leq P_g(\sum_{i=1}^{n} \text{size}(\lceil(E_i \cup \{t_i\})\sigma\rceil))$ and by Statement 18 of Lemma 4 we have $T_g \leq P_g(\sum_{i=1}^{n} \text{size}((E_i \cup \{t_i\})\sigma))$; using the same lemma, we have

$$
\begin{aligned}
T_g \leq P_g\left(\sum_{i=1}^{n}\left(\text{size}(E_i \cup \{t_i\}) + \text{size}\left(\bigcup_x x\sigma\right)\right)\right) \leq \\
\leq P_g\left(\sum_{i=1}^{n}\left(\text{size}(E_i) + \text{size}(t_i) + \sum_x \text{size}(x\sigma)\right)\right) \leq \\
\leq P_g\left(\sum_{i=1}^{n}(2\,\text{size}(\mathcal{S})) + n \times \sum_x(\text{size}(x\sigma))\right) \leq \\
\leq P_g\left(2 \times n \times \text{size}(\mathcal{S}) + n \times \sum_x(2 \times \text{size}(\mathcal{S}))\right) \leq \\
\leq P_g\left(2 \times n \times \text{size}(\mathcal{S}) + 2 \times n \times (\text{size}(\mathcal{S}))^2\right) \leq \\
\leq P_g\left(4 \times n \times (\text{size}(\mathcal{S}))^2\right) \leq P_g\left(4 \times (n \times \text{size}(\mathcal{S}) + |\mathbb{E}(\mathcal{S})|)^2\right) = \\
= O\left((\text{measure}(\mathcal{S}))^{2m'}\right).
\end{aligned}
$$

On the other hand, let us consider $T_{\lceil\cdot\rceil}$. We have $T_{\lceil\cdot\rceil} \leq P_{\lceil\cdot\rceil}(n \times \text{size}(\mathcal{S}\sigma) + |\mathbb{E}(\mathcal{S}\sigma)|)$. One can see that the number of edges in DAG-representation of $\mathcal{S}\sigma$ (where every variable x of $\mathcal{S}$ is replaced by $x\sigma$) will not exceed the number of edges in $\mathcal{S}$ plus the number of edges of all $x\sigma$: $|\mathbb{E}(\mathcal{S}\sigma)| \leq |\mathbb{E}(\mathcal{S})| + \sum_{x \in \text{Vars}(\mathcal{S})} |\mathbb{E}(x\sigma)|$. And since σ is normalized, we can use Lemma 20: $T_{\lceil\cdot\rceil} \leq P_{\lceil\cdot\rceil}(n \times \text{size}(\mathcal{S}\sigma) + |\mathbb{E}(\mathcal{S})| + \sum_{x \in \text{Vars}(\mathcal{S})} (\text{size}(x\sigma))^2)$.

Then, using Lemma 4 (Statement 15) we obtain $\mathrm{Sub}(\mathcal{S}\sigma) = \mathrm{Sub}(\mathcal{S})\,\sigma \cup \mathrm{Sub}(\mathrm{Vars}(\mathcal{S})\,\sigma)$, and thus, $\mathrm{size}(\mathcal{S}\sigma) \leq |\mathrm{Sub}(\mathcal{S})\,\sigma| + \sum_{x\in\mathrm{Vars}(\mathcal{S})} \mathrm{size}(x\sigma)$. From Statement 16 of Lemma 4 follows that $|\mathrm{Sub}(\mathcal{S})\,\sigma| \leq \mathrm{size}(\mathcal{S})$. Since $\mathrm{size}(x\sigma) \leq 2 \times \mathrm{size}(\mathcal{S})$ and $|\mathrm{Vars}(\mathcal{S})| \leq \mathrm{size}(\mathcal{S})$, we obtain $\mathrm{size}(\mathcal{S}\sigma) \leq \mathrm{size}(\mathcal{S}) + 2 \times (\mathrm{size}(\mathcal{S}))^2$. In the same way, $\sum_{x\in\mathrm{Vars}(\mathcal{S})}(\mathrm{size}(x\sigma))^2 \leq \mathrm{size}(\mathcal{S}) \times (2 \times \mathrm{size}(\mathcal{S}))^2$. Therefore, $T_{\vdash} \leq P_{\vdash}(n \times (\mathrm{size}(\mathcal{S}) + 2 \times (\mathrm{size}(\mathcal{S}))^2) + |\mathbb{E}(\mathcal{S})| + \mathrm{size}(\mathcal{S}) \times (2 \times \mathrm{size}(\mathcal{S}))^2) = O\left((\mathrm{measure}(\mathcal{S}))^{3m''}\right)$.

Summing up, $T = O\left((\mathrm{measure}(\mathcal{S}))^{3m''} + (\mathrm{measure}(\mathcal{S}))^{2m'}\right)$ that shows that a test of a proof returned by the algorithm takes polynomial time what gives us a class of complexity. □

On the other hand, we can reuse a technique presented in [RT03] to show that the satisfiability of a constraint system is an NP-hard problem. The authors encoded 3-SAT problem into an insecurity problem of a single-session sequential protocol. Because the steps of the protocol are linearly ordered, the finding of an attack is reduced to the satisfiability problem of a single well-formed constraint system. Since the class of constraint systems we considered covers the case presented in this work, we can conclude NP-hardness of our problem. Thus,

Theorem 3. *Satisfiability of general DY+ACI constraint systems is NP-complete.*

Moreover,

Corollary 4. *Satisfiability of constraint system within DY is NP-complete.*

3.4 Undecidability of a subterm deduction system

We demonstrate here that the well-formedness is a quite powerful restriction for constraint systems: satisfiability of well-formed constraints may be decidable in some theories, while satisfiability of general ones may be undecidable in the same theories.

Let us call a *subterm deduction system* a set of rules of two forms:

- composition rules: for all *public* functional symbols f, $x_1, \ldots, x_k \to f(x_1, \ldots, x_k)$
- decomposition rules: $t_1, \ldots, t_m \to s$, where s is a subterm of t_i for some i.

We show that the satisfiability of constraint system within subterm deduction system is undecidable in general. More precisely:

Instance: a subterm deduction system D, a constraint system C.

Question: is C satisfiable?

To show this, we reduce the halting problem of a Deterministic Turing Machine (TM) M that works on a single tape. We consider the tape alphabet $\Gamma = \{0, 1, \flat\}$, and $\flat$ is the blank symbol. The states of the TM M are in a finite set $Q = \{q_1, q_2, \ldots, q_n\}$. Without loss of generality we can assume that q_1 (resp. q_n) is the unique initial (resp. accepting) state.

In order to represent Turing machine configuration as terms we shall introduce a set of variables $\mathcal{X}$ and an alphabet $\mathcal{F}$

$$\mathcal{F} := \{0, 1, \flat, \bot\} \cup Q,$$

where $\mathcal{F} \setminus \{\bot\}$ are public functional symbols.

The TM configuration with tape $\perp\, abcde\, \perp$, (where $\perp$ is an endmarker), with symbol d under the head, and state q will be represented by the following term of $q(c(b(a(\perp), d(e(\perp), x)$ where $x \in \mathcal{X}$ and $a, b, c, d, e \in \{0, 1, \flat\}$.

The composition rules we consider for the TM are $u \to f(u)$ for each $f \in \{0, 1, \flat\}$ and $u, v, w \to q(u, v, w)$ for each $q \in Q$. For each TM transition of M we will introduce some decomposition deduction rule that can be applied on a term representation $q(u, v, q'(u', v', x'))$ iff the transition can be applied to a configuration represented by $q(u, v, _)$ and generate a configuration represented by $q'(u', v', _)$. For each TM instruction of type: *"In state q reading a go to state q' and write b"*, we define the following rule for $a, b \in \{0, 1, \flat\}$:

$$q(u, a(v), q'(u, b(v), x)) \to q'(u, b(v), x)$$

For each instruction of type: *"In state q reading a go to state q' and move right"*, we define the following rules for $a \in \{0, 1, \flat\}$:

$$q(u, a(v), q'(a(u), v, x)) \to q'(a(u), v, x)$$

A rule is for extending the tape on the right when needed:

$$q(u, \perp, q'(\flat(u), \perp, x)) \to q'(\flat(u), \perp, x)$$

For each instruction of type: *"In state q reading a go to state q' and move left"*, we define the following rules for $a \in \{0, 1, \flat\}$:

$$q(a(u), v, q'(u, a(v), x)) \to q'(u, a(v), x)$$

A rule is for extending the tape on the left when needed:

$$q(\perp, a(v), q'(\perp, \flat(a(v)), x)) \to q'(\perp, \flat(a(v)), x)$$

The resulting deduction system D_M is obviously a subterm deduction system.

Let us consider a constraint $\mathcal{S}$ to be solved modulo D_M:

$$\{q_1(\perp, \perp, x)\} \rhd q_n(\perp, \perp, y)$$

This constraint is satisfiable iff there is a sequence of transitions of M from a configuration with initial state q_1 and empty tape to a configuration with an accepting state and empty tape. Hence the constraint solving problem is undecidable.

Let us recall the definition of some properties of constraint systems. These two properties are natural for modeling standard security protocols:

Variable origination: $\forall i,\ \forall x \in \mathrm{Vars}\,(E_i)\ \ \exists j < i\ \ x \in \mathrm{Vars}\,(t_j)$,

Monotonicity: $j < i \implies E_j \subseteq E_i$.

Note that $\{\{q_1(\perp, \perp, x)\} \rhd q_n(\perp, \perp, y)\}$ is obviously monotonic.

As a consequence, satisfiability of monotonic constraint systems (but without variable origination) is undecidable. Here is another constraint system, where variable origination is satisfied, but monotony is not. It can be used for reducing the halting problem again:

$$\{\{\perp\} \rhd x,\ \{q_1(\perp, \perp, x)\} \rhd q_n(\perp, \perp, y)\}$$

As a consequence, satisfiability of constraint systems with variable origination (but without monotonicity) is undecidable.

We should note by contrast (see [Bau05]) that constraint solving in subterm convergent theories is decidable if the constraint system $\mathcal{S} = \{E_i \triangleright t_i\}_{i=1,\ldots,n}$ satisfies both variable origination and monotonicity.

Conclusions

We have presented a decision procedure for solving general deducibility constraints in the case of Dolev-Yao deduction system with and without an ACI operator. Up to our knowledge there exist only one work [Maz06] that examines DY constraint systems with no restrictions on its structure, but in this work complex keys are not allowed; moreover no equational theory was considered.

The problem we solve appears to be NP-complete. Meanwhile, if we consider arbitrary subterm-convergent deduction system, the satisfiability of general deducibility constraints gets undecidable. Note that for well-formed constraints with subterm-convergent deduction systems was shown to be decidable [Bau05].

In the next part we will show several applications of the presented theoretical result in domains of cryptographic protocols and Web Services.

Critics

We do not report any effective implementation of the decision procedure. We made an attempt to directly implement the proposed method based on Proposition 8 (p. 49), but the prototype was too slow even on very simple constraint systems. It seems that the problem requires more effective approaches, like constraint solving techniques described in, e.g., [CLS03]. A technique based on this idea was implemented in CL-AtSe [Tur06, Tur03], an effective tool for detecting protocols and services insecurity problems. We also tried to develop a technique in the same spirit for general constraints, but since the assumptions are relaxed with regard to well-formed constraints, an estimation of such technique showed that it will not be much better than an exhaustive search.

Another point to criticize is assuming only atomic keys for public key cryptography. This could be a restriction when considering specific procedures of public/private keys generation. Otherwise, an atomic piece of data suffice to model public key.

Research directions

As we mentioned, the algorithm for efficiently solving the satisfiability problem for system of general constraints is an issue. It would be good to have a tool that is able to cope with general constraints and being comparably as efficient as CL-AtSe, that cope with well-formed constraints. While both problems are NP-complete, there exist no efficient methods nor tools for the former, which is a gap to fill.

An extension that could find a good application is to consider also a negation of deducibility constraints (*negative constraints*). That is, a constraint systems in form $\{E_i \rhd t_i\}_{i=1,\ldots,n} \cup \{F_j \not\rhd p_j\}_{j=1,\ldots,k}$. A ground substitution σ is a model of such constraint system if and only if

$\ulcorner t_i \urcorner \sigma \in \mathrm{Der}(\ulcorner E_i \sigma \urcorner)$ for all $i = 1, \ldots, n$ and $\ulcorner p_i \urcorner \sigma \notin \mathrm{Der}(\ulcorner F_i \sigma \urcorner)$ for all $j = 1, \ldots, k$. For example, as will be discussed in Chapter 4, this type of constraints could help to directly express a problem of automatic composition of Web Services that satisfies additional non-disclosure properties, i.e. to build automatically a composition guaranteeing that the communication participants are not able to deduce some sensitive information they are not supposed to.

In this work we considered only an ACI symbol which can be used for modeling sets. What will happen if we consider XOR on modular exponentiation? These theories are already assumed for the case of well-formed constraints, but not for general ones. If we obtain a decision procedure taking into account these (and other) equational theories, it will permit us to reason more neatly about cryptographic schemes.

Part II

Applications

Chapter 4

Web Services composition

Contents

4.1 Introduction 68
4.1.1 Orchestration of Web Services 69
4.1.2 Choreography of Web Services 71
4.1.3 Distributed orchestration of Web Services 72
4.1.4 Terms 73
4.2 Motivation 75
4.2.1 Orchestration example 75
4.2.2 Distributed orchestration example 78
4.3 Distributed orchestration model 82
4.3.1 Distributed orchestration problem input 83
4.3.2 Execution model 83
4.3.3 Problem statement 88
4.4 Solution approach 88
4.4.1 Reduction to deducibility constraints 89
4.4.2 Constraint system resolution 89
4.5 AVANTSSAR Orchestrator 89
4.5.1 AVANTSSAR Validation Platform 90
4.5.2 Input Format 91
4.5.3 Output Format 96
4.5.4 Approach overview 96
4.5.5 Architecture overview 99
4.5.6 Security Loopback 103
4.5.7 A toy example 105
4.5.8 Some experimental results 109
4.6 Conclusions 112

4.1 Introduction

Web Services implement a Service Oriented Architecture (SOA) [Gro06, OASb]. One of the guiding principles of SOA is *composability*. This principle relies on other SOA principles like reusability (the ability of services to be reused) and interoperability (the ability of services to work together). The important benefit of supporting the composability is the possibility to create more complex services with new functionalities that were not provided by an already existing one.

In order to illustrate a composite Web Services we show a simple variant of an example that became classical. It is a travel agent case study [Haa02] that has inspired different problem cases discussed in, e.g., [SK03, vRTC+04, Men02]. A travel agent receives a request from a customer to organize a trip containing the date and itinerary of the travel. The client expects from the travel agent to book a flight, hotel and a car in every city he stays. The travel agent should do the corresponding reservations using such available Web Services as Hotel Web Service, Flight Web Service and Car Rental Web Service by sending appropriate requests, and in a successful case, return to the customer an integral report with the reservation details. Since this is a frequent procedure to do by the travel agent, he could organize it in a Web Service in such a way that the client simply sends his request to this new Travel Web Service and all the necessary reservations are done automatically. The Travel Web Service is a composed one, since it reuses already existing Web Services, and, as you can see, it provides a new functionality that was not proposed by any other aforementioned Web Service. In this chapter we will discuss an automatic way of building such composed services. Note some difficulties here: since the existing services can accept only requests conforming their specification, the composed service must be able to construct such requests, for example, by adapting data received from the client.

The automated composition of Web Services techniques are of keen interest due to the complexity and error-proneness of this task as well as its high dissemination nowadays. Mainly, there are two approaches for composing them: choreography and orchestration [Pel03b]. In the choreography approach there is no central entity and each Web Service is responsible to implement its part of the composed service; the communication occurs directly between the services. By contrast, during the orchestration all the communications are passing through a *mediator* (sometimes also called *orchestrator*), a core of the process that aggregates the existing services in order to provide a new functionality.

Since the interface of a Web Service is usually described using Web Services Description Language (WSDL) standard [Wor01], we will suppose that a Web Service is defined by a set of operations which can be independently invoked by the user. Indeed, if we abstract away some technical details, a WSDL document defines services as a set of operations, every operation is represented by its name and input and output message, where a message is an abstract, typed definition of the data being communicated.

A Web Service represented by WSDL description is a stateless entity [BHM+07]. But usually this information is not enough to correctly use a service that provides multiple operation. Generally the user must follow a certain scenario to interact with a service [BKL01, vRTC+04]. For example, the user may first need to be authenticated before using other operations of the Web Service. This kind of scenarios is sometimes called Web service behavior protocol [BIPT09]. In this work we suppose that the behavior protocol of Web Service is given together with a service description, and we do not discuss the ways of obtaining it. Although, there exist works in this direction: for example, in [BIPT09] the authors present an approach of automatic extraction of the behavior protocols directly from WSDL and deployed Web Service by means of testing.

Accepting this assumption, we will think of a service as a protocol role, which is modeled

in terms of *strands* [FJHG99] (in short, a finite sequence[19] of receive/reply actions), where messages are represented as first-order terms. Thus, we will capture both a Web Services's WSDL description and its use cases.

Besides the list of operations (with its input and output message patterns), a WSDL description may specify secure bindings (like HTTP over SSL) and also a security policy using WS-SecurityPolicy (WSSP) standard [DLHBH+02]. By means of WSSP, a service provider can attach a security policy on parts of messages that are specified in the operations, e.g. specify messages parts that are required to be signed, encrypted, etc.

In order to be publicly used, the Web Services are usually published in Universal Description, Discovery and Integration (UDDI) [UDD04] or ebXML [OASa] registry. Such specifications offer users a unified way to find service providers through a centralized registry. It can be seen as some rough analogue of a telephone directory. We will say a Web Service is *available* if we know one can use it and we know its interface and behavior, that normally corresponds to a fact that it is published in some UDDI or ebXML registry. Several individual companies and industry groups are also starting to use "private" UDDI directories to integrate and streamline access to their internal services [CDK+02]. Thus, some services might be accessible only from specific organizations. We will also try to take this possible restriction into account in a model for Web Services composition we will present.

While the majority of works on automatic composition of Web Services do not take into account security policies when adapting messages, or do not adapt them at all, we will go further and will also consider a Mediator having deduction ability that allows him not only to proceed message forwarding, but also to perform a complex message adaptation satisfying the security policies of available services in addition to the functionality requirements. That is, using the messages collected so far during the communication, a mediator can decompose messages on parts, regroup, build new ones and feed them to the available Web Services to obtain an answer that probably could not be obtained by more simple procedures.

4.1.1 Orchestration of Web Services

The centralized approach to *static* [BG06] (or *syntactic* [TBG07]) Web Services composition is the orchestration. Web Services orchestration [Pel03a] deals with a central entity called an *orchestrator* or a *mediator* that does for a glue between the client and the community of available services.

Lots of works on service composition including ours rely on a so-called *behavior model* of services [CGL+08], i.e. a model where the services are stateful and the client must follow some scenario. In the case of automata-based models, the behavior is usually presented as a Kripke structure, where transitions are labeled with services' operations [Pat09]. Here we will give a small overview of related works.

A considerable part of the approaches for the orchestration is based on so-called *Roman Model* [CGL+08]. The available services are supposed to be stateful and presented as state transition systems (state machines). The orchestration problem is informally stated as follows: given a community of available services and a target service. One must build a mapping (called *orchestrator*) which delegates all the activities of the target service to the community, such that the client can use the orchestrated community exactly in the same way as if he would use the target service. This mapping is calculated from a *simulation relation* between the target service

[19] We want to warn that our composition method is constrained by a bound on the number of instances of Web Services and on number of service calls. That is why we do not consider infinite behavior of Web Services, like loops.

and the (asynchronous product of the services of) community [BCF08, CGL+08, BCGP08] and is often presented by a state-machine.

In [Pat09] the author extends a model used in [BCD+03] (based on deterministic finite state machine framework) by considering non-determinism in services model and allowing shared memory. Moreover, the approach allows to find a *finite orchestrator generator* that can derive the set of all possible orchestrators.

In all these works services accept only data from a finite domain. This has motivated an extended and complex framework called *COLOMBO* [BCD+05] for Web Services composition which deals with messages from infinite domains. The composition problem is equivalent to finding a *mediator* service which uses messages to interact with the available services and the client such that the overall behavior of the mediated system faithfully simulates the behavior of given goal service. However, even in this model the cryptographic primitives are left out of scope.

A mediator synthesis framework was presented in [IIS10]. The aim of this work is to find an automated solution for the *protocol interoperability problem*. To make cooperate two protocols P and Q, presented as Labeled Transition System [Kel76] within environment E, the authors propose to synthesize a mediator M such that in parallel composition $P||Q||E||M$, P and Q are able to coordinate by reaching their final states. It is supposed that for each protocol P there is a corresponding ontology O_P that describe the meaning of protocols' actions. A common ontology O identifying a "common language" for two protocols is also assumed. Moreover, a mapping function from O_P to O is given for all protocols P. Using these mappings, the abstract versions A_P and A_Q of the given protocols P and Q can be built. These protocols use the same language of actions: the actions that are not covered by the common ontology are abstracted to some τ action corresponding to a communication with the third parties represented by the environment E. The obtained abstract protocols are tested whether they have compatible traces. In the positive case, the mediator M that consists of two components (M_C that speaks only the common language, and M_T that speaks only third parties language) can be built. The formalization of mediator synthesis is claimed as a part of current work. Note also that a finite set of actions in protocol specifications is assumed.

In [MFP11] a synthesis of *security adaptors* given a *security adaptation contract* was considered. The security adaptor, represented as a deterministic state machine, play the role of a mediator. While cryptography adaptation is taken into account, a lot of information is already given in form of adaptation contract. It is a state machine (extended with *environment* that stores parameters' types and values) whose alphabet consist of *security adaptation vectors*. Such a vector is, in general[20], a pair of tuples (channel, send/receive operation, contract term) denoted as $\langle c?\mathcal{T} \triangleright c'!\mathcal{T}'\rangle$, that informally states that whenever the adaptor receives an action matching the first tuple of the pair, it must eventually send the action represented by the second tuple of the pair. A contract term here represents a sequence of actions that one should perform over the known and received data (e.g., encryption and decryption) in order to match the received message (in case the term is used in LHS of the vector) or to produce a message to send (if used in RHS). An adaptor compliant with a given security adaptation contract C is a deterministic state machine which is simulated[21] by some deterministic state machine which accepts the greatest language compliant with C. Informally, the desired interactions between the adaptor and the services (with security checks and transformations to be performed on every intercepted message) are described in the contracts. Based on these contracts and behavior of

[20] Can be also a single tuple $\langle c?\mathcal{T}\rangle$ or $\langle c!\mathcal{T}\rangle$.

[21] Authors use a special notion of simulation, where final states are simulated by final states

the available services, the adaptor is synthesized to avoid deadlocks and livelocks situations. After synthesizing the adaptor, authors propose to verify it together with other Web Services with regard to some global security properties (preserving secrecy in presence of active or passive intruders). In case the adaptor is vulnerable, they propose to refine it in order to avoid attacks. Comparing to our approach we don't have any predefined security adaptation contract which can be seen as a better level of automation of the mediator generation; meanwhile we don't handle a rich behavior that can be expressed using state machines, and thus, no problems concerning livelocks can be faced in principal.

A procedure for generating "simple" orchestrators by given *contracts* of Web Services and the client was presented in [Pad08]. The Web Services as well as the client are presented by their behavior called a contract expressed using a fragment [NH87] of the Calculus of Communicating Systems (CCS). More precisely, a contract σ defined by the following grammar $\sigma ::= 0|\alpha.\sigma|\sigma + \sigma|\sigma \oplus \sigma|\operatorname{rec} x.\sigma|x$. That is, idle process, an input (like $a, b, \dots$) or output (dual to input, like $\bar{a}, \bar{b}, \dots$) action from some denumerable set, an external choice between two processes (depending on the party it communicates with), an internal choice (abstracted away in non-deterministic way some internal decision), and a recursive construction using variable x every free occurrence of which is replaced by a process $\operatorname{rec} x.\sigma$. The problem is to build an orchestrator that supervises the interaction between the client and services in order to guarantee the client's satisfaction (note that the orchestrator is able to affect the internal decisions neither of services nor client). The orchestrator is a bounded, directional, controlled buffer: the buffer can store a finite number of messages, it distinguishes messages sent to the client and to the services, and it is controlled by orchestration actions: it can receive a message from a client (asynchronously, without delivering it to a service) or from a service, send a previously received from a service message to a client and vice versa: send a previously received from the client message to a service, and also can execute a synchronous message transfer from client to service and in the opposite direction. Note that in contrast to our approach, while here the considered behavior is richer (e.g. possibility of loops via recursion), the message structure is abstracted away and (probably as a consequence of this limitation) no message security aspect is taken into account.

4.1.2 Choreography of Web Services

As we stated above, the choreography does not assume the existence of a central entity that steers the communication. In this type of composition, services communicate directly with each other. Here we discuss a few works related to the automated choreography of Web Services.

A composition method for conversation specification style, where a conversation is a sequence of messages exchanged by peers, was presented in [BFHS03]. The available services are represented as an e-composition schema: a finite set of message classes M, a finite set of abstract peers (services) P and a finite set of directed channels C. Each channel consists of two endpoints and a set of message classes allowed to be sent over this channel. The composition problem in this setting is to find a Mealy implementation (a Mealy machine) for each peer such that the set of possible conversations that can be seen from a "global watcher" is equivalent to some given conversation specification, e.g. presented as a regular language. The equivalence is considered modulo two operations "prepone" and "projection-join" presented in this work. In other words, by an abstract representation of available services and possible communication, the method reduces to building a set of Mealy machines that implements services. We should mention that the Mealy machines do not distinguish message, but only message classes, so only a finite set of (atomic) messages (message classes) is assumed.

In [BCD+09] the authors suggest a way to synthesize and verify cryptographic protocols given

a set of multiparty sessions that represent possible conversations between the participants. It is notable that the illustrating example they used was inspired by Web Services. The problem input consists of a set of multiparty sessions. Each is represented as directed graphs. Its nodes are labeled with protocol roles names and edges labeled with message descriptors and two sets of typed variables: the first one states the variables in which a role from the source node can write, and the second one — those variables from which the destination role can read. The authors propose to build first the projection for each role and then reinforce it by adding a security layer in order to guarantee integrity and secrecy properties. Moreover, a prototype of the compiler implementing this approach generates interfaces and implementations for the generated protocols in ML [MTH90].

A choreography model that focuses on the evolution of data involved in the collaboration is presented in [LW10]. The data and its possible transformations are represented as so-called *artifacts*, a Petri net-based formalism. The evolution of this data may depend on its current location (which agent is currently accessing to it). This location information is represented as a Petri net. An artifact together with location information form a special form of an artifact, called location-aware artifact. The choreography problem is presented as a controller synthesis problem: given a set of artifacts and set of goal states, one has to build a controller (also an artifact) which in composition with the given artifacts rules out any behavior that makes the goal states unreachable. Moreover, to restrict the behavior by limiting the order of actions, the authors introduce policies that are also modeled with artifacts. The authors advocate the possibility of employing existing tools for automated construction of a choreography. Note that this approach does not deal with cryptography.

4.1.3 Distributed orchestration of Web Services

In many cases a single entity (device, organization) is not able to orchestrate the Web Services due to the missing resources (e.g. absence of data requested by available services) or because of access limitations: some services are limited to a protected private network. But if partner organizations are involved in the orchestration, every party can contribute to satisfy client requests. In *distributed* or *decentralized orchestration* (e.g. [PE09, BMM06]), each partner can invoke his available services and also communicate with other partners. In this way a Mediator is distributed between the partners.

In [Pat09] the author presented another point of view on the distributed orchestration which is really close to the choreography approach with some dynamicity flavor. He proposes to attach one *local orchestrator* to every available service. The main motivation in this approach is that the community of services is not fully observable due to the fact that the services can reside in different places and can become temporarily or fully inaccessible due to network issues. These local orchestrators broadcast messages in order to every local orchestrator could reconstruct the current state of the community and history on its turn, and then can issue commands to the service it is attached to.

Another approach inspired from contract-based orchestration proposed in [Pad08] was presented in [QBHG09]. The authors build *memoryless orchestrators*, a restricted form of *simple orchestrators* [Pad08] that can constrain external choices of the client and services using a *priority order*. These orchestrators can be distributed more easily, since they can be implemented in such a way that they communicate amongst each other "as locally as possible".

Our approach

Summing up

Orchestration and Choreography are special cases of the Distributed Orchestration.

We present in in this work a scalable Web Service composition approach, that we also call *distributed orchestration*, relying on the notion of *partner* corresponding to an organization. We dedicated one partner to communicate with the Client and call him a Mediator. Each partner in a composition implements its own part of the orchestration (similarly to *choreography servers* in [PE09]). In this setting standard orchestration is a special case in which only one partner is involved, whereas choreography is another case in which there is one partner per available service.

However even in such a cooperation sensitive data should not be propagated beyond the organizational border (a company will not share secrets with partners). This is why we will introduce some restrictions on communication between partners. We will show later how distributed orchestration is still possible in such constrained setting.

Several related "distributed orchestration" notions have been advocated for in the literature (e.g. [BMM06]). However in inter-organizational business processes it is crucial to protect sensitive data of each organization providing a component service in some composition, and our main motivation is to advance the state of the art by taking into account the security policies (including so-called *non-disclosure* policy) while computing a composition.

For the non-distributed case and without implementation some initial ideas were presented in [CMR08] and [CMR09] which later took the shape of automatized tool discussed in § 4.5.

4.1.4 Terms

Representation of Services and their policies

We represent a Web Service by a sequence of *actions*, where an action is either a reception or an emission of a message pattern. Why a pattern? Since we are using behavioral model, the subsequent actions executed by the Web Service depend on values that are received.

Example 22 Let us we consider "identity" Web Service. If the service received some value a, then it should return a, however if it received b, it must return b. This is attained by using *variables*. We define the identity Web Service as "receive X, send X".

Note that the value of the variable once instantiated will be used not only inside one Web Service operation, but also along all the remaining operations.

We write $?r$ for receiving message (or message pattern) r, and $!s$ for sending message (message pattern) s. We call a finite sequence of actions a *strand* [FHG98, FHG99]. A strand having even number of actions that starts from ? and always alternates ? with ! is called a *normal strand* and represents a *synchronous* Web Service, that is, where each request implies an immediate reply. For the reason of simplicity of the formal model we consider only synchronous available services. The execution of consecutive receive-send actions $?r.!s$ in a normal strand together with the corresponding send and receive actions of the caller is called an *invocation* of a synchronous service. We express message patterns (like s and r) as first-order terms.

Definition 4.1.1 An *action* ρ is:

- either a reception $?_f r$, where r (for "receive") is a term expressing an acceptable pattern for an incoming message, and f (for "from") is an expected sender.
- or an emission $!_t s$, where s (for "send") is a term expressing a pattern for a message to be emitted, and t (for "to") is an expected receiver.

In the case where the sender (resp. receiver) is not known or is not relevant, we will write $!s$ (resp. $?r$).

Sometimes we will refer to another action, *assertion*, represented as an equality check

- Equality of two terms p and q: $\mathrm{eq}(p, q)$. This express a relation between two messages that should be checked.

We define in a natural way an application of substitution σ to an action ρ:

$$\rho\sigma = \begin{cases} ?_f(r\sigma), & \text{if } \rho = ?_f r; \\ !_t(s\sigma), & \text{if } \rho = !_t s; \\ \mathrm{eq}(p\sigma, q\sigma), & \text{if } \rho = \mathrm{eq}(p, q). \end{cases}$$

We will say that action $?_f r$ is *ground*, if r is a ground term; likewise, action $!_t s$ is ground if term s is ground and action $\mathrm{eq}(p, q)$ is ground if p and q are ground terms.

Definition 4.1.2 A *strand* is a finite sequence of actions. A *normal strand* is a strand of even length starting with a reception and that always alternate reception with emission.

Summing up

We represent a Web Service as a strand, a synchronous Web Service as a normal strand.

The functionality of a service is entirely encoded into its input-output message patterns.

Example 23 A Web Service that calculates a sine of a given value can be represented as $?X.!sin(X)$, where sin is a functional symbol.

We rely only on syntactic representation of services.

Example 24 If we need to feed a value of sine of an angle a to some service that specified as $?sin(a)$, then the other service that calculates sines of a given value must use the same symbol sin but not $sine$ or sn for this purpose, i.e. represented as $?X.!sin(X)$. Otherwise these symbols are supposed to represent different functions.

As we mentioned above, the Web Services can have security policies that give additional constraints on the messages that can be consumed and produced by Web Services. All these requirements are also supposed to be already applied on message patterns of our Web Services representation, i.e. the messages appeared in strands might be of form "encrypted", "signed" etc.

For this, we consider the signature defined in Table 2.1 (page 28) that is a variant of one used in a classical Dolev-Yao model [DY83]. Note that instead of $\mathrm{apply}(h, m)$ we will write $h(m)$, that can represent both cryptographic hash function or some functionality, like $sin(X)$, since we assume it as some one-way function.

Example 25 A Web Service that calculates the sine of a given value, and whose security policy says that the incoming messages must be encrypted with his public key K is represented as $?\,\mathrm{aenc}(X, K).!sin(X)$, while without the policy it would be represented as $?X.!sin(X)$.

In this way, where both security policy and a payloads are encoded using the same term algebra, we can reason at once on the compatibility of security policies as well as on functional composition of Web Services.

Mediator

We adopt a terminology from [BCD+03, BCD+05] and call *mediator* a service that adapts and dispatches requests from a client to the community of available services. It may perform DY operations on messages in order to adapt them. Moreover, we suppose that the mediator possesses some initial knowledge (finite set of messages) that he can also use. Apart from being a service, we expect from the mediator to be *executable*, that is, he should be able to construct a message that he sends from his current knowledge (i.e. initial knowledge plus the messages collected so far) and using only DY operations (see, e.g., Table 2.2 at page 32).

An example discussed in the next section gives an idea about the mediator service. In the case of distributed orchestration, where several mediating entities are considered, we will call them *partners*. These partners can possess different knowledge that they don't want to share but can use it in order to achieve common goals. Moreover, they can have access to different services that are not available to others.

Partners can perform simple Dolev-Yao operations on the messages to adapt them such that the result can be consumed by the community. Since these operations are "crypto-aware", the adaptation also handles the security policies.

4.2 Motivation

In this section we give two simple motivating examples. First, we discuss an instance of the Web Services orchestration problem and sketch the solution approach. And then, we extend this example to the case of distributed orchestration.

4.2.1 Orchestration example

Suppose, there is a demand on translating texts from French to English. It is known that the texts were obtained by automatically recognized hand-written documents, and thus, contain some misspells.

User wants to send a text in French and receive its English translation. Of course, in order to translate any text, it should not contain mistakes, thus, the automatically recognized text should be preliminary corrected. That is, we can express it using the following specification:

$$!t.?en(corr(t)).$$

We assume an existence of two available services:

1. SpellChecker: a service that corrects spelling. We don't care about how he does it (for example, it can use semantic network to choose the closest word from list of possible options), but only about its interface and behavior, that is, what messages it accepts, what answer he gives and in what order we should invoke his operations. In our case, the service is simple and has a unique operation. Its model:

   $$?T.!\,\mathrm{pair}\,(corr(T), n(T))\,,$$

   where T is a text, $corr(T)$ is the corrected text, $n(T)$ is a number of corrections done.

2. Translator: a service producing an automatic translation of given text from French to English. Its security policy requires all incoming messages to be encrypted with its public key. The specification we assume for him in this example is:

   $$?\,\mathrm{aenc}\,(M, K_{tr})\,.!en(M),$$

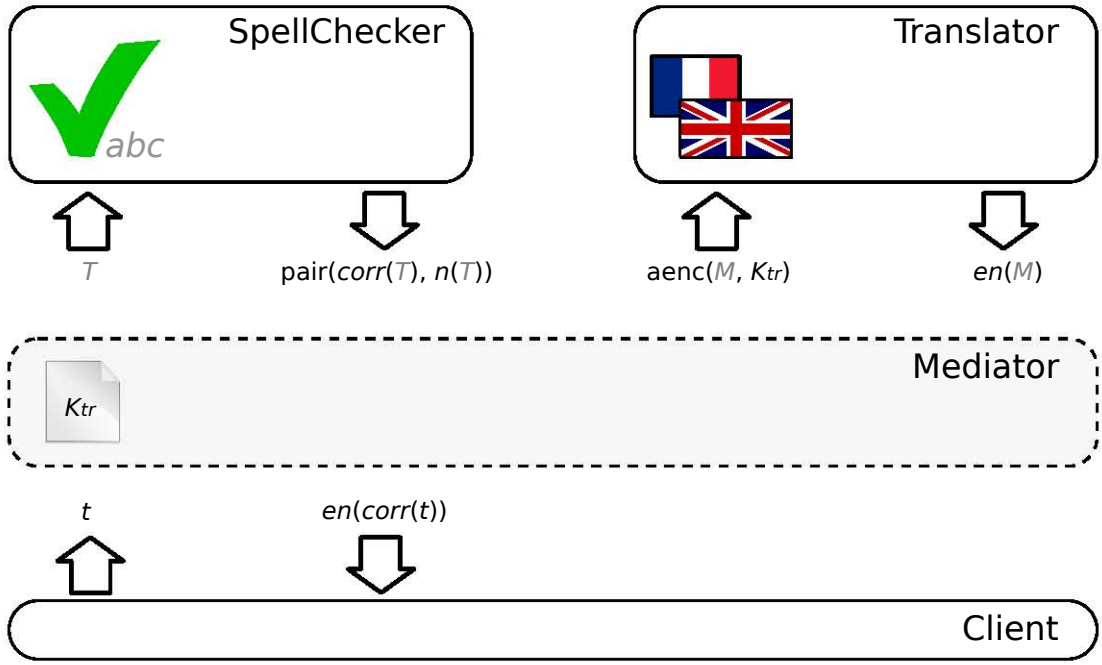


Figure 4.1: Illustration for the orchestration example

where M is a text, $en(M)$ is a translation of M into English and K_{tr} is a public key of the Translator. That is, it receives a message M (which is supposed to be correct) encrypted with its public key and replies with the translation.

The question is how to satisfy the Client's request, possibly using the available services. We will suppose that the Mediator we want to build initially knows only a public key of the Translator K_{tr} (a publicly available information).

We present an idea of how to answer this question. This problem can be solved in more general settings (§ 4.3) using deducibility constraint systems presented in Part I.

Solution idea

Assuming a finite (and fixed) number of available services invocations (e.g. we can execute invocations of services not more than 5 times) we can guess a linear order on it (what service should be invoked first, what next, etc., and where in this sequence we should communicate with the Client). Suppose that the following sequence corresponds to the guessed order:

1. Receive a request from the Client;
2. Invoke SpellChecker;
3. Invoke Translator;
4. Send a response to the Client.

Using this sequence of calls and specification of the services we can build a system of deducibility constraints (see Figure 4.2).

$$\begin{cases} \{K_{tr}, t\} & \triangleright & T \\ \{K_{tr}, t, \text{pair}\,(corr(T), n(T))\} & \triangleright & \text{aenc}\,(M, K_{tr}) \\ \{K_{tr}, t, \text{pair}\,(corr(T), n(T)), en(M)\} & \triangleright & en(corr(t)) \end{cases} \quad (4.1)\ (4.2)\ (4.3)$$

with T, M — variables.

Figure 4.2: A constraint system describing a possible orchestration.

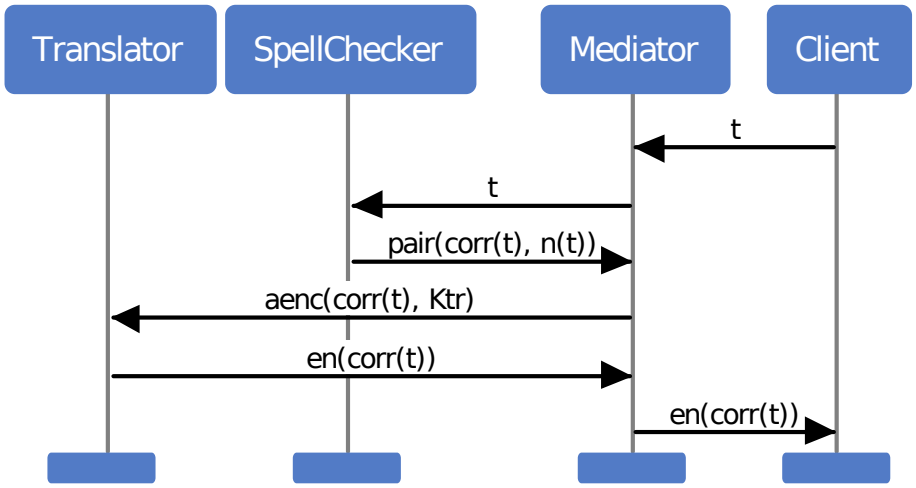


Figure 4.3: Solution for the orchestration problem example

Summing up

Idea is to build and solve a well-formed constraint system.

Now we explain how the constraint system displayed in Figure 4.2 is built. First the Mediator receives t from the Client, i.e. his knowledge is increased with the received message and thus becomes $\{K_{tr}, t\}$. Second, in order to send a request to SpellChecker, the Mediator must deduce some T, a message to be accepted by the Spellchecker, from his knowledge (Constraint 4.1). Third, when he obtains the response, he adds it to his knowledge and tries to build a request acceptable by the Translator, i.e. that matches the expected pattern (Constraint 4.2). Finally, when he receives the response from the Translator, he tries to deduce the response for the Client (Constraint 4.3).

Note that the constraint systems built by this procedure are well-formed, that is the knowledge of the unique Mediator is increasing, and each variable appears first in the right-hand side of some constraint. As we mentioned in Part I, lots of work was done on the resolution of the constraints of this type. Moreover, since this type of constraints are used for analyzing security protocols, the technique has already been implemented. This implementation is reused in AVANTSSAR Orchestrator, a tool which is able to solve the orchestration problems like in the presented example. We will sketch this tool in § 4.5.

The unique solution of the constraint system depicted in Figure 4.2 is a substitution $\{T \mapsto t; M \mapsto corr(t)\}$, which together with the guessed order can be interpreted as follows (see Figure 4.3): first Mediator sends to the SpellChecker t (text received from the Client), then he receives a response: $\text{pair}\,(corr(t), n(t))$. As the Mediator can decompose a pair he extracts the needed first part ($corr(t)$) and encrypt it with K_{tr}. The result is sent to the Translator that replies with $en(corr(t))$, the message expected by the Client. Thus, Mediator can forward it and complete.

We have demonstrated some abilities of the Mediator. In this simple example it *decomposed* a concatenated message to throw away a part that is not needed (here, number of corrections done) and *encrypted* a message with the necessary public key in order to adapt it in such way that the resulting message will be accepted by the service, since the service's policy is satisfied.

In the next example we show how *several* mediators may collaborate. In order to solve the analogous problem in these settings, a technique for resolution of well-formed constraints is not enough. Since we have to take into account knowledges of several different composers working in parallel, the constraint systems obtained in this case will probably violate the knowledge monotonicity property, and thus will not be well-formed.

4.2.2 Distributed orchestration example

For the distributed orchestration we will consider multiple cooperating mediators, called *partners*. We distinguish one of them (still called Mediator) who communicates with the client, while all others do not. The partners are free to invoke available services they can access, but the cooperation between them is restricted by communication patterns and *non-disclosure* policy conditioned by inter-organizational relationships (no sensitive data must be propagated to other organizations).

Suppose that the available services described in the previous example are not free and can serve only registered users (for the reason of simplicity we suppose that the Translator and the SpellChecker have a unique registered user each). Moreover, the Mediator has no credential to log in and use the Translator service, but it has an account for the SpellChecker (note that both Translator and SpellChecker are still reachable).

Fortunately, there is a partner who has an account for the Translator and can help to satisfy the client requests (see Figure 4.4), but does not want to reveal his credentials to the Mediator.

In this setting, we extend the specification of the Translator in a way that it also requires login and password to answer a request:

$$?\,\mathrm{aenc}\left(\mathrm{pair}\left(\mathrm{pair}\left(usr_{tr}, pwd_{tr}\right), M\right), K_{tr}\right).!en(M),$$

where usr_{tr} is a login of the registered user and pwd_{tr} is the corresponding password; K_{tr} is a public key of the Translator.

The same we do with another service: the corresponding specification of the SpellChecker becomes:

$$?\,\mathrm{aenc}\left(\mathrm{pair}\left(\mathrm{pair}\left(usr_{sc}, pwd_{sc}\right), T\right), K_{sc}\right).\\ !\,\mathrm{pair}\left(corr(T), n(T)\right),$$

where usr_{sc} is a login of the registered user and pwd_{sc} is the corresponding password; K_{sc} is a public key of the SpellChecker.

We suppose that Mediator and Partner share a symmetric key k and all communications between them should be encrypted with the shared key.

The problem is, as was sketched before, to find a feasible communication scenario between all the parties, such that *(i)* all requests of the Client are satisfied and *(ii)* no partner can extract a sensitive data of another partner from a message received from the latter. In our example we can consider pwd_{sc} and pwd_{tr} as sensitive for the Mediator and Partner correspondingly. Later, in § 4.3 we will present a *non-disclosure condition* which is sufficient to ensure *(ii)*.

Solution idea

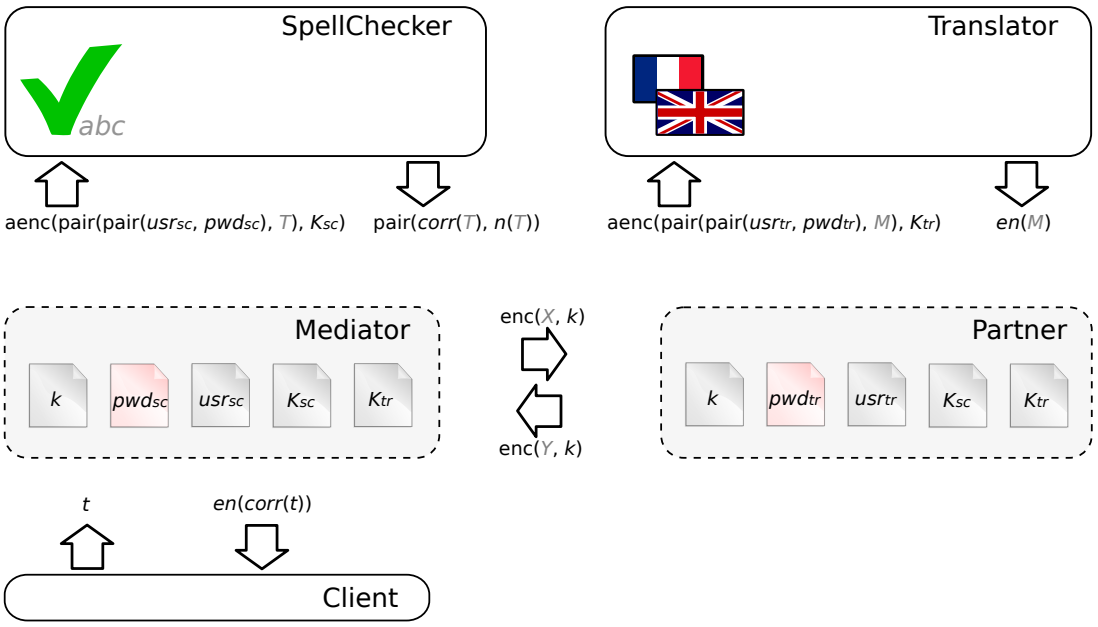


Figure 4.4: Illustration for the distributed orchestration problem example

Summing up

The idea is to build and solve a general constraint system.

We can solve this problem if the number of *interactions* (i.e. invocations or send/receive pairs) is bounded. Suppose for simplicity that we can use every available service at most once, and that we allow one round of communication between the Mediator and the Partner (Mediator sends request and the Partner replies).

Again, as in previous example, we have to choose an interleaving of Client's and Partners' actions and invocations of available services (note that the number of all possible interleavings is finite). Assume the selected interleaving is:

1. Client → Mediator
2. Mediator ↔ SpellChecker
3. Mediator → Partner
4. Partner ↔ Translator
5. Partner → Mediator
6. Mediator → Client

where $A \to B$ stands for "A sends and B receives", and $A \leftrightarrow B$ stands for "B is invoked by A". The corresponding constraint system, constructed in the similar way as in the previous example, is depicted in Figure 4.5.

We give a short explanation of every part (LHS and RHS) of every constraint in Figure 4.5:

- Constraint (4.4):
  - $\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t\}$ is a knowledge of the Mediator after he received a message t from the Client.

$$\left\{\begin{array}{ll}
\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t\} \triangleright & \text{aenc}\left(\text{pair}\left(\text{pair}\left(usr_{sc}, pwd_{sc}\right), T\right), K_{sc}\right) \quad (4.4)\\
\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}\left(corr(T), n(T)\right)\} \triangleright & \text{enc}\left(X, k\right) \quad (4.5)\\
\{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}\left(X, k\right)\} \triangleright & \text{aenc}\left(\text{pair}\left(\text{pair}\left(usr_{tr}, pwd_{tr}\right), M\right), K_{tr}\right) \quad (4.6)\\
\{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}\left(X, k\right), en(M)\} \triangleright & \text{enc}\left(Y, k\right) \quad (4.7)\\
\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}\left(corr(T), n(T)\right), \text{enc}\left(Y, k\right)\} \triangleright & en(corr(t)) \quad (4.8)
\end{array}\right.$$

with T, M, X, Y — variables.

Figure 4.5: A constraint system describing a possible distributed orchestration.

  - $\text{aenc}\left(\text{pair}\left(\text{pair}\left(usr_{sc}, pwd_{sc}\right), T\right), K_{sc}\right)$ is the only message pattern that can be accepted by the SpellChecker. Mediator has to build some message compliant with this pattern in order to invoke it.

- Constraint (4.5):
  - $\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}\left(corr(T), n(T)\right)\}$ is a knowledge of the Mediator when he received a reply from the SpellChecker.
  - $\text{enc}\left(X, k\right)$. This message (pattern) the Mediator has to send to the Partner.
- Constraint (4.6):
  - $\{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}\left(X, k\right)\}$ — a knowledge of the Partner after he received a message from the Mediator.
  - $\text{aenc}\left(\text{pair}\left(\text{pair}\left(usr_{tr}, pwd_{tr}\right), M\right), K_{tr}\right)$ — the only message pattern that can be accepted by the Translator. Partner must build some message conforming to this pattern in order to invoke it.
- Constraint (4.7):
  - $\{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}\left(X, k\right), en(M)\}$ — a knowledge of the Partner after the Translator replied him.
  - $\text{enc}\left(Y, k\right)$. The Partner must send a message conforming with this pattern to the Mediator.
- Constraint (4.8):
  - $\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}\left(corr(T), n(T)\right), \text{enc}\left(Y, k\right)\}$ — a knowledge of the Mediator after he received a message from the Partner.
  - $en(corr(t))$ — a message expected by the Client that the Mediator must finally construct.

We can see that the obtained constraint system is not well-formed, and thus, the existing fore-mentioned solving methods (except [Maz06]) are not applicable. Trying to answer a question of an anonymous reviewer of the related paper submission, the obtained system of constraints is not well-formed because the property called knowledge monotonicity does not hold: we cannot order the constraints in such way that the LHS of the every next constraint of the system includes the LHS of the previous one. This is due to the fact that there are multiple entities (partners) whose knowledge we must take into account, while in the case of non-distributed orchestration

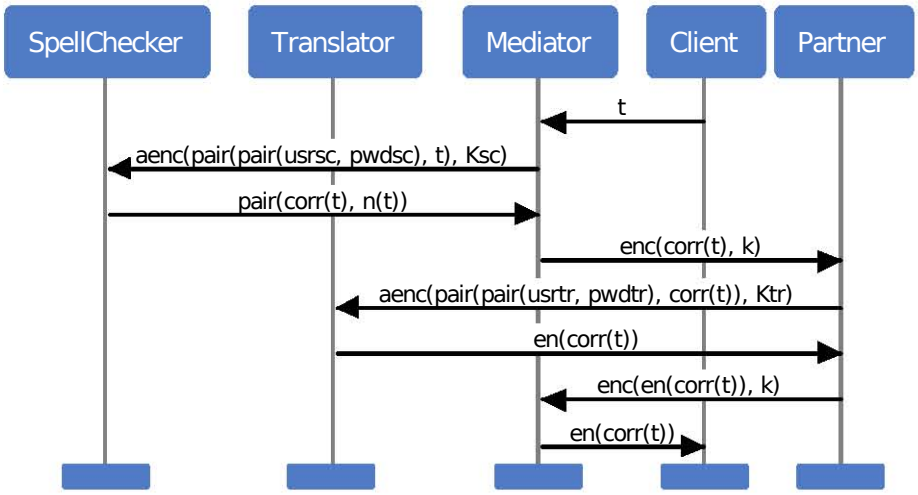


Figure 4.6: Solution for the distributed orchestration problem example

as well as in the case of protocol insecurity problem with Dolev-Yao intruder there exists the only entity whose knowledge is expressed in LHS of the constraints. For the orchestration it is the knowledge of the Mediator, and for the protocols, it is the knowledge of the intruder.

Another remark: we cannot treat the constraint system in a modular manner, i.e. consider separately a system of constraints for each partner, since the obtained constraint systems are not independent. For the shown example a constraint subsystem for the Mediator contains a message that should be received from the Partner. Moreover, this value is not ground and may vary depending on the solution for the constraint subsystem describing the communication of the Partner. Note that the latter in its turn also depend on a message received from the Mediator.

Thus, if we do not take into account the fact that the constraint subsystems are dependent, we may solve them separately, but we will be probably unable to join solutions, since they can be "incompatible", i.e. a solution for one subsystem implies one value for some variable, while a solution for another may imply another value for the same variable. Moreover, these subsystems are even not well-formed, since in general they will not satisfy the variable origination property: for example, the LHS of the first constraint may contain a non-ground term (in our case, for all the constraints relevant to the Partner, the Partner's knowledge, i.e. LHS, contains $\mathrm{enc}\,(X, k)$, and thus X does not appear in right-hand sides of "previous" constraints of the subsystem describing the Partner's steps).

One of the possible solutions that can be obtained using technique of Chapter 2, of this constraint system is

$$\sigma = \{T \mapsto t, M \mapsto corr(t), X \mapsto corr(t), Y \mapsto en(corr(t))\}\,.$$

Mediator gets text t from the Client, sends it encrypted together with his login data to the SpellChecker, then from the reply he extracts the corrected version $corr(t)$ of the text, sends it to the Partner, who concatenates it together with his translator login/password and sends it encrypted to the Translator. Partner forwards the obtained response to the Mediator who returns it to the Client (see Figure 4.6).

Note that neither $X\sigma$ nor $Y\sigma$, that is the messages sent between Mediator and Partner, contain any password pwd_{tr}, pwd_{sc}. This means that in this solution a sensitive information was not leaked from the Mediator to Partner or backward. We want to recall that we want to prevent the direct leakage of sensitive data from one partner to another, while we allow the

messages sent to the available services to contain some sensitive data (in our example, pwd_{sc} is contained in the message sent by the Mediator to the SpellChecker, but is not contained in the message sent by the Mediator to the Partner)

One can imagine another solution (that requires another interleaving of interactions), where Partner sends to Mediator his login and password for the Translator service, and then the Mediator do everything alone: receives message from the Cilent, invokes SpellChecker, invokes Translator and then replies to the Client. In this case, the inter-organizational policy is violated, although the Client's request is satisfied. In the next section we present a formal condition that is sufficient to avoid disclosure of sensitive data. Moreover, this formal condition admits an automatic procedure for finding communications that satisfies it.

4.3 Distributed orchestration model

In this section we introduce a model for the distributed orchestration. Note that the standard (non-distributed) orchestration is the special case with the unique mediator. Informally, the distributed orchestration problem is stated as follows: given a community of the available services, a Client, a set of partners each with some initial knowledge, a set of communication channels between the partners and the accessibility of the available services to partners, we have to find a feasible communication scenario between partners, available services and the Client, such that the Client reaches his final state (i.e. could execute all his actions). We consider the accessibility of the services to partners in order to pass easier from distributed orchestration to choreography if needed, since in choreography every partner should be able to access only its own available service, while the access to other services should be done via the corresponding attached partners.

In this communication scenario the messages circulated amongst partners have to be sent on channels of some predefined set and they must be compatible with message patterns defined per channel. Moreover, some non-disclosure policies imposed on the communications are specified as a set of sensitive atomic data per partner, which should not be extractable from messages sent to other partners.

To ensure the latter we will consider a stronger property: we *don't allow any occurrence of sensitive data as a subterm of these messages*. Indeed, as deduction rules in Table 2.4 (page 51) do not produce new atoms, a partner is unable to extract any sensitive data from a message that does not contain it. In this way the partners are guaranteed not to reveal directly their confidential information to other ones, but this information still can be used to invoke the available services. Meanwhile, one can see that this restriction is too strong and it can filter out some solutions that could work. For example, if a sensitive data k is used as a key to encrypt some message m, then this encrypted message $\mathrm{enc}(m, k)$ can be sent to the partner, since he will be unable to deduce k from $\mathrm{enc}(m, k)$. However, as k is present as a subterm in $\mathrm{enc}(m, k)$, this communication will not satisfy our stronger property.

Note that in order to directly solve the problem with non-disclosure policies, one should consider more complex techniques that are able to cope with satisfiability of constraint systems that includes also a negation: $E \not\vdash t$ (term t should *not* be deducible from set of terms E), where ground substitution σ is a solution of such a constraint, iff $\ulcorner t\sigma \urcorner \notin \mathrm{Der}(\ulcorner E\sigma \urcorner)$. In this way we can express properly the fact that some sensitive data is not deducible from the final knowledge of a Partner. Unfortunately we are not aware of any such results.

4.3.1 Distributed orchestration problem input

In this section we formalize the input of the considered problem.

We assume we are given:

- A set of available services $\mathbb{S} = \{S_1, \ldots, S_n\}$.
  An available service S_i is represented by its name and a normal (for the sake of simplicity) strand, i.e. $S_i = \langle i, A_i \rangle$, where $A_i = ?r_1^i.!s_1^i. \ldots . ?r_{e_i}^i.!s_{e_i}^i$.

- A client C.
  We can think of the client as a stand-alone available service $\langle 0, A_0 \rangle$, but A_0 is a strand which is not necessarily normal.

- A set of partners $\mathbb{P} = \{P_1, \ldots, P_k\}$ (P_1 is a Mediator) and for each partner P_i, a set of sensitive atoms N_i that he does not want to share with partners. Partner P_i is then represented by its name i, his current knowledge K_i and a set of sensitive atomic values $N_i \subseteq \mathrm{Sub}(K_i)$, i.e. $P_i = \langle i, K_i, N_i \rangle$.

- A set of communication channels $\mathbb{C} = \{C_1, \ldots, C_u\}$ between partners.
  Communication channel $C \in \mathbb{C}$ is a tuple $i \xrightarrow{p} j$, where i and j are names of partners P_i and P_j correspondingly and all messages sent through this channel from P_i to P_j must match pattern p. We assume that pattern p does not contain sensitive atoms as subterms, i.e. $\mathrm{Sub}(p) \cap N_i = \varnothing$.

- A set of accessible services per each partner $AS = \left\{\langle p_i, s_i \rangle_{i=1,\ldots,v}\right\}$.
  Partner P_{p_i} can access available service S_{s_i} iff $\langle p_i, s_i \rangle \in AS$.

- An upper bound on the number of interactions m.
  We allow at most m communication events, like invocation of an available service, transmission of a message from a partner to another, transmission of a message from Client to Mediator or in the opposite way.

The generic input is illustrated in Figure 4.7 (the upper bound m is not shown).

As we mentioned before, the distributed orchestration have two special cases: orchestration and choreography.

The input for the former is illustrated in Figure 4.8: There is only one Mediator P_1 (and no other partners), thus, $\mathbb{C} = \varnothing$ and no need of N_1. Moreover, all services are accessible by the Mediator: $AS = \left\{\langle 1, i \rangle_{i=1,\ldots,n}\right\}$.

For the latter the input is depicted in Figure 4.9: there is one Partner per available service, i.e. $k = n + 1$, and this service is accessible only for the corresponding Partner, i.e. $AS = \{\langle i + 1, i \rangle\}_{i=1,\ldots,n}$.

We assume that the sets of variables used to describe the actions of each available service (and in the Client) are pairwise disjoint, i.e. $\mathrm{Vars}(S_i) \cap \mathrm{Vars}(S_j) = \varnothing$, if $i \neq j$ and for all i, $\mathrm{Vars}(S_i) \cap \mathrm{Vars}(C) = \varnothing$.

4.3.2 Execution model

We define a *non-disclosure condition* (or *non-disclosure policy*) according to what we have already announced before: a sensitive atom of a partner never occurs as a subterm of a message emitted by him. We will impose this policy only on messages emitted by one partner to another, while the communication with available services is free from this condition.

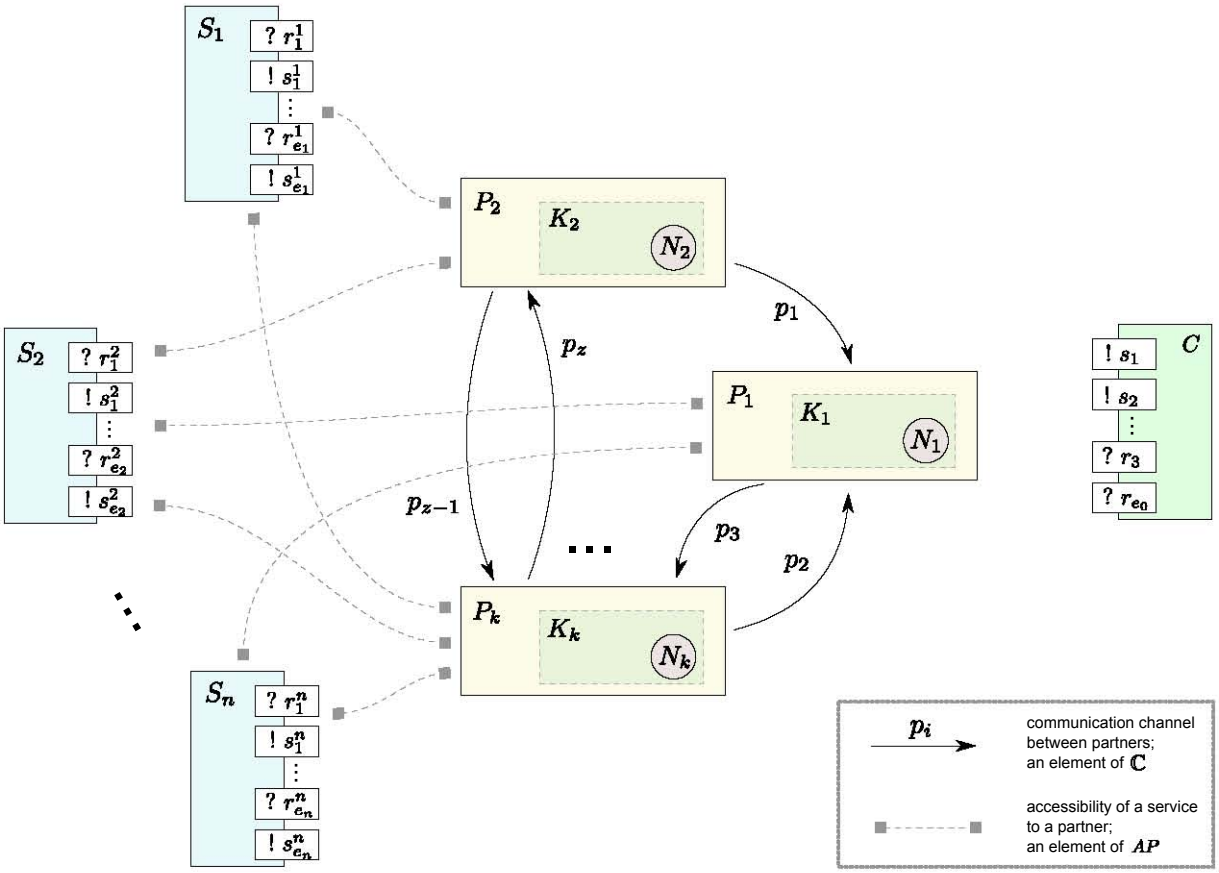


Figure 4.7: Generic input example for distributed orchestration problem

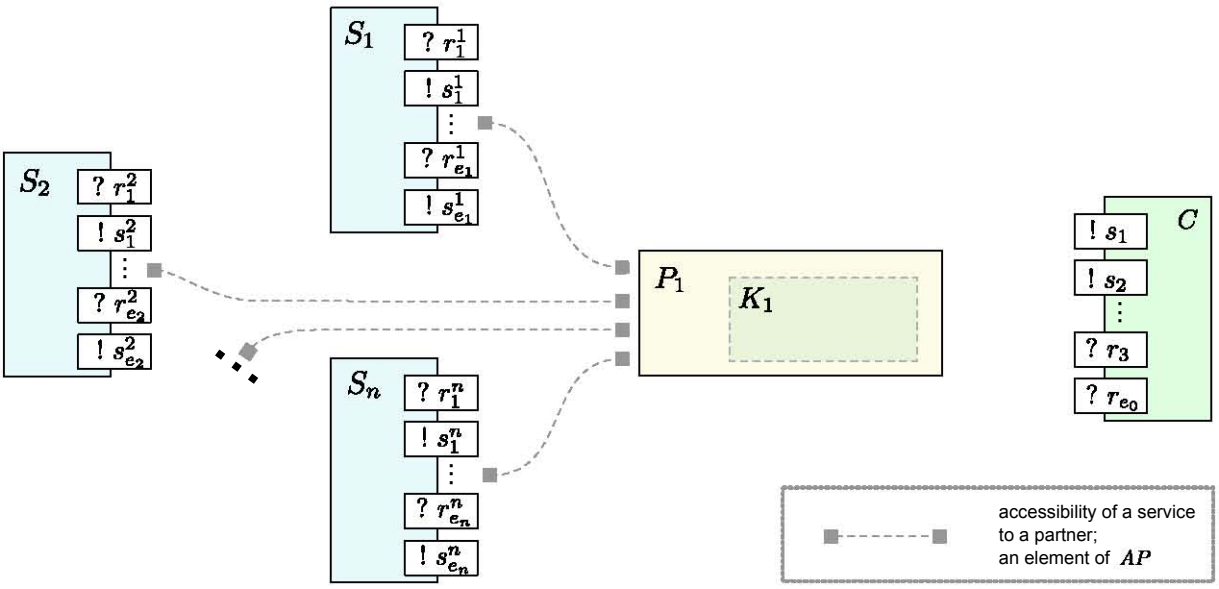


Figure 4.8: A special case of input: orchestration problem

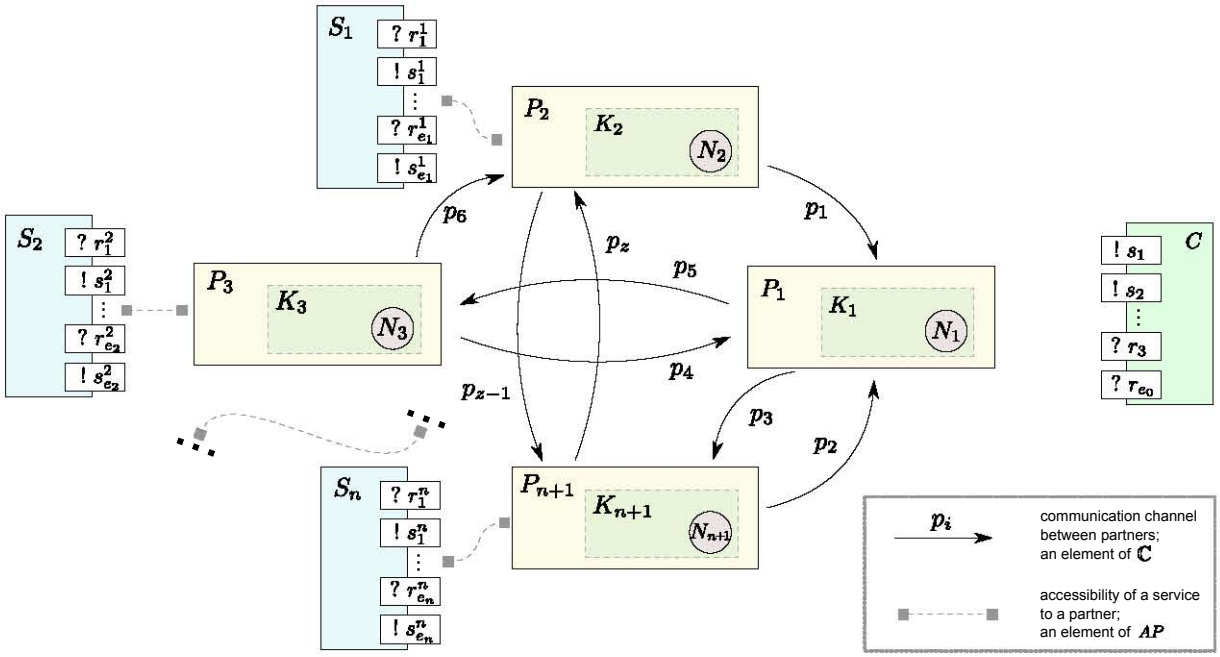


Figure 4.9: A special case of input: choreography problem

Definition 4.3.1 A *non-disclosure condition* $\mathcal{H}$ is a set of equations $\left\{\mathrm{Sub}\left(m_i\right) \cap N_{j_i} \stackrel{?}{=} \varnothing\right\}_{i=1,\ldots,l}$, where m_i is a term and N_{j_i} is a set of atoms. We say that a ground substitution σ is a solution of (or satisfies) $\mathcal{H}$ iff for all $i = 1, \ldots, l$ an equality $\mathrm{Sub}\left(m_i\sigma\right) \cap N_{j_i} = \varnothing$ holds. We call $\mathcal{H}$ *satisfiable* if it admits a solution.

Now we can present a *configuration* that reflects a global state the distributed orchestration.

Definition 4.3.2 A *configuration* is a tuple $\langle \{S_1, S_2, \ldots, S_n\}, C, \{P_1, \ldots, P_k\}, \mathcal{S}, \mathcal{H} \rangle$, i.e. a set of current states of available services, a state of the client, a set of states of the partners, a constraint system and a non-disclosure condition to be satisfied.

We define a set of transitions in Figure 4.10 that allow us to evolve from one configuration to another.

Transition 4.9 expresses that Partner $P_i = \langle i, K_i, N_i \rangle$ can invoke available service $S_j = \langle j, A_j \rangle$, iff it is accessible and he is able to build a message (ground term) that is compatible with the expected pattern. The reply of S_j will become a part of the partner's knowledge. Similarly for the message exchange of the Mediator P_1 and the Client C, except that the Client can initiate a sending; here we have two transitions: one for sending and one for receiving (Transitions 4.10, 4.11). A partner $P_i = \langle i, K_i, N_i \rangle$ can send a message to a partner P_j, iff there exists a channel $C_{ij} = i \stackrel{p}{\rightharpoonup} j \in \mathbb{C}$ between them such that partner P_i can build a message conforming with pattern p and this message will not contain sensitive data from N_i as a subterm (Transition 4.12). In the last transition we used refresh$(\cdot)$ function that replaces all variable names in a given term with fresh ones. It is done in order to make messages sent by the same channel independent. Note that in this setting services cannot be reused a second time, but we are free to add several instances for services into the problem input.

One can see that the configuration contains some conditions to be satisfied: a constraint system and a non-disclosure condition. Thus, passing from one configuration containing conditions that can be or already satisfied to another one using a transition may add some new

$$\frac{\left\langle \left\{ \left\langle j, ?r.!s.A'_j \right\rangle \right\} \cup S', C, \{\langle i, K_i, N_i \rangle\} \cup P', \mathcal{S}, \mathcal{H} \right\rangle \qquad [\text{if } \langle i, j \rangle \in AS]}{\left\langle \left\{ \left\langle j, A'_j \right\rangle \right\} \cup S', C, \{\langle i, K_i \cup \{s\}, N_i \rangle\} \cup P', \mathcal{S} \cup \{K_i \triangleright r\}, \mathcal{H} \right\rangle} \tag{4.9}$$

$$\frac{\langle S, \langle 0, !s.A' \rangle, \{\langle 1, K_1, N_1 \rangle\} \cup P', \mathcal{S}, \mathcal{H} \rangle}{\langle S, \langle 0, A' \rangle, \{\langle 1, K_1 \cup \{s\}, N_1 \rangle\} \cup P', \mathcal{S}, \mathcal{H} \rangle} \tag{4.10}$$

$$\frac{\langle S, \langle 0, ?r.A' \rangle, \{\langle 1, K_1, N_1 \rangle\} \cup P', \mathcal{S}, \mathcal{H} \rangle}{\langle S, \langle 0, A' \rangle, \{\langle 1, K_1, N_1 \rangle\} \cup P', \mathcal{S} \cup \{K_1 \triangleright r\}, \mathcal{H} \rangle} \tag{4.11}$$

$$\frac{\langle S, C, \{\langle i, K_i, N_i \rangle, \langle j, K_j, N_j \rangle\} \cup P', \mathcal{S}, \mathcal{H} \rangle \qquad [\text{if } i \xrightarrow{p} j \in \mathbb{C};\ q = \text{refresh}(p)]}{\left\langle S, C, \{\langle i, K_i, N_i \rangle, \langle j, K_j \cup \{q\}, N_j \rangle\} \cup P', \mathcal{S} \cup \{K_i \triangleright q\}, \mathcal{H} \cup \{\text{Sub}(q) \cap N_i \stackrel{?}{=} \varnothing\} \right\rangle} \tag{4.12}$$

where refresh(t) is a term obtained by replacing all variables of t with fresh ones.

Figure 4.10: Transition system

conditions that possibly cannot be satisfied or that are not compatible with ones of the initial configuration.

That is why we introduce two notions:

- *symbolic execution*, where we do not care about feasibility of a step, but simply apply syntactically transitions of Figure 4.10 and postpone this problem in a form of conditions to satisfy, and
- *execution*, where we require that conditions are satisfied.

Definition 4.3.3 A sequence of length l of configurations starting with initial one $\langle \mathbb{S}, C, \mathbb{P}, \varnothing, \varnothing \rangle$ and obtained by applying transitions from Figure 4.10 is called *symbolic execution* SE of length l.

Definition 4.3.4 An *execution* E is a pair $\langle SE, \sigma \rangle$ of a symbolic execution SE and ground substitution σ, such that for the last configuration $\left\langle \mathbb{S}^l, C^l, \mathbb{P}^l, \mathcal{S}^l, \mathcal{H}^l \right\rangle$ of SE the $\mathcal{S}^l$ and $\mathcal{H}^l$ are both satisfied by σ.

Example 26 We refer to an example described in § 4.2.2. First we formalize the problem input and then show the symbolic execution that leads to the presented constraint system and corresponding non-disclosure condition. We denote SpellChecker as

$$S_1 = \langle 1, ?\,\text{aenc}\,(\text{pair}\,(\text{pair}\,(usr_{sc}, pwd_{sc}), T), K_{sc}).!\,\text{pair}\,(corr(T), n(T)) \rangle,$$

Translator as

$$S_2 = \langle 2, ?\,\text{aenc}\,(\text{pair}\,(\text{pair}\,(usr_{tr}, pwd_{tr}), M), K_{tr}).!en(M) \rangle,$$

Client as

$$C = \langle 0, !t.?en(corr(t)) \rangle,$$

Mediator as

$$P_1 = \langle 1, \{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k\}, \{pwd_{sc}\} \rangle,$$

Partner as

$$P_2 = \langle 2, \{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k\}, \{pwd_{tr}\} \rangle.$$

The communication channels are: $\mathbb{C} = \left\{1 \overset{\text{enc}(X,k)}{\rightharpoonup} 2, 2 \overset{\text{enc}(P,k)}{\rightharpoonup} 1\right\}$. We also considered full service accessibility is $AS = \{\langle 1,1\rangle, \langle 1,2\rangle, \langle 2,1\rangle, \langle 2,2\rangle\}$. Number of interactions is 6. We start from initial configuration $\langle \{S_1, S_2\}, C, \{P_1, P_2\}, \varnothing, \varnothing\rangle$.

$$\langle\ \{\ \langle 1, ?\,\text{aenc}\,(\text{pair}\,(\text{pair}\,(usr_{sc}, pwd_{sc}), T), K_{sc}).!\,\text{pair}\,(corr(T), n(T))\rangle,$$
$$\langle 2, ?\,\text{aenc}\,(\text{pair}\,(\text{pair}\,(usr_{tr}, pwd_{tr}), M), K_{tr}).!en(M)\rangle\ \},$$
$$\langle 0, !t.?en(corr(t))\rangle,$$
$$\{\ \langle 1, \{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k\}, \{pwd_{sc}\}\rangle,$$
$$\langle 2, \{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k\}, \{pwd_{tr}\}\rangle\ \},$$
$$\varnothing,$$
$$\varnothing\ \rangle$$

$\Downarrow$ (Transition 4.11)

$$\langle\ \{\ \langle 1, ?\,\text{aenc}\,(\text{pair}\,(\text{pair}\,(usr_{sc}, pwd_{sc}), T), K_{sc}).!\,\text{pair}\,(corr(T), n(T))\rangle,$$
$$\langle 2, ?\,\text{aenc}\,(\text{pair}\,(\text{pair}\,(usr_{tr}, pwd_{tr}), M), K_{tr}).!en(M)\rangle\ \},$$
$$\langle 0, ?en(corr(t))\rangle,$$
$$\{\ \langle 1, \{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t\}, \{pwd_{sc}\}\rangle,$$
$$\langle 2, \{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k\}, \{pwd_{tr}\}\rangle\ \},$$
$$\varnothing,$$
$$\varnothing\ \rangle$$

$\Downarrow$ (Transition 4.9, $i = 1, j = 1$)

$$\langle\ \{\ \langle 1, \varnothing\rangle,$$
$$\langle 2, ?\,\text{aenc}\,(\text{pair}\,(\text{pair}\,(usr_{tr}, pwd_{tr}), M), K_{tr}).!en(M)\rangle\ \},$$
$$\langle 0, ?en(corr(t))\rangle,$$
$$\{\ \langle 1, \{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}\,(corr(T), n(T))\}, \{pwd_{sc}\}\rangle,$$
$$\langle 2, \{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k\}, \{pwd_{tr}\}\rangle\ \},$$
$$\{\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t\} \triangleright \text{aenc}\,(\text{pair}\,(\text{pair}\,(usr_{sc}, pwd_{sc}), T), K_{sc})\},$$
$$\varnothing\ \rangle$$

$\Downarrow$ (Transition 4.12, $i = 1, j = 2$, $1 \overset{\text{enc}(X,k)}{\rightharpoonup} 2 \in \mathbb{C}, \text{refresh}(\text{enc}\,(X, k)) = \text{enc}\,(X', k)$)

$$\langle\ \{\ \langle 1, \varnothing\rangle,$$
$$\langle 2, ?\,\text{aenc}\,(\text{pair}\,(\text{pair}\,(usr_{tr}, pwd_{tr}), M), K_{tr}).!en(M)\rangle\ \},$$
$$\langle 0, ?en(corr(t))\rangle,$$
$$\{\ \langle 1, \{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}\,(corr(T), n(T))\}, \{pwd_{sc}\}\rangle,$$
$$\langle 2, \{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}\,(X', k)\}, \{pwd_{tr}\}\rangle\ \},$$
$$\{\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t\} \triangleright \text{aenc}\,(\text{pair}\,(\text{pair}\,(usr_{sc}, pwd_{sc}), T), K_{sc}),$$
$$\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}\,(corr(T), n(T))\} \triangleright \text{enc}\,(X', k)\},$$
$$\{\text{Sub}\,(\text{enc}\,(X', k)) \cap \{pwd_{tr}\} \overset{?}{=} \varnothing\}\ \rangle$$

$\Downarrow$ (Transition 4.9, $i = 2, j = 2$)

$$\langle\ \{\ \langle 1, \varnothing\rangle, \langle 2, \varnothing\rangle\},$$
$$\langle 0, ?en(corr(t))\rangle,$$
$$\{\ \langle 1, \{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}\,(corr(T), n(T))\}, \{pwd_{sc}\}\rangle,$$
$$\langle 2, \{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}\,(X', k), en(M)\}, \{pwd_{tr}\}\rangle\ \},$$
$$\{\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t\} \triangleright \text{aenc}\,(\text{pair}\,(\text{pair}\,(usr_{sc}, pwd_{sc}), T), K_{sc}),$$
$$\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}\,(corr(T), n(T))\} \triangleright \text{enc}\,(X', k),$$
$$\{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}\,(X', k)\} \triangleright \text{aenc}\,(\text{pair}\,(\text{pair}\,(usr_{tr}, pwd_{tr}), M), K_{tr})\ \},$$
$$\{\text{Sub}\,(\text{enc}\,(X', k)) \cap \{pwd_{sc}\} \overset{?}{=} \varnothing\}\ \rangle$$

$$\Downarrow \text{(Transition 4.12, } i=2, j=1,\ 2 \overset{\text{enc}(Y,k)}{\rightarrow} 1 \in \mathbb{C}, \text{refresh}(\text{enc}(Y,k)) = \text{enc}(Y',k))$$

$$\begin{array}{c}
\langle\ \{\ \langle 1,\varnothing\rangle, \langle 2,\varnothing\rangle\},\\
\hdashline
\langle 0, ?en(corr(t))\rangle,\\
\hdashline
\{\ \langle 1, \{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}(corr(T), n(T)), \text{enc}(Y', k)\}, \{pwd_{sc}\}\rangle,\\
\langle 2, \{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}(X', k), en(M)\}, \{pwd_{tr}\}\rangle\ \},\\
\hdashline
\{\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t\} \triangleright \text{aenc}(\text{pair}(\text{pair}(usr_{sc}, pwd_{sc}), T), K_{sc}),\\
\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}(corr(T), n(T))\} \triangleright \text{enc}(X', k),\\
\{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}(X', k)\} \triangleright \text{aenc}(\text{pair}(\text{pair}(usr_{tr}, pwd_{tr}), M), K_{tr}),\\
\{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}(X', k), en(M)\} \triangleright \text{enc}(Y', k)\ \},\\
\hdashline
\{\text{Sub}(\text{enc}(X', k)) \cap \{pwd_{sc}\} \stackrel{?}{=} \varnothing,\\
\text{Sub}(\text{enc}(Y', k)) \cap \{pwd_{tr}\} \stackrel{?}{=} \varnothing\}\ \rangle
\end{array}$$

$$\Downarrow \text{(Transition 4.10)}$$

$$\begin{array}{c}
\langle\ \{\ \langle 1,\varnothing\rangle, \langle 2,\varnothing\rangle\},\\
\hdashline
\langle 0, \varnothing\rangle,\\
\hdashline
\{\ \langle 1, \{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}(corr(T), n(T)), \text{enc}(Y', k)\}, \{pwd_{sc}\}\rangle,\\
\langle 2, \{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}(X', k), en(M)\}, \{pwd_{tr}\}\rangle\ \},\\
\hdashline
\{\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t\} \triangleright \text{aenc}(\text{pair}(\text{pair}(usr_{sc}, pwd_{sc}), T), K_{sc}),\\
\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}(corr(T), n(T))\} \triangleright \text{enc}(X', k),\\
\{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}(X', k)\} \triangleright \text{aenc}(\text{pair}(\text{pair}(usr_{tr}, pwd_{tr}), M), K_{tr}),\\
\{K_{tr}, K_{sc}, usr_{tr}, pwd_{tr}, k, \text{enc}(X', k), en(M)\} \triangleright \text{enc}(Y', k),\\
\{K_{tr}, K_{sc}, usr_{sc}, pwd_{sc}, k, t, \text{pair}(corr(T), n(T)), \text{enc}(Y', k)\} \triangleright en(corr(t))\ \},\\
\hdashline
\{\text{Sub}(\text{enc}(X', k)) \cap \{pwd_{sc}\} \stackrel{?}{=} \varnothing,\\
\text{Sub}(\text{enc}(Y', k)) \cap \{pwd_{tr}\} \stackrel{?}{=} \varnothing\}\ \rangle
\end{array}$$

Note that in the last configuration the Client has no more actions to perform.

4.3.3 Problem statement

Given a problem input as described above (see § 4.3.1) is there an execution $E = \langle SE, \sigma\rangle$ of length $l \leq m$ such that the sequence of actions of the Client is empty at the end, i.e. the last configuration of SE is in form $\left\langle \mathbb{S}^l, \langle 0, \varnothing\rangle, \mathbb{P}^l, \mathcal{S}^l, \mathcal{H}^l \right\rangle$?

Example 27 If we consider the problem of Example 26 and denote the symbolic execution shown in this example as SE, then the execution $\langle SE, \sigma\rangle$, where $\sigma = \{T \mapsto t, M \mapsto corr(t), X' \mapsto corr(t), Y' \mapsto en(corr(t))\}$ is its solution. Note that this solution specifies the communication depicted in Figure 4.6.

We can see that an execution E defines a message flow, and thus, in particular, we can extract a sequence of actions performed by every Partner.

4.4 Solution approach

We will reduce the problem of distributed orchestration to the satisfiability of deducibility constraint system under non-disclosure of sensitive data condition (§ 4.4.1) and then refer to a decision procedure (§ 4.4.2). Taking into account the non-disclosure condition does not bring any problem and requires only a very minor change in the reasoning about the problem decidability described in Part I.

4.4.1 Reduction to deducibility constraints

We reduce the distributed orchestration problem to the satisfiability of a deducibility constraint system under non-disclosure of sensitive data condition and then discuss a decision procedure under the hypothesis of bounded number of interactions.

Since one can build finitely many different symbolic executions for a fixed problem input, we can guess a symbolic execution with its final configuration $\langle \mathbb{S}^l, C^l, \mathbb{P}^l, \mathcal{S}^l, \mathcal{H}^l \rangle$ where the Client has no more actions to perform (i.e. $C^l = \langle 0, \varnothing \rangle$). An example of such symbolic execution is presented in Example 26. Then, building the desired execution is equivalent to finding such a ground substitution σ that satisfies both $\mathcal{S}^l$ and $\mathcal{H}^l$.

4.4.2 Constraint system resolution

We refer to Part I for the technique for solving constraint systems within Dolev-Yao deduction system. Under non-restrictive assumption that in the problem input there exists at least one atomic value which is not sensitive to any of partners, and the assumptions stated in the problem input (§ 4.3.1) hold, we can easily adapt the mentioned technique in such way that the following theorem holds:

Theorem 4. *Satisfiability of deducibility constraint system within Dolev-Yao deduction system under non-disclosure condition is in NP.*

Proof idea. It is enough to precise the value of α in Definition 2.1.12. We must use an atom (whose existence we assumed) that is not sensitive to any partner. Therefore, if there exists a solution σ that satisfies constraint system $\mathcal{S}$ and non-disclosure condition $\mathcal{H}$, then $\pi(\mathrm{H}(\sigma))$ will also (i) satisfy $\mathcal{S}$ and (ii) satisfy $\mathcal{H}$, since by definition of $\pi(\mathrm{H}(\sigma))$, $(\mathrm{Sub}(\pi(H(x\sigma))) \setminus \mathrm{Sub}(x\sigma)) \cap \mathcal{A} \subseteq \{\alpha\}$. □

And thus,

Corollary 5. *The problem of distributed orchestration is decidable and NP-complete*[22].

Note also that having a desired execution $E = \langle SE, \sigma \rangle$, and thus a sequence of actions performed by the Partners as well as their initial knowledges, we can extract a *prudent implementation* of the Partners as services (see details in [CR10a]). This approach is skin-deep discussed in `Trace2ASLan` tool description, § 4.5.5 at p. 99.

4.5 AVANTSSAR Orchestrator

The non-distributed case of the orchestration (see the example from § 4.2.1, p. 75) has been implemented as *AVANTSSAR Orchestrator* [avab], a part of the *AVANTSSAR Validation Platform* [avaa]. Thanks to analogy with cryptographic protocol insecurity problem explained in § 4.5.4, we were able to reuse an existing tool based on resolution of well-formed deducibility constraint systems for solving the orchestration problem.

Moreover, the implementation is richer than the presented model since it allows more complex behavior of the available services and the client. The extension concerns the way how the Web Services are represented: instead of using a sequence of actions, in AVANTSSAR a more elaborated model (a transition system) which includes branching with complex conditions is used.

[22] See Chapter 3

Comparing to the existing tools (which are not a lot) that deal with automated Web Services orchestration, like [TPC+ro, CMS+09], to our knowledge, these tools abstract away the security policies attached to the services, while we consider them as an additional constraints.

Summing up

Mediator service = Goal Service

In the context of AVANTSSAR, the Mediator service is referred as *Goal service* (since it is a target, resulting service of the orchestration problem) and we will use this term in this section to keep the terminology consistent with the AVANTSSAR project.

We also consider a variation of the problem whereby an abstract characterization of the Goal service is given in place of the client. In this case, the orchestration problem amounts to finding a concrete implementation of the Goal service in such a way that this implementation would be "executable".

4.5.1 AVANTSSAR Validation Platform

The Orchestrator is part of the AVANTSSAR Platform [avaa], an automated tool-set for validating trust and security aspects of service-oriented architectures (see Figure 4.11). The overall objective of the platform is to generate from an orchestration problem a solution that meets some given security requirements (e.g., secrecy and authentication properties). The input language of the platform is ASLan (see § 4.5.2 for some details).

Summing up

The aim is to build an orchestration that meets desired security properties.

The AVANTSSAR platform offers the possibility to check whether an orchestrated service (e.g. generated by the Orchestrator tool) or a community of given services is vulnerable to an active Dolev-Yao intruder. For that the modeler has to integrate security properties to verify to the orchestration problem specification in ASLan format. The orchestration problem together with security properties to be validated come to the input of the Orchestrator that solves the orchestration problem by satisfying the functionality requirements[23] (e.g. ability of satisfy Client's requests). The Orchestrator outputs the solution (input enriched with the newly generated Goal (Mediator) service) by preserving the security properties to be validated also in ASLan format which comes to the input of the Validator. Automatic validation tools (CL-AtSe § 4.5.5, SATMC [ACC07, AC08, ACC+08] and OFMC [BMV05b, BMar, MV09, MVBar]) that form AVANTSSAR Validator will check whether the property is satisfied by the generated orchestration. Summing up, the Orchestrator solves the orchestration problem, and then the solution (with the properties to be validated) are transferred to the AVANTSSAR Validator.

If the specification meets the validation goals then the orchestration solution is considered as safe with regard to the user's requirements, otherwise a verification report including the violation proof is returned. In the latter case the Orchestrator is able to backtrack and try an alternative solution. This is illustrated in Figure 4.11 by the returning-arrow from the Validator to the Orchestrator.

To summarize the platform performs as follows:

1. It generates an orchestration (a Mediator);
2. It verifies security properties on it;
3. If the orchestration is vulnerable, it generates a new Mediator and jumps to Step 2. Otherwise the generated orchestration is considered as safe.

[23] Here we mean also taking into account the "local" security policies of the available services, but not the resistance to the external attacker, i.e. some "global" security properties

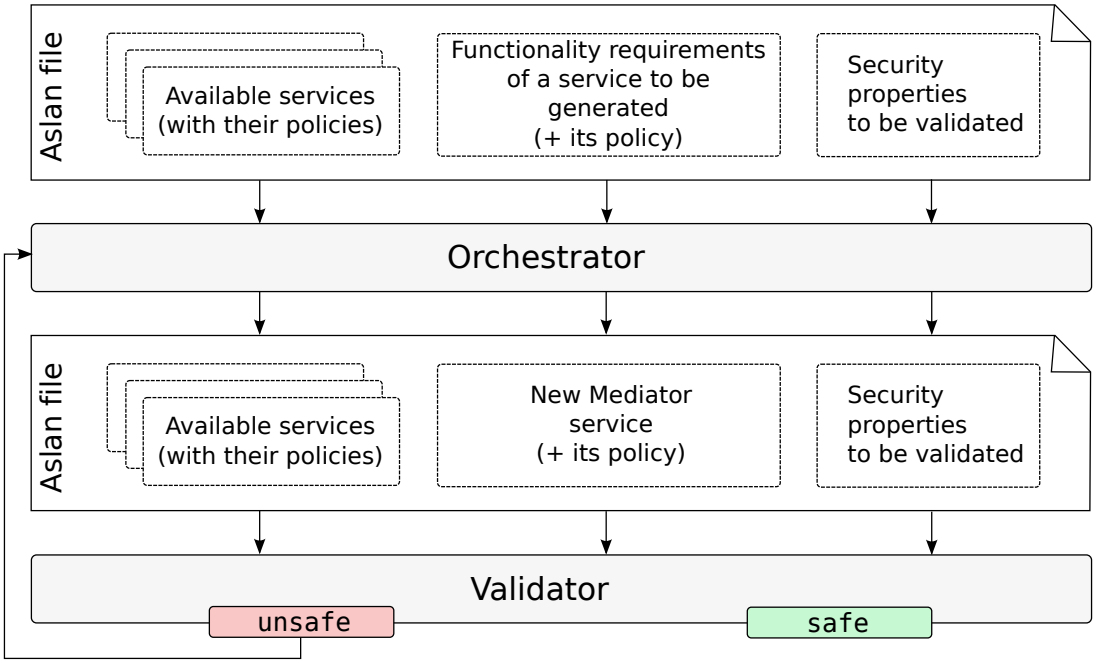


Figure 4.11: The AVANTSSAR Validation platform

4.5.2 Input Format

An input specification of an orchestration problem is an ASLan file with some reserved identifiers that have a special meaning. Here, we will only recall the main features of ASLan, pointing the reader to the AVANTSSAR project's Deliverable 2.1 (*Requirements for modeling and ASLan v.1* [AVA08]) and the updated Deliverable 2.3 (*ASLan final version with dynamic service and policy composition*, updated: [AVA11]), for more details on the language.

Then, we sketch how an orchestration problem should be presented in ASLan language.

ASLan

ASLan is defined by extending and refining the Intermediate Format IF [AVI03b], a specification language developed in the context of the FP5 project AVISPA. IF is an expressive language for specifying security protocols and their properties, based on set rewriting. Moreover, IF comes with mature tool support, namely the AVISPA Tool and all of its back-ends, which provide the basis for the back-ends of the AVANTSSAR Platform that we have been developing. ASLan extends IF with a number of important features so as to express diverse security policies, security goals, communication and intruder models at a suitable abstraction level, and thereby allow for the formal specification and analysis of complex services and service-oriented architectures. Most notable extensions are:

Horn Clauses. In ASLan, invariants of the system can be defined by a set of (definite) Horn clauses. They allow for the incorporation of authorization logics in specifications of services.

LTL goals. Some complex security properties can be specified in Linear Temporal Logic.

An ASLan file consists of several sections, among which:

Section Inits contains one or more initial states of the transition system. A state of a transition system is a set of variable-free *facts*.

Section Rules specifies the transitions of the transition system.

A *transition* is a rule containing two parts, a left-hand side (LHS) and right-hand side (RHS). The rule can fire in a state whenever its LHS holds in that state. Moreover, a transition can be labeled with a list of existentially quantified variables whose purpose is to introduce new constants representing fresh data (e.g. nonces).

Note that in ASLan an identifier starting from a lower-case symbol defines a constant or a function, while one starting from the uppercase — a variable.

Example 28 Sample transition.

```
step sampleTransition(BankAgent,Request,X):=
 state_BankingService(BankAgent, 1).
 iknows(Request)
 & equals(Request, pair(BankAgent,X))
 =>
 state_BankingService(BankAgent, 2).
 iknows(response)
```

where

`step` is a keyword used to define a new transition;

`sampleTransition` is a transition name;

`BankAgent,Request,X` are parameters of the transition;

`state_BankingService(BankAgent, 1)`, `iknows(Request)`, `state_BankingService(BankAgent, 2)`, `iknows(response)` are facts;

`equals(Request, pair(BankAgent,X))` is a condition;

`state_BankingService(BankAgent, 1).iknows(Request) & equals(Request, pair(BankAgent,X))` is the LHS of the transition;

`state_BankingService(BankAgent, 2).iknows(response)` is the RHS of the transition.

This transition represents the behavior of a banking service that receives a `Request` that must contain as a first part BankAgent and then reacts by replying with a `response` and moving to another state.

More precisely, the transition can be fired if there exist values `val1` and `val2` of variables `BankAgent` and `X` correspondingly such that `state_BankingService(val1, 1)` and `iknows(pair(val1,val2))` are in the current state. Note that we have substituted here `Request` with its value `pair(val1,val2)` to satisfy the equality condition `equals(Request, pair(BankAgent,X))`. The result of firing the transition is to replace the fact `state_BankingService(val, 1)` by the fact `state_BankingService(val, 2)` and add a new fact `iknows(response)`.

Message sending and receiving are specified using `iknows` facts: the `iknows` in the LHS of a transition stands for receiving a message, while in the RHS of a transition it stands for sending a message. Some correspondence between ASLan messages and formalism used in theoretical parts (see Table 2.1, page 28) is given in Table 4.2. The fact `iknows(pair(val1,val2))`

ASLan term	Corresponding term and/or Description
`scrypt(k,m)`	enc (m, k)
`crypt(k,m)`	aenc (m, k)
`pair(p,q)`	pair (p, q)
`inv(k)`	priv (k) for public key k, `inv(inv(k))= k`
`crypt(inv(k),m)`	pair $(m, \text{sig}(m, \text{priv}(k)))$ for public key k (if it is known by everyone)
`apply(f,m)`	apply (f, m)

Table 4.2: Correspondence between ASLan messages and formalism used so far

of Example 28 will not disappear from the current state if the transition is fired, because the predicate `iknows` is persistent: once a message is emitted, it becomes a part of the knowledge of the environment (i.e., of the network or of the intruder) and the environment does not "forget" it. In our case, where we are looking for the orchestration, the Mediator will be this central entity to whom all the emitted messages come and who stores them for reusing.

If the LHS of a transition holds in the current state, then it is assumed that the knowledge (represented by a set of ground facts) of the corresponding service is enough to build the messages stated in `iknows` in the RHS of the transition.

In order to specify service states, we use one predicate per service. By convention the predicate name starts with `state_` followed by the service name, e.g. `state_BankingService` from Example 28.

Section Goals contains security goals that can be defined as attack states (special final states of the transition system) or by means of LTL formulae.

Example 29 Sample attack state.

```
attack_state stateName(Msg):=
    fact1(Msg).
    fact2(Msg)
```

Here, attack state `stateName` is reached, if there exists a value `val` of variable `Msg` such that `fact1(val)` and `fact2(val)` are in the current state of the transition system.

Some attack states and LTL formulas (with reserved names) are dedicated to specify the orchestration problem (see the next sub-subsection "Specification of the orchestration problem").

Section HornClauses contains a finite set of Horn clauses (HC). The facts that can be inferred from this set of HC are not kept as a part of the current state of the transition system, but should be recalculated every time to test whether a given transition can fire. This can be useful for specifying some authorization logic which can be dynamic.

Two kinds of input specification are supported by the platform. In both cases, the available services are defined by their transition system.

The first option is to define a Client service, a service whose requests should all be satisfied. That is, the result of solving the orchestration problem is a service (Goal service, or Mediator) that is able to reply to every request of the Client. To this end it may use the available services. All communications of the Goal service should be reflected in the output.

The second option is to partially define a Goal service. One should only specify the part related to the communication with the putative client. As for the output, a new Goal service is issued, which extends the Goal service given in the input with the necessary communications with available services.

Note that all internal abilities outside of performing Dolev-Yao operations are supposed to be designed as available services.

Specification of the orchestration problem

To specify orchestration problems, we need to distinguish the key parts, such as the Goal service or Client service.

As a state of the Goal service we use

`state_OrchestrationGoal`(<list of parameters>);

similarly, we denote the state of the Client by the predicate:

`state_OrchestrationClient`(<list of parameters>).

A sample transition of the Client service is given in Example 30. Generally, a state of a service is represented by a predicate whose name starts with `state_` followed by a service name, e.g. `state_Service1`. (An example of an hypothetical banking service transition was given in Example 28.)

The orchestration problem is specified (besides the set of available services) with an LTL formula that must be satisfied by the computed orchestration. For example, to specify an orchestration where the emergency service is not invoked (the service stays in its initial state `state_Emergency(0)`) while the Client reaches the state `state_OrchestrationClient(A,99)` (for some `A`), we define an LTL formula:

```
goal orchestrationConstraint(A) :=
       and(G(state_emergency(0)), F(state_orchestrationClient(A,99)))
```

The `orchestrationConstraint` word is a reserved keyword to express that the goal is to be processed by the Orchestrator, in contrast with the other (security) goals that are to be processed by the Validator.

Another way to specify an orchestration problem is to define a special attack state using the keyword `orchestrationFinalState`. The orchestration problem is solved when this state is reached. Example 30 contains such a specification.

An aspect that affects the orchestration is the knowledge of the Goal service. In fact, the abilities of the Goal service directly depend on his knowledge: the credentials it possesses, the functions it can apply, etc. To define the initial knowledge, a Goal state predicate (`state_OrchestrationGoal`) is employed: the arguments placed in the initial state are considered as the initial knowledge of the Goal. The same holds for `state_OrchestrationClient` in the initial state for the Client.

We recall that the message sending and receiving are specified using `iknows` facts: the `iknows` in the LHS of a transition stands for receiving a message, while in the RHS of a transition it stands for sending a message.

The full list of conventions can be found in [AVA10].

Orchestration problem with a specified Client In the case where a Client is given as input, one must provide the specification of

- The available services;
- The Client service;
- The initial knowledge of the Goal service;
- `attack_state orchestrationFinalState` and/or `goal orchestrationConstraint`.

All communications defined in the Client service specification are considered to be established with the Goal service only.

For instance, to define the orchestration problem as a state to be reached, one can identify the final transition to be satisfied by the goal by writing `iknows(some_fresh_constant)` on the RHS of the Client's final transition and define an attack state named `orchestrationFinalState` as a one-fact state that contains `iknows(some_fresh_constant)` (see Example 30).

Example 30

```
        ....
        step step_5(Ag, Stp, Dummy_A):=
        state_OrchestrationClient(Ag, 5,Dummy_A).
        iknows(a)
    =>
        state_OrchestrationClient(Ag, 6, a).
        iknows(finish)

        ...
    attack_state orchestrationFinalState() :=
    iknows(finish)
```

Orchestration problem with a partially specified Goal service In the case where a Goal is specified as input, one must provide:

- The specification of available services;
- The partial specification of the Goal service, related to communication with the Client;
- The initial knowledge of the Goal service.

The specification of `goal orchestrationConstraint` is optional.
The attack state `orchestrationFinalState` will be ignored.

Note that by default the output will contain a specification of the putative client. To suppress this, it suffice to add the following line into the input specification:

```
% @orchestrator(no_client)
```

This meta information will be interpreted by the tool and the specification of the putative client will not appear in the output. It can be useful in some cases to facilitate the further validation of the orchestration result.

Services	Protocols
Available service/Client	Protocol role
Mediator	Intruder
Final state	Attack state

Table 4.3: WS Orchestration vs Protocol Analysis

4.5.3 Output Format

The output format for the Orchestrator is the standard ASLan language (e.g. [AVA11]), that is taken as input by the Validator. Given an ASLan file in input as described in section 4.5.2, the Orchestrator enriches the file with new rules concerning

- a *Client* service, when this is generated by the Orchestrator rather than specified by the modeler, that specifies the requests made by a client, and
- a *Goal* service, ensuring that the requests of a Client can be properly answered.

The introduced rules refer to the state fact of either the Client service or of the Goal service, of the forms

```
state_OrchestrationGoal(<list of parameters>);
state_OrchestrationClient(<list of parameters>);
```

respectively.

4.5.4 Approach overview

The general idea is to use the resemblance of the orchestration problem with the reachability problem used in cryptographic protocol domain, and the to reuse existing tools initially intended for solving protocols insecurity problem to solve the orchestration problem.

Summing up

Orchestration problem resembles a case of protocol insecurity problem. Thus, we can reuse existing tool.

The orchestration problem is stated as a reachability of a final state of the given transition system. The same approach can be used to define and solve some cases of the insecurity problem of cryptographic protocols. As the behavioral descriptions of the services are similar to the descriptions of *roles* in security protocols [CDL+99] and the Mediator capabilities are identical to Dolev-Yao intruder's ones, we will reuse a tool from security protocol verification with some preprocessing to tackle this problem. We give some equivalence between the orchestration problem and protocol insecurity problem in Table 4.3.

We represent the available services and the client service as protocol roles. The intruder with Dolev-Yao capabilities, who has full control over the network, will play the role of a Mediator: he tries to lead the given transition system from its initial state to a final accepting one. That is why final states are encoded as attack states (the point of view of the intruder). If he succeeds to achieve the final state, then it means that he is able to satisfy all Client's requests having only initial knowledge of the Goal service and being able to invoke the available services (see Figure 4.12).

In order to check whether the attack state can be reached, we have employed[24] a version of

[24] Remark that any tool that accept ASLan as input language, i.e. OFMC [BMV05b] or SATMC [AC08], could be used instead of CL-AtSe. More generally, if we do not stick to ASLan language, any tool that is able to find flaws in security protocols, and more precisely solves reachability problem, can be reused by the same principle.

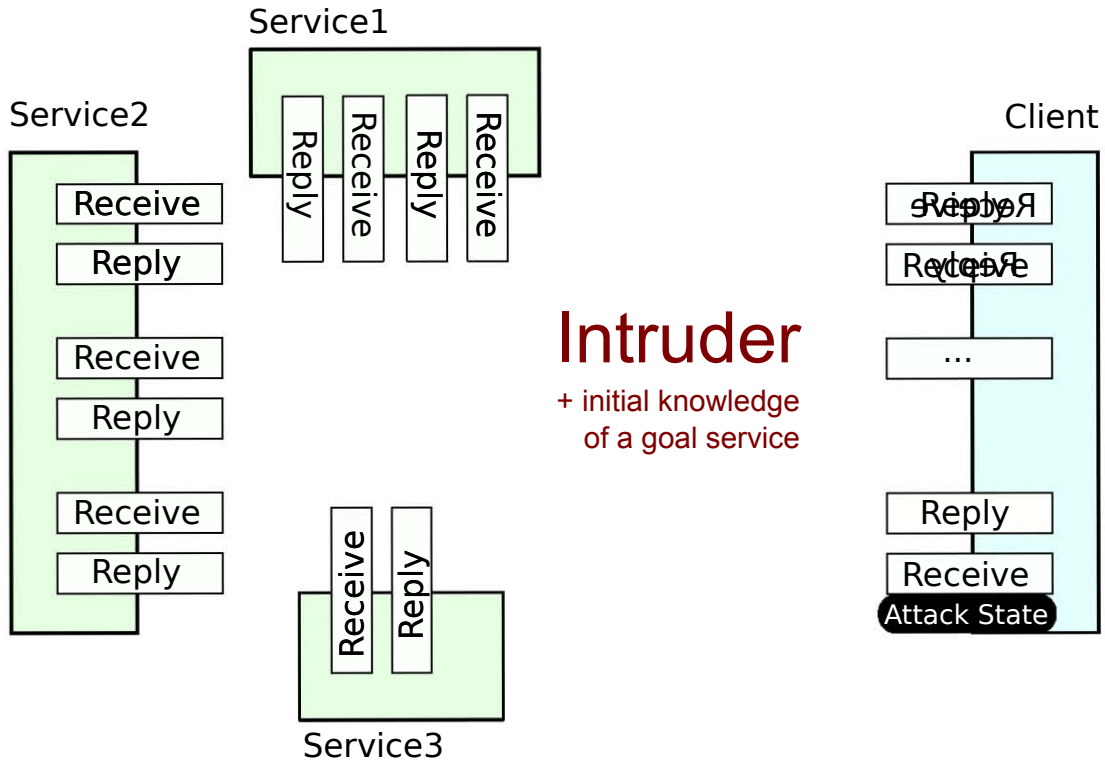


Figure 4.12: Services as protocol roles

the CL-AtSe tool [Tur06, avab] initially intended for solving protocols or Web Services insecurity problem: the result is a trace containing the sequence of messages sent and received by the intruder. From the trace we then extract an executable ASLan specification of the Goal service (see Figure 4.13). In fact, if the intruder can perform the necessary steps to build the messages that appear in the trace, we are guaranteed that the resulting service is executable, since we assumed that the Goal service has at least the same message construction capabilities as a Dolev-Yao intruder. The operations for building messages are not given in the resulting ASLan specification, but only invocations of necessary available services with already constructed request-messages are present.

If the partial specification of the goal is given, the tool first creates a corresponding putative Client service (see Figure 4.14) and thereby reduce the problem input to one discussed above.

Limitations There are still several limitations for the specification of the orchestration problem mainly due to the high expressive power of the ASLan language that cannot be handled when performing orchestration.

Some stronger restrictions are imposed particularly to the Goal-style input for the orchestration problem in order to make a reduction from Goal-style to Client-style input correct. We will not give here details, but the interested reader can find the list of restrictions in [AVA10].

Note also that every transition used in the specification can be fired at most some fixed number of times, by default once, that shows the impossibility of using loops that support arbitrary number iterations not known in advance.

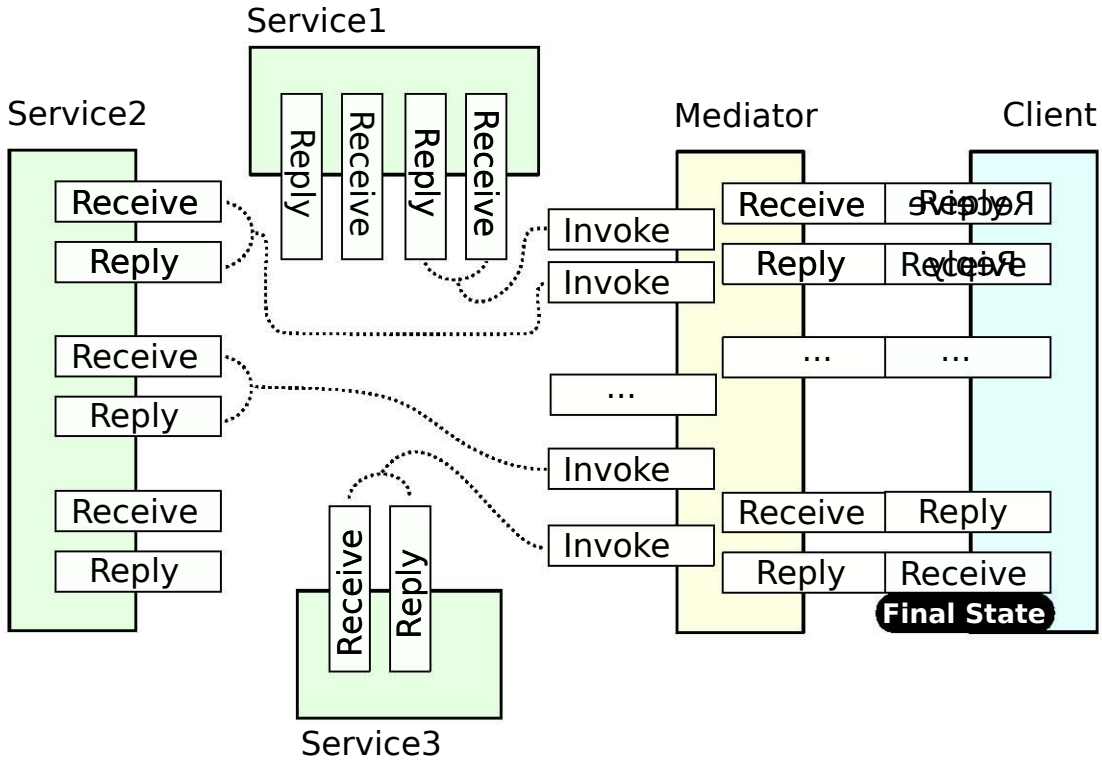


Figure 4.13: Solution to the orchestration problem

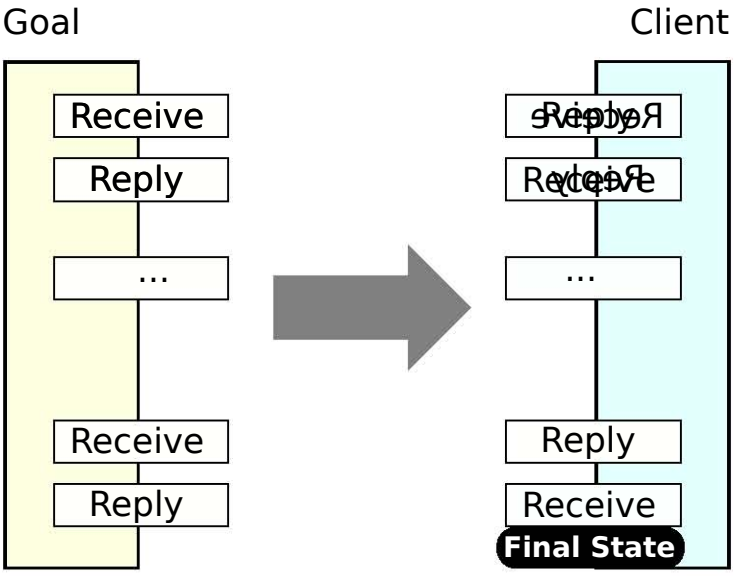


Figure 4.14: Goal and its putative Client

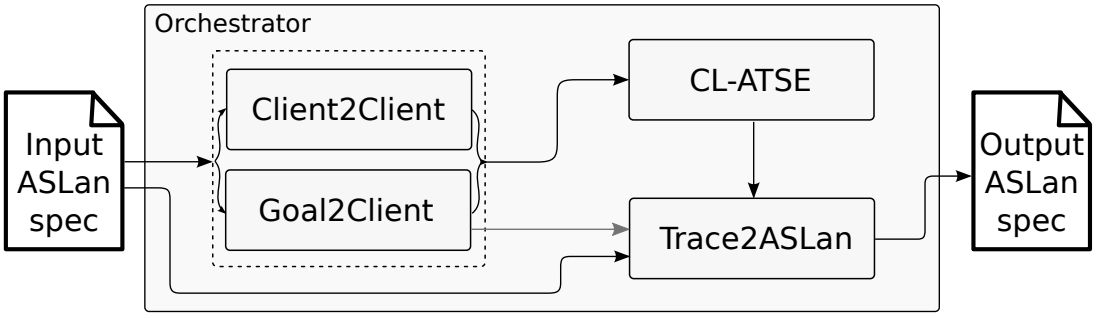


Figure 4.15: Tool architecture

4.5.5 Architecture overview

The Orchestrator takes as input an ASLan file with a specification of the available services and either a specification of the Client or a partial specification of the Goal. It produces as output an ASLan file with the specification of the available services, a full specification of the Goal, and a specification of the Client (a putative one, if it was not given as input).

Figure 4.15 depicts the architecture of the Orchestrator. Input specifications are sent to `Client2Client`, a tool that prepares inputs for CL-AtSe. If `Client2Client` detects that the input is not in a format corresponding to a Client, then `Goal2Client` is invoked with the same input specification. `Goal2Client` prepares the input for CL-AtSe assuming that a Goal specification is given. It also generates additional data to help `Trace2ASLan` integrate the resulting service into an ASLan file. The output of CL-AtSe is given as input to `Trace2ASLan` together with the input specification, and any additional data generated by `Goal2Client`. Then, `Trace2ASLan` integrates the resulting Goal service into the input specification (adding the security properties specified by the modeler for the validation of the generated orchestration).

Now we shortly discuss the mentioned components of the Orchestrator.

`Client2Client`

The purpose of this simple component is to prepare input for the case where a Client specification is given to CL-AtSe. It checks the suitability of the input against the requirements given in § 4.5.2 and compliance to the limitations (see [AVA10]) and initializes the intruder knowledge with the initial knowledge of the Goal.

`Goal2Client`

The purpose of this component is to prepare input when a partial Goal specification is given to CL-AtSe. It generates a Client specification from the partial Goal specification given as input. In more detail, it

1. Checks the suitability of the input against the requirements and limitations;
2. Generates a Client specification from the Goal specification and integrates it into the input specification, removing the Goal;
3. Replaces the intruder knowledge by the initial knowledge of the Goal service; and
4. Outputs the generated secret name, the name of the agent executing the newly created Client service, the well-ordered sequence of goal steps and the renaming table that was done

during transformation of the Goal to the Client to have a link with the input specification, into two auxiliary files, if it was requested with a command-line option.

We give an example of the transformation of a one-transition Goal service to the corresponding Client. Suppose that we have the following specification of the Goal service (the "squared sum" service described in § 4.5.7) as input:

Example 31

```
step step_2 (GOAL,D_I1,D_I2, D_Set_36,SID,I1,I2) :=
 state_OrchestrationGoal(GOAL,1,D_I1, D_I2,D_Set_36,SID).
 iknows(pair(I1,I2))
 =>
 state_OrchestrationGoal(GOAL,2,I1,I2, D_Set_36,SID).
 iknows(apply(times,pair(apply(plus,pair(I1,I2)),
    apply(plus,pair(I1,I2))))).
 secret(apply(times,pair(I1,I2)),sec_prop,D_Set_36).
 contains(GOAL,D_Set_36)
```

The `Goal2Client` returns an ASLan file, where the specification above is replaced with the corresponding Client service.

Example 32

```
  step step_0001(ClientAgent_0000):=
        state_OrchestrationClient(ClientAgent_0000, 1).
        iknows(start)
    =>
        state_OrchestrationClient(ClientAgent_0000, 2).
        iknows(pair(var_Renamed_I1_0000_0000, var_Renamed_I2_0000_0000))

  step step_0000(ClientAgent_0000):=
        state_OrchestrationClient(ClientAgent_0000, 2).
        iknows(apply(times, pair(apply(plus,
           pair(var_Renamed_I1_0000_0000, var_Renamed_I2_0000_0000)),
           apply(plus, pair(var_Renamed_I1_0000_0000,
           var_Renamed_I2_0000_0000)))))
    =>
        state_OrchestrationClient(ClientAgent_0000, 3).
        iknows(finish_0000)
```

In its first transition `step_0001`, the Client sends a message that would be accepted by the Goal from Example 31 (the first dummy send is not counted), and in its second transition Client expects to receive a message that has to be sent by the Goal if the Goal received a message sent by the Client. Here, constants `var_Renamed_I1_0000_0000` and `var_Renamed_I2_0000_0000` represent variables `I1` and `I2`, respectively.

This transformation of variables into constants leads to a correct result when we try to solve an orchestration problem thanks to the restriction on the atomicity of variables used in the Goal specification.

When building a Client, predicates about security properties are ignored, e.g., `secret(apply(times,pair(I1,I2)),sec_prop,D_Set_36)` in Example 31. They are later joined with the resulting specification by the integrator part of the `Trace2ASLan` tool.

CL-AtSe

This is a key component of the generator part of the Orchestrator. CL-AtSe is a Constraint Logic based Attack Searcher for security protocols and services. It relies on a technique of deducibility constraints resolution for well-formed constraint systems. The main idea in CL-AtSe consists in running the protocol or set of services in all possible ways by representing families of traces with positive or negative[25] constraints on the intruder knowledge, on variable values, on sets, etc. Thus, each run of a service step consists in adding new constraints on the current intruder and environment state, reducing these constraints down to a normalized form for which satisfiability is easily decidable, and decide whether some security property has been violated up to this point. CL-AtSe does not limit the service in any way except for bounding the maximal number of times a service can be iterated, in the case such an iteration is allowed in the specification (otherwise, the analysis might be non-terminating).

Given an ASLan specification, it generates a trace — the sequence of communication events (send/receive) between a mediator (played by the intruder) and the available services (that can be used in the composition) — such that a special attack state is reached and an LTL formula is satisfied.

In general, any tool that is able to solve protocol insecurity problems could be used instead. Besides some other advantages, CL-AtSe was chosen since it is based on solving of (well-formed) deducibility constraints described in Part I.

A detailed description of the tool can be found in [Tur06, AT09] and [AVA10, avab].

`Trace2ASLan`

The objective of this tool is to translate the trace describing the Goal service behavior to an ASLan specification then to integrate the latter to the ASLan file given in input. In this way we will obtain a specification that is ready to be verified (with regard to the desired security properties) by AVANTSSAR Validator.

The tool first proceeds by *end-point projection* [CHY07] of the conversation trace returned by CL-AtSe to extract the local behavior of the Goal service. As pointed out in [MK08] this is not straightforward when cryptography is allowed in the message patterns. The main problem with the use of cryptography is that the structure of a message maybe differently seen by its corresponding sender and receiver due to the current knowledge collected by the participants so far. To illustrate this property we consider the simple conversation of Example 33:

Example 33 We consider a setting where a service A initially knowing some symmetric key k and some message m and a service B with no initial knowledge have the following conversation:

$$A \to B : \mathrm{enc}\,(m, k)$$

One can think of writing the following projections for A and B, where each service is parametrized by its initial knowledge in brackets:

$$A[k, m] = !\,\mathrm{enc}\,(m, k) \qquad\qquad B[\,] = ?\,\mathrm{enc}\,(m, k)$$

The problem with the naive projection of Example 33 is that from the point of view of B the received message cannot be checked against the provided pattern $\mathrm{enc}\,(m, k)$ since B at

[25] Here we mean disequalities, but not negative deducibility constraints.

the reception of the message knows neither message itself nor key k. Thus, verifying security properties in the presence of DY intruder for such a projection of B is not really meaningful since this behavior does not correspond to what could be a real implementation of the service. Since B has no means to verify whether the received message is "good" or not, it will accept any message in his first step. In the worst case this can lead to situations where "naively" projected services shadow some attacks when verifying, while these attacks remain feasible for the real implementation. An example of a shadowed attack on the authentication performed by a Dolev-Yao intruder with no initial knowledge for the described case is sketched below.

While in the case of naive projection B once received the expected message can be sure that it was emitted by A, in the real case, where B cannot check the content of the message ($B[\,] = ?X$), the intruder can send any message to B. In this case B is not guaranteed the message was emitted by A. Thus, the authentication on the message received by B is preserved in naive approach while can be violated in a real implementation.

In short, the specification obtained by the naive projection cannot always have the correctly implementation.

To remove this mismatch between what is verified and what could be really implemented we propose to take into account more information that we have: the current knowledge of the service. The pattern to be sent or received by a service must reflect such message structure that can be checked at this step. Later, if new information is learned that could help to explore further the structure of the message, an additional check (usually in a form of equality) must be performed. This corresponds to a *prudent* projection [CR10a], where the service checks his input as thoroughly as possible by performing all the required correlation and security checks on the received messages. Such projection is operational in the sense that it is possible to specify the sequence of actions to be performed (message composition and decomposition) at each communication step.

Let us extend Example 33.

Example 34 Service A initially knows k, m and n, and service B has no initial knowledge. Consider the following conversation:

$$\begin{aligned} A \to B: &\quad \mathrm{enc}\,(m,k) \\ A \to B: &\quad k \\ B \to A: &\quad m \\ A \to B: &\quad \mathrm{enc}\,(\mathrm{pair}\,(n,n),k) \end{aligned}$$

The prudent projection of B in this case is:

$$B[\,] = ?X.\,?K.\,\mathrm{eq}\,(X,\mathrm{enc}\,(M,K))\,.\,!M.\,?\,\mathrm{enc}\,(\mathrm{pair}\,(N,N),K)$$

where $\mathrm{eq}\,(X,\mathrm{enc}\,(M,K))$ is a condition meaning $X = \mathrm{enc}\,(M,K)$ that must be satisfied upon the reception of K. First B receives $\mathrm{enc}\,(m,k)$ but have no means to check the content of the received message, thus it will accept anything and store it in variable X. Then it receives key k and since we know the structure of the first message received, now B can check it ($\mathrm{eq}\,(X,\mathrm{enc}\,(M,K))$) by decrypting X with K and the resulting message is assigned to M. Then, he can send this value back since we know that M should have value m. Finally, it receives a pair of yet unknown values, whose values can be checked to be equal (and that will be stored in variable N) encrypted by already known key stored in K.

Note that there is another (equivalent) way to present a prudent projection of B in Example 34: transfer an implicit check that is encoded into the message pattern to the explicit check. That is, the last action $?\,\mathrm{enc}\,(\mathrm{pair}\,(N,N),K)$ can be written as $?Y.\,\mathrm{eq}\,(Y,\mathrm{enc}\,(\mathrm{pair}\,(N,N),K))$.

The prudent projection is based on the notion of *reachability* of message parts (more precisely, subterms) depending on the current knowledge of the service. A subterm t is *reachable* at some step i if $t \in \mathrm{Der}_{DY}(K_i)$, where K_i is a current knowledge of the service collected so far.

Note that in case of reception this exactly makes precise the parts that can be checked for correlation with their possible occurrences within previously received messages (or within the current reception itself, like digital signature verification). In case of sending it makes precise the parts needed to compose the message to be sent.[26]

While we confined ourselves by several examples and general explications, the algorithm for building prudent projection can be found in [AVA10].

4.5.6 Security Loopback

As mentioned above, when specifying an orchestration problem, a modeler can add security properties to be automatically validated against the obtained orchestration. Or, as a possibility, once an orchestration is obtained, the modeler can add security properties into the specification and try to validate them using the validation tools of the platform.

It is possible that the composition generated by the Orchestrator does not satisfy the desired security properties, because the Orchestrator itself does not deal with the validation of global security properties.

In this case the user can indicate to the Orchestrator that the generated composition is vulnerable, and then, the Orchestrator should produce a *different* mediator service, if possible.

In this subsection we will give an overview of the Orchestrator's facility for the security loopback.

Summing up

The Orchestrator is able to generate on-demand a new different solution by restoring a previously saved position in his search tree.

The core of the Orchestrator, CL-AtSe, that produces a trace from which the Goal service is recreated using `Trace2ASLan` tool, is able to save its current state, i.e., the current position of the search tree. When a trace is found, the tool writes down its state to a file. The idea is to resume searching of a trace starting from the saved point. In this way, CL-AtSe can run through the entire search tree and probably output different possible traces (for example, it can use alternative available services that offer the necessary functionality, but in more secure way). From the new traces, new specifications of the Goal service (Mediator) are reconstructed. Note the drawback of the approach is incompleteness, i.e., the Orchestrator is sure to return an orchestration if one exists, but is not guaranteed to return all possible orchestrations.

In Figure 4.16 the data needed to restore the process of the orchestration lookup is highlighted:

- An input ASLan specification (needed to integrate in it the specification of the generated Mediator),
- A current state of CL-AtSe (needed to find a new attack trace from which the Mediator will be built), and
- In case of Goal-style input, the helper files (needed, e.g., to reconstruct the initial variable names after `Goal2Client` preprocessing).

[26] We recall that by construction of the trace describing the mediator all the messages it sends at some step are reachable by it at that step.

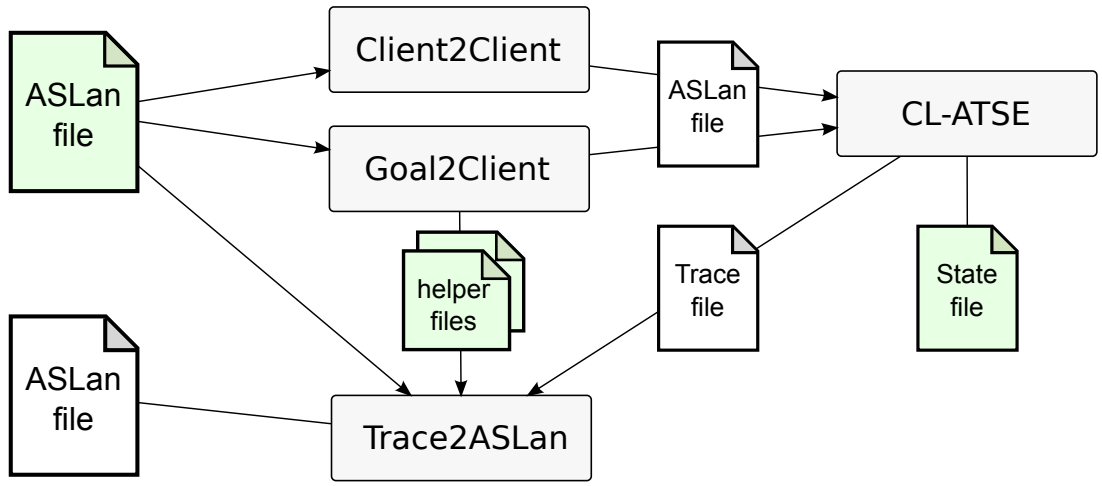


Figure 4.16: Data flow inside the Orchestrator tool.

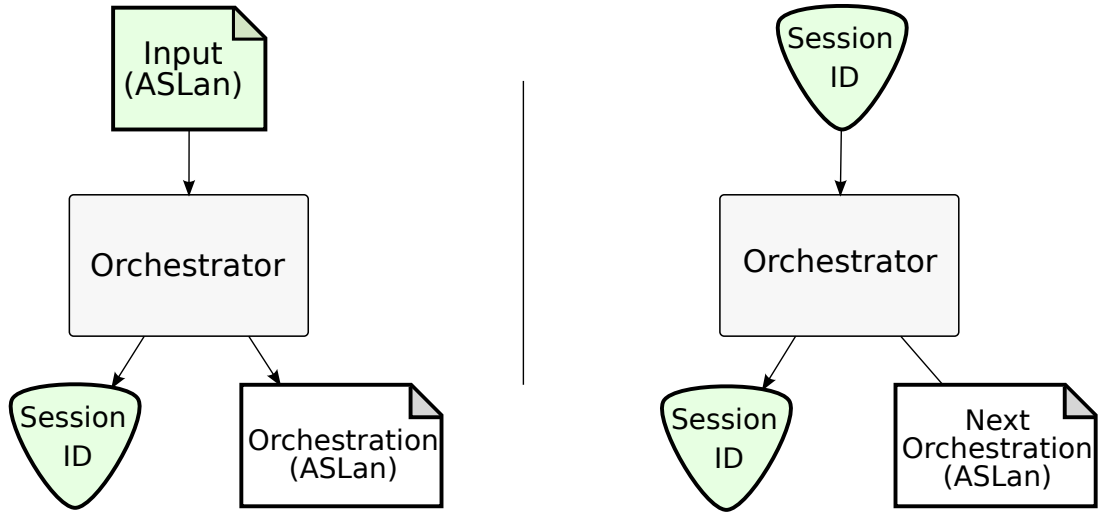


Figure 4.17: First and following invocations of the Orchestrator

Before starting the generation of the orchestration, the tool allocates a unique identifier (referred further as *sessionID*) to identify the current job. This identifier is also used to name the files that contain data mentioned above needed to restore the orchestration lookup. The *sessionID* is also returned by the tool together with the generated orchestration, such that user can "ask" the Orchestrator to generate the next possible orchestration by re-feeding this identifier (see Figure 4.17).

As the files necessary for resuming the orchestration generation can be recovered based on the *sessionID*, the tool can easily continue its execution. The orchestration chain then is shown in the Figure 4.18 (the data recovered by *sessionID* is marked out).

Thus, we can re-invoke the Orchestrator to obtain the next possible orchestration, if the previous one did not satisfy the security (or any other) requirements.

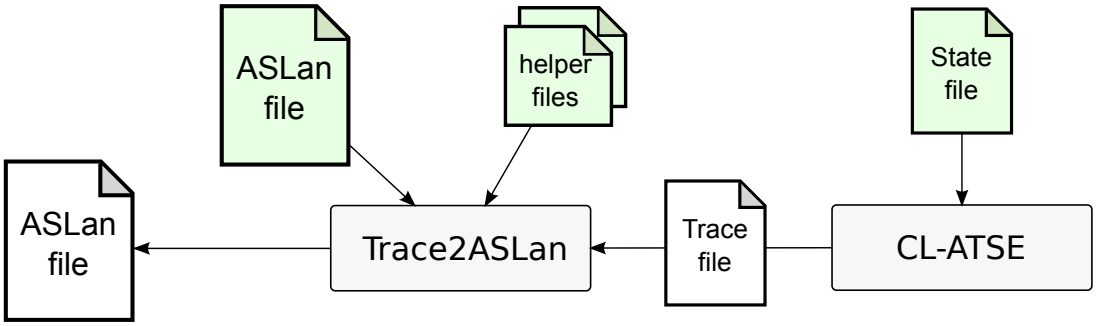


Figure 4.18: Resume to generating the next orchestration

4.5.7 A toy example

In this section we present a simple orchestration problem and a way it can be specified in ASLan. We also give some key parts of the Orchestrator tool's output for this specification.

Suppose that we want to create a service that accepts two numbers and returns the square of their sum. The target service is not supposed to carry out the multiplication and addition operation, but rather to rely on available services for this: the multiplier (which multiplies two given values) and the adder (which computes the addition of two given numbers).

First, we must specify the available services.

Available services:

Adder: receives two values a, b, and sends $a + b$.

```
step step_0 (A,SID,I,J) :=
 state_Adder(A,1,dummy_msg,dummy_msg,SID).
 iknows(pair(I,J))
 =>
 state_Adder(A,2,I,J,SID).
 iknows(apply(plus,pair(I,J)))
```

Its state-fact in the initial state is:

```
  state_Adder(a,1,dummy_msg,dummy_msg,3)
```

Multiplier: receives two values a, b, and sends $a * b$.

```
step step_1 (M,SID,I,J) :=
 state_Multiplier(M,1,dummy_msg,dummy_msg,SID).
 iknows(pair(I,J))
 =>
 state_Multiplier(M,2,I,J,SID).
 iknows(apply(times,pair(I,J)))
```

Its state-fact in the initial state is:

```
  state_Multiplier(m,1,dummy_msg,dummy_msg,4)
```

Then, as there are two options for specifying the input, we have to choose between the partial Goal description and the Client description.

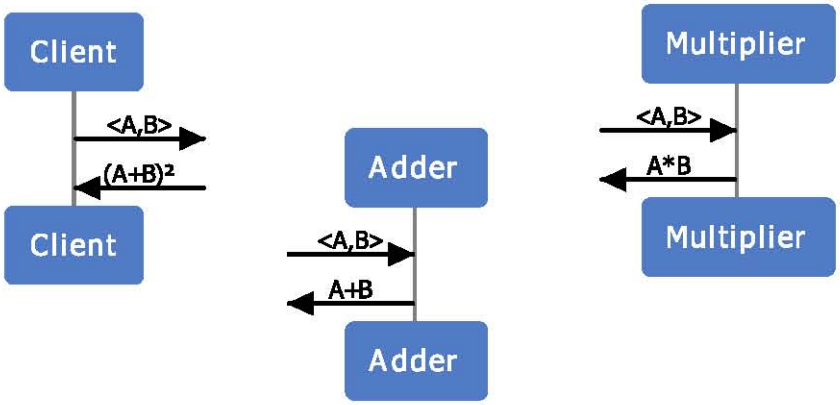


Figure 4.19: Toy example: available services and the Client

Target services:

Client: sends a, b; expects $(a+b)*(a+b)$.

Goal: receives a, b; returns $(a+b)*(a+b)$.

Client case

If the modeler chose to specify the behavior of the Client to define an orchestration problem, then the Client specification is:

```
step step_2 (C,SID,I1,I2) :=
 state_OrchestrationClient(C,1,dummy_nonce, dummy_nonce,SID).
=[exists I2,I1]=>
 state_OrchestrationClient(C,2,I1,I2, SID).
 iknows(pair(I1,I2)).
 keepsecret(apply(times,pair(I1,I2)), secret_value_id)

step step_3 (C,I1,I2,SID) :=
 state_OrchestrationClient(C,2,I1,I2, SID).
 iknows(apply(times,pair(apply(plus,pair(I1,I2)),
    apply(plus,pair(I1,I2)))))
 =>
 state_OrchestrationClient(C,3,I1,I2, SID).
 iknows(end_orchestration)
```

Note here `=[exists I2,I1]=>` shows the generation of random values for variables `I1` and `I2` (a so-called nonce generation).

Its initial state is:

```
 state_OrchestrationClient(c,1,dummy_nonce, dummy_nonce,5).
```

The last `iknows(end_orchestration)` is a mark of the final Client transition (see the corresponding paragraph in § 4.5.2 and Example 30); the corresponding attack state is defined as:

```
attack_state orchestrationFinalState (ASGoal) :=
 iknows(end_orchestration)
```

The fact `keepsecret(apply(times,pair(I1,I2)), secret_value_id)` is given as an example of security property to be verified. It is not related to an orchestration problem, but to validation. The corresponding attack state is defined as follows:

```
attack_state to_validate(MGoal) :=
 iknows(MGoal).
 keepsecret(MGoal,secret_value_id)
```

Conforming to the requirements listed in § 4.5.2, we should also define an initial knowledge of the Goal service:

```
 state_OrchestrationGoal(g)
```

That is, he knows only some value g. Could be empty.

Output As output of the tool we will obtain a Goal specification integrated into the input file[27]:

```
...
step step_OG_1 (OGoal,SID,G,Dummy_I1_n3,I1_n3,Dummy_I2_n3,I2_n3,
   Dummy_X4,Dummy_X7,Dummy_Eo) :=
 state_OGoal (OGoal,SID,1,G,Dummy_I1_n3,Dummy_I2_n3,
    Dummy_X4,Dummy_X7,Dummy_Eo).
 iknows(pair(I1_n3,I2_n3))
 =>
 state_OGoal (OGoal,SID,3,G,I1_n3,I2_n3, Dummy_X4,Dummy_X7,Dummy_Eo).
 iknows(pair(I1_n3,I2_n3))

step step_OG_2 (OGoal,SID,G,I1_n3,I2_n3, Dummy_X4,X4,Dummy_X7,Dummy_Eo)
   :=
 state_OGoal (OGoal,SID,3,G,I1_n3,I2_n3, Dummy_X4,Dummy_X7,Dummy_Eo).
 iknows(X4)
 =>
 state_OGoal (OGoal,SID,5,G,I1_n3,I2_n3, X4,Dummy_X7,Dummy_Eo).
 iknows(pair(X4,X4))

step step_OG_3 (OGoal,SID,G,I1_n3,I2_n3, X4,Dummy_X7,X7,Dummy_Eo) :=
 state_OGoal (OGoal,SID,5,G,I1_n3,I2_n3, X4,Dummy_X7,Dummy_Eo).
 iknows(X7)
 =>
 state_OGoal (OGoal,SID,7,G,I1_n3,I2_n3, X4,X7,Dummy_Eo).
 iknows(X7)

step step_OG_4(OGoal,SID,G,I1_n3,I2_n3, X4,X7,Dummy_Eo,Eo) :=
 state_OGoal (OGoal,SID,7,G,I1_n3,I2_n3, X4,X7,Dummy_Eo).
 iknows(Eo)
 =>
 state_OGoal (OGoal,SID,8,G,I1_n3,I2_n3,X4,X7,Eo)
...
```

[27] We replaced `OrchestrationGoal` with `OGoal` and `End_orchestration` with `Eo` in tool output for better display

Goal case

The other option is to partially define the Goal service, i.e. specify a part related to the communications with the Client:

```
step step_2 (GOAL,SID,I1,I2) :=
  state_OrchestrationGoal(GOAL,1, dummy_msg,dummy_msg,SID).
  iknows(pair(I1,I2))
=>
  state_OrchestrationGoal(GOAL,2,I1,I2, SID).
  iknows(apply(times,pair(apply(plus,pair(I1,I2)),
     apply(plus,pair(I1,I2))))).
  keepsecret(apply(times,pair(I1,I2)),secret_value_id)
```

Here, we also added a secrecy property to be verified by the Validator. The corresponding attack state is:

```
section goals:

attack_state to_validate (MGoal,ASGoal) :=
iknows(MGoal).
keepsecret(MGoal,secret_value_id).
```

The initial state includes

```
state_OrchestrationGoal(g,1,dummy_msg,dummy_msg,5)
```

to initialize the Goal service.

Output We give a part of the output containing a specification of the putative Client and full specification of the Goal[28]:

```
...
step step_OG_1(OGoal,SID,G,X_Int1,Dummy_msg,X_Int5,
   Dummy_I1,I1,Dummy_I2,I2,Dummy_X7,Dummy_X10) :=
 state_OGoal(OGoal,SID,1,G,X_Int1,Dummy_msg,X_Int5,
    Dummy_I1,Dummy_I2,Dummy_X7,Dummy_X10).
 iknows(pair(I1,I2))
 =>
 state_OGoal(OGoal,SID,3,G,X_Int1,Dummy_msg,X_Int5,I1,I2,
    Dummy_X7,Dummy_X10).
 iknows(pair(I1,I2))

step step_OG_2(OGoal,SID,G,X_Int1,Dummy_msg,X_Int5,I1,I2,
   Dummy_X7,X7,Dummy_X10) :=
 state_OGoal(OGoal,SID,3,G,X_Int1,Dummy_msg,X_Int5,I1,I2,
    Dummy_X7,Dummy_X10).
 iknows(X7)
 =>
 state_OGoal(OGoal,SID,5,G,X_Int1,Dummy_msg,X_Int5,I1,I2, X7,Dummy_X10).
 iknows(pair(X7,X7))

step step_OG_3(OGoal,SID,G,X_Int1,Dummy_msg,X_Int5,I1,I2,
   X7,Dummy_X10,X10) :=
 state_OGoal(OGoal,SID,5,G,X_Int1,Dummy_msg,X_Int5,I1,I2, X7,Dummy_X10).
```

[28] We replaced `OrchestrationGoal` with `OGoal` in tool output for better display

```
 iknows(X10)
 =>
 state_OGoal(OGoal,SID,7,G,X_Int1,Dummy_msg,X_Int5,I1,I2, X7,X10).
 iknows(X10)

step step_0001(ClientAgent_0000,IID_0000) :=
 state_OrchestrationClient(ClientAgent_0000,IID_0000,1)
 =>
 state_OrchestrationClient(ClientAgent_0000,IID_0000,2).
 iknows(pair(var_Renamed_I1_0000_0000, var_Renamed_I2_0000_0000))

step step_0000(ClientAgent_0000,IID_0000) :=
 state_OrchestrationClient(ClientAgent_0000,IID_0000,2).
 iknows(apply(times,pair(apply(plus,pair(
    var_Renamed_I1_0000_0000,var_Renamed_I2_0000_0000)),
    apply(plus,pair(var_Renamed_I1_0000_0000,
    var_Renamed_I2_0000_0000)))))
 =>
 state_OrchestrationClient(ClientAgent_0000,IID_0000,3).
 iknows(finish_0000)
...
```

4.5.8 Some experimental results

We roughly describe one case study for which we modeled in ASLan an orchestration problem and sketched the resulting composed service produced by AVANTSSAR Orchestrator.

Then, we give tool execution time for several case studies for which the orchestration problem was modeled in ASLan, and, in order to give some hints about the complexity of problems, we also present some quantitative characterization of the considered specifications.

Digital Contract Signing case study

We describe in the following an experiment we had with the Orchestrator tool on the Digital Contract Signing case study (DCS). DCS describes two parties that have secure access to a trusted third party Web site, a Business Portal (BP), in order to digitally sign a contract. First, BP generates an electronic document corresponding to the terms of agreement between the two parties. Then, the first party accesses BP using a Web browser, views the contract and signs it using a digital certificate. BP verifies the generated signature and stores it. The second party, in turn, connects to the BP Web site, checks the status of the existing signature and then co-signs the contract after viewing it. Once the signatures have been completely verified by the business portal, the signers are notified. Then, the contract is archived for long-term conservation. The BP's internal system is Web service enabled. It delegates the processing of proof elements (signatures, signed documents, timestamps) to a Security Server (SS) using SOAP messages.

Three available trusted services are in the disposal of SS: a Time Stamper (TS), a Public Key Infrastructure (PKI), that returns information about the validity of a given certificate and an Archiver (ARC), an external storage facility.

The orchestration problem here is to generate a Mediator that emulates SS: satisfy BP's requests while relying on the community of available services (namely TS, PKI and ARC). Figure 4.20 represents the solution generated by the tool.

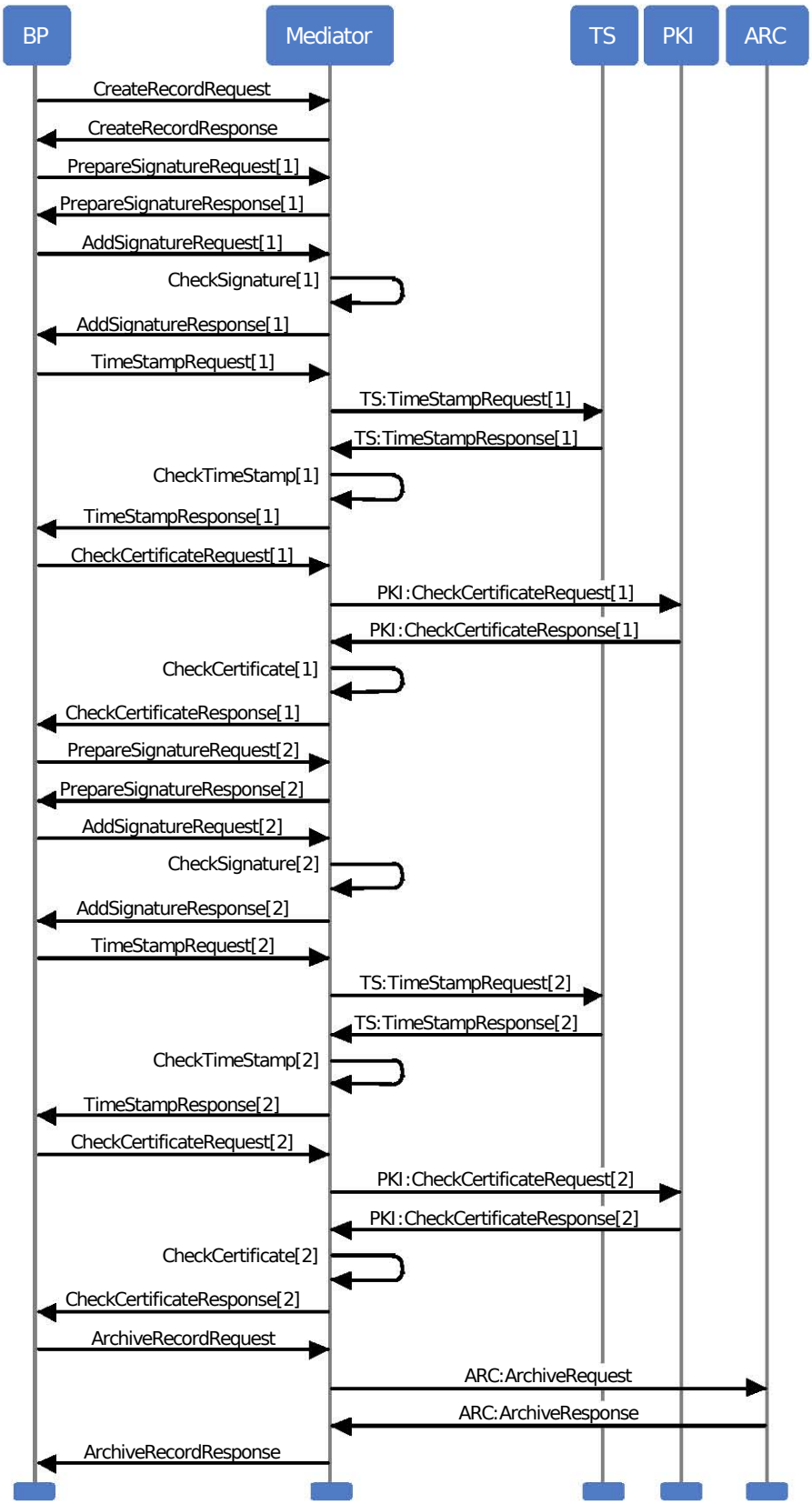


Figure 4.20: A Mediator service for DCS orchestration problem

Input problem				Mediator generation	
Case study	Number of available services	Number of transitions	Number of Horn Clauses	Number of generated transitions	Running time
CRP	4	25	17	17	4.1 sec.
DCS	3	18	0	22	4.6 sec.
PB	2	20	2	4	1.7 sec.

Table 4.4: AVANTSSAR Orchestrator benchmark

Indeed, the generated Mediator service (SS) expects an initialization message to start the digital signature procedure and acknowledge the reception. Then BP invokes SS to get the signature policy for the first signer. Then BP transfers the contract signed by the first signer to SS which should check the signature and produce an assertion about its validity that BP expects back as a response. Then SS is asked to time stamp the signature and provide the corresponding assertion. To obtain the time stamp, SS must invoke the trusted TS. After this SS is asked to check whether the certificate used by the signer was not revoked, and if it was not, to return the corresponding assertion. Indeed, SS contacts PKI and tries to derive the needed assertion from its response. If SS succeeds the first round of the signature procedure is successfully ended. Similar steps are needed to collect the second signer's data before SS is asked by BP to archive the documents and proofs collected during the signature procedure. Therefore SS invokes ARC with the appropriate message, the latter acknowledges the reception and finally SS calls back BP to signal the successful end of the signature procedure.

Assertions produced by SS are claims made by some issuer and stating some property for the parameters they transport. An equivalent in the Web Service standards stack is SAML [OAS05] assertions, which we simply model using first-order terms. The presence of an assertion in some received message by BP represent an additional constraint to the orchestration problem since SS will have to provide it. In this case-study one of the assumptions was that BP trusts SS as issuer for the assertions it required for example about the validity of one signer's certificate. To produce this assertions SS have to contact an internal service: the Assertions Provider (AP) which permits to provide the good assertion only if a positive answer about the validity of the certificate is given by a trusted third-party (here PKI). AP plays a role similar to the *trust engine* in *rely-guarantee* method introduced in [GTC+04]. Note that the calls to AP are abstracted in Fig. 4.20 by the returning arrows to SS.

We underline here the expressiveness of assertions for the considered orchestration problem, since they can describe for example the need to use only certain schema for the time stamps, or only PKI's offering the *Online Certificate Status Protocol (OCSP)* versus those using the classical *Certificate Revocation List (CRL)*. This can be easily done by tuning the AP service behavior to match the wanted expectancies.

Benchmark

We have tested the tool on several industrial case studies, like *Digital Contract Signing* (DCS) and *Public Bidding* (PB) which are originated from commercial products of the OpenTrust company and *Car Registration Process* (CRP), a case study proposed by Siemens AG.

In all cases the tool successfully terminated and output the expected composed service. Execution time and some quantitative characteristics are given in Table 4.4).

4.6 Conclusions

A survey of works on automatic composition of services presented in § 4.1 showed that there is no model that takes into account security policies of services and in the same time admits a fully automatic composition procedure. We tried to fill this gap by proposing our approach for Web Services composition.

We presented a new model for distributed orchestration of Web Services under non-disclosure condition. This model allows to describe both orchestration and choreography, since our notion of distributed orchestration is more general. The Web Services are presented as sequence of actions (receive/send) on which security policies are already applied, i.e. the actions of Web Services are "cryptography aware": the message patterns to be received may include cryptographic primitives as well as ones to be send. The problem is to build a Mediator, a service, whose parts are provided by partner organizations that can reuse available services, which is able to satisfy all requests of a given client. Moreover, partners do not want to transfer sensitive data to each other (non-disclosure condition), and this is also taken into account in the model.

We showed that the problem of distributed orchestration in the presented model, under assumption that the number of interactions is bounded, can be reduced to satisfying a system of deducibility constraints and a non-disclosure condition. Making a very minor update of the decision procedure introduced in Part I we could reuse it to solve the mentioned satisfiability problem, and thus also the distributed orchestration.

We reported an implementation called AVANTSSAR Orchestrator for the case of (non-distributed) orchestration. The Orchestrator uses analogy of the orchestration problem with the protocol insecurity problem and employs CL-AtSe, a tool from cryptographic protocol analysis domain. Since the input language of CL-AtSe allows richer constructions, like branching, we can express a more elaborated behavior of the Web Services than one presented in our model. The Orchestrator as a part of AVANTSSAR Validation platform allows for finding an orchestration that satisfies some global security properties to resist an active Dolev-Yao attacker.

The tool was successfully tested on several industrial case studies. The Orchestrator is deployed and available at `http://avantssar.loria.fr/OrchestratorWI/`. It was implemented in OCaml and Java and its source contains of more than 20'000 lines of code[29].

Critics There are several aspects that are not taken into account. For example, a linear workflow without branches makes a restriction on the class of services we can express. Note that in automata-based approaches the workflow is much more expressive since it allows loops and branches, but usually the message structure is abstracted away.

Another related restriction is that our method is constrained by a bounded number of instances of Web Services. In reality it is not the case, since one can invoke a Web Services as many times as needed. The same underlying reasons make impossible to consider loops. However, for non-distributed case, one may employ tools (like ProVerif [Bla01]) that are able to cope with protocol insecurity problem using unbounded analysis. But in this case, solving the orchestration problem may be non-terminating.

The definition of non-disclosure condition as we presented may be too restrictive, but like discussed in § 4.3, to handle more natural approach we need more elaborated techniques.

The so-called "security loopback" of AVANTSSAR Validation platform provides only incomplete procedure to get a secure orchestration. That is, we cannot guarantee that if there exist

[29] Including the source code of CL-AtSe

an orchestration satisfying the given security properties, it will be found, which is surely can be seen as another disadvantage.

Research directions One of the research directions is to extend the model by considering more complex behavior of Web Services and the generated Mediator while preserving the possibility to automatically solve the distributed orchestration problem. Although the branching can be possibly added without much problems, the loops create difficulties for the presented method, as we have to consider infinite number of interactions in order to deal with full-fledged loops.

The solution could possibly be found in combining of automata-based composition approaches and one presented in this work. But this requires more detailed research.

As was mentioned several times before, an efforts can be made in order to more naturally model non-disclosure condition. Some initial tracks to solve this problem has already been given in § 4.3, p. 82 and in Part I.

Another useful direction is is a detailed study of Web Services standards, like BPEL [Oas06], WSDL [Wor01], WS-SecurityPolicy (WSSP) [DLHBH+02, Oas07] and others. An explicit correspondence between security-aware messages in our model and its presentation in the world of standards would be a step to a deeper integration of automatic composition techniques in its industrial usage.

Chapter 5

Verification of cryptographic protocols

Contents

5.1 Introduction 115
5.1.1 Attacker Models 117
5.1.2 From protocol sessions to constraints solving 119
5.2 Motivation 123
5.3 Formal model 124
5.4 Solution approach 128
5.5 XML rewriting attacks 129
5.5.1 Example of discovering an attack exploiting XML format 130
5.6 Conclusions 133

5.1 Introduction

Cryptographic protocols is an underlying layer needed to protect digital communication and guarantee desired security properties, like secrecy, authentication, fairness, non-repudiation and others [Guo08] against some malicious entity that is usually called *adversary*, *attacker*, *intruder*, *saboteur* or *penetrator*. It uses cryptographic primitives like encryption and digital signature to achieve its goals.

However an imprudent use of cryptography may lead to some logical attacks, in which the intruder does not need to break the encryption schemes, but simply smartly plays with messages by performing eligible operations, abusing the initial idea of the protocol usage. Moreover, interprotocol attacks can also take place, where some messages from one protocol are reused by the intruder to inject them into the communication of the other.

We would like to note that despite the variety of security properties, in this chapter we will concentrate on the *secrecy* property, that is, a checking whether in a given protocol session the given value cannot be known by an intruder.

The critical role of the cryptographic protocols in ubiquitous digital communications caused the necessity of their *formal* and *automatic verification*. For doing this a formal model is needed. Moreover, since the development of decision procedures (not to mention their effective implementation) is not an easy task, some abstractions in the model must take place.

While the messages transferred via some media in a real communication are presented as bit-strings[30] in a logical level, symbolic models represent them as terms (see Definition 2.1.1, p. 28) reflecting the structure of the message. For example, an encryption of a message `'messagetoencrypt'` (`0x7468 6576 6572 7973 6563 7265 746b 6579`) with some key `'theverysecretkey'` (`0x6d65 7373 6167 6574 6f65 6e63 7279 7074`) can be presented as term $\mathrm{enc}(a, k)$ (we don't care about the actual values: a represents an encrypted message and k represents the encryption key), whereas a real message with a specific encryption scheme (here AES) will look like `0xcd54 381e 3b8f 5981 f108 76e9 4e64 b4b6`. The approach where the real bit-string representation of the messages are not taken into account and abstracted by some symbols is called *symbolic analysis* (see, e.g., [MS03, Bor01, CK07]).

Moreover, usually a *perfect cryptography assumption* is considered, where the cryptographic schemes for encryption, signature and one-way functions are considered as black boxes, and no one can, for example, decrypt an encrypted message without having the necessary key (see, e.g., [DY83]). A lot of protocols were proposed but still, after some time some logical attacks were discovered. The classical example is a man-in-the-middle attack found by G. Lowe in 1995 [Low95] on a Needham-Schroeder public key authentication protocol [NS78] presented in 1978. In this attack the intruder have no need to break cryptography schemes, but smartly "play" with messages.

Later, some additional properties of cryptographic schemes have been considered in symbolic analysis. For example, an RSA encryption scheme [RSA78] has a commutativity property, i.e. $\mathrm{enc}(\mathrm{enc}(m, p), q) = \mathrm{enc}(\mathrm{enc}(m, q), p)$, which was studied, e.g., in [CKRT04] in the context of protocol analysis. The benefit of considering the properties indicating additional information on cryptographic schemes is to better reflect the the real world into the assumed model. This can help reveal such attacks [CKRT05, CLS03] that could not be discovered under perfect cryptography assumption.

Besides the properties of encryption schemes, it is useful to consider also properties of other symbols. For example, in this work we consider an associative commutative idempotent symbol that can be used to represent (non-empty) sets of elements that can be useful in the context of Web Services [CLR07]. The point is that using XML representation, messages may be processed without examining its full structure, but only choosing the necessary elements (cf. choosing an element from a set). We will show an example of modeling XML-rewriting attack in § 5.5, p. 129.

Another point to consider is an appropriate intruder model. Depending on the target usage of the protocol, different models of attacker might be assumed. For example, in [CGH+05] the authors advocated for the usage of suited attacker model. Following their motivation, the considered attacker may be too weak, and the security could be proven formally, while it does not hold in the actual implementation environment. On the other hand, a too strong attacker model can place restrictive constraints on the design of the protocol that impact resource usage and overall functionality.

While the active Dolev-Yao intruder is assumed most often, we contribute with another, weaker intruder model, where multiple non-communicating intruders are considered in contrast to one that controls whole network.

We give a small non-exhaustive review of intruder models considered in literature.

[30] Strings composed with 1 and 0: $\{0,1\}^*$. We will also use their hexadecimal representation.

5.1.1 Attacker Models

Active Dolev-Yao intruder

In 1983 Danny Dolev and Andrew C. Yao presented a formal intruder model (that was informally proposed in 1978 by Roger M. Needham and Michael D. Schroeder in [NS78]) that they call "active eavesdropper" [DY83]. Literally, they describe intruder as

> ... someone who first taps the communication line to obtain messages and then tries everything he can in order to discover the plaintext ... we will assume the following about a saboteur:
>
> a) He can obtain any message passing through the network.
>
> b) He is a legitimate user of the network, and thus in particular can initiate a conversation with any other user.
>
> c) He will have the opportunity to be a receiver to any user A. ...

This model was widely adopted by the community and became a classical (for example, according to Google Scholar[31] this work is cited more than 2000 times). In general, the operations that the intruder can perform on messages can vary (e.g. in [DY83] only pairing/projection and asymmetric encryption/decryption are allowed) but usually the set of considered operations more or less resembles the ones presented in Table 2.4, p. 51.

Passive Dolev-Yao intruder

The passive Dolev-Yao intruder is a much weaker form of the active Dolev-Yao attacker described above, where the latter is limited only by overhears and he cannot intervene into the execution of the protocol session. In other words, if we consider statements that Dolev and Yao assumed about the intruder, the only first (a) holds.

The problem of deciding whether the intruder can derive the secret from some knowledge is sometimes called *intruder deduction problem*, or *intruder knowledge problem*. A lot of works was done for this case (e.g. [DLLT08, CKRT05, CKR+03, CLS03, LLT05, Del06a, CLT04, CKRT04]) by assuming different algebraic properties of symbols in considered term algebra.

Intruder in wireless networks

Compromised node In [NH06] the authors presented a framework for specification and security analysis of communication protocols for mobile wireless networks. The considered attacker is an ordinary node with Dolev-Yao capabilities and cannot control all communication channels, but only those allowed by the network topology. In other words, it is a standard Dolev-Yao active intruder that controls a part of the communication channels.

Over-the-Air An intruder model mentioned in [AVI03a, Kø05] was intended to check protocols insecurity in wireless communications, where intruder may hear all traffic and send messages, but not prevent messages from reaching their destination. A related property of communication channels called *Atomicity of Transmission* (AOT) was considered in [CGH+05]. It states that the intruder cannot both block and overhear a message sent by a communication channel having AOT property.

[31] `http://scholar.google.com`

Remark that the existing techniques for active Dolev-Yao intruder model (e.g. [RT01]) can be easily adapted to solve the problem in these settings.

Multiple intruders

Machiavellian intruders Multiple "Machiavellian" intruders model was considered in [SMC00]. These intruders (possessing different initial knowledge) have the same abilities as a Dolev-Yao one, but there is a limitation on messages they can emit. The motivation of such a model is natural: the Machiavellian intruders are not interested to send long-term secrets to the network (since other intruders can intercept them), and messages that intruders can send should look like (have the same *skeleton* as) legitimate messages of the protocol.

Moreover, the authors also showed an equivalence between multiple communicating Dolev-Yao intruders and a unique one.

Multi-Attacker In [ABCC11] the authors presented a Multi-Attacker threat model, which can be briefly described as follows: each participant may behave as a Dolev-Yao attacker, but will never reveal his long-term secrets. We can see that it resembles the Machiavellian intruders, but the restriction on messages the attackers can send is relaxed: an intruder is not obliged to send only messages that match legitimate ones.

In other words, the participants follow the normal course of the protocol but in parallel may act as DY attackers (with the only restriction not to send long-term secret).

The goal of the paper is to be able to find a new kind of attacks, like *retaliation attack*, where a protocol participant being a victim of an attack by another one may revenge and proceed a counter-attack against the offender, or even anticipate it before the initial attack completes. This may be possible due to the deviation from the ordinary execution of the protocol when preparing a first attack.

The authors tried to encode such settings in a way to be able to use existing tool for protocol analysis with DY intruder. First, in order to enforce the requirements about long-term secrets, an agent is not allowed to construct and send messages that contain long-term secrets. Second, since the considered tool works only with bounded sessions protocol analysis, the proposed approach has other limits. The most important is that the agents may execute only a bounded number of DY operations (like encryption/decryption). This is a "step backwards" with regard to the DY intruder, since the state-of-the-art techniques allow to solve protocol insecurity problems where the intruder is not limited in the number of operations he can apply on the messages he knows.

We would like to note that using our technique of general deducibility constraints satisfiability we may relax a restriction on the number of operations performed by the agents for the case of retaliation attacks on secrecy. However we do not give the details, since the formal model of the appropriate technique can be described in the same spirit as presented in Sections 4.3 and 5.3.

Colluding participants A problem about how to know whether a set of colluded participants in the protocol can derive a data that is designed to be hidden from them is tackled in [LM96]. The protocol is described as a transition system:

- finite set of states, where a state defines knowledge of each participant,
- finite set of transition labels, represented by a sender, receiver and message allowed to be sent, and
- transition partial function that maps a state and a transition label to another state.

The transition function is partial, since the authors suppose that a user cannot send a message to another if he does not know his name or is unable to build the message from his current knowledge. Once a message sent, the receiver's knowledge is increased with the received message.

Message is represented by an atomic value (an element from one of the finite sets of keys, user names or data) that can be encrypted in order by some keys. Once a participant receives a message, he decrypts it as much as he can and then can use it in further communication. Every message sent by a user has its own unique identifier which is transmitted together with it.

At some point the colluding participant can start to communicate in order to derive a sensitive data. The possible communication is constrained by a transition system of collusion process similar to one that describes the protocol, except the fact that the colluding users send *all* their knowledge when communicate (thus, no need of allowed message in transition labels).

The collusion problem is stated as follows: given a transition system of collusion process Ψ, its initial state w and a set of sensitive data T, it is asked whether exists a valid path ρ (called *collusion path*) in Ψ starting from w and finished on some state c in which some colluding user possesses the information T.

Two important assumptions are made: w must correspond to some reachable state in the protocol transition system and the order in which a user receives information is immaterial (recall that a user decrypt everything he can in the newly received message). The authors proposed an algorithm that solves this problem and returns a collusion path if one exists.

Non-communicating intruders We propose to consider a model with multiple "local" non-communicating intruders each with Dolev-Yao active intruder capabilities, but controlling only several communication channels. Moreover these intruders cannot communicate during the protocol execution, but only then when they quit the protocol. This model may be motivated by a case where a spy succeeded to install several small devices on parts of deployed network (e.g. network cables) in some organization such that each device controls their own communication channels, but have no means to communicate directly due to network configuration and physical conditions. After some time the spy can return to this organization and get back his devices. Then he is able to read and join the collected information to derive some secrets (note that the devices are not only passive eavesdroppers, but can affect the protocol execution).

In these settings the active Dolev-Yao intruder is a special case of such model.

5.1.2 From protocol sessions to constraints solving

Here we recall an approach for deciding cryptographic protocols insecurity problem with bounded number of sessions in the presence of an active Dolev-Yao intruder, since a model we will present for the case of multiple non-communicating intruders model can be seen as its extension.

As we mentioned in the introductory part, the protocol is usually described by a set of roles that can be played by some agents. Role has as parameter at least an agent who will play it, and may also have other parameters, like, e.g., public and private keys of the agent or some shared with another agent key, etc. If we instantiated all the roles of the protocol by assigning some values to the roles' parameters, we obtain a *protocol session*.

The problem we consider here assumes only protocol sessions, but not protocol in general. That is, if one finds no attack assuming some given set of protocol sessions, we cannot guarantee, that the protocol preserves some security properties in the case where we can run as many instances of the protocol as we want in parallel. However, if an attack is detected, then the protocol is vulnerable.

The first works that introduce symbolic constraints to solve the insecurity problem of cryptographic protocols are [MS01] and [CV01]. Moreover, in these works an efficient technique for solving the NP-complete deducibility constraint systems satisfiability problem was presented. Later, this technique was implemented in CL-AtSe [Tur06] and generalized in, e.g., [CLS03, BDC09, CDM11]. Below we discuss an example how to reduce a secrecy problem of a given protocol session to the resolution of well-formed constraint system.

Let us consider again a classical Needham-Schroeder Public Key protocol (NSPK) [NS78]. Alice (A) sends to Bob (B) a pair of her name and a nonce (N_A) encrypted with Bob's public key (K_B). Bob replies with a pair of received nonce and one he has just generated (N_B) encrypted with Alice's public key (K_A). Then, Alice sends back to Bob his nonce encrypted with K_B. In "Alice-Bob" notation the protocol looks like:

$$\begin{aligned} A \to B: &\quad \mathrm{aenc}\left(\mathrm{pair}\left(A, N_A\right), K_B\right) \\ B \to A: &\quad \mathrm{aenc}\left(\mathrm{pair}\left(N_A, N_B\right), K_A\right) \\ A \to B: &\quad \mathrm{aenc}\left(N_B, K_B\right) \end{aligned}$$

The purpose of this protocol is to mutually authenticate the participants, since the correspondence between an agent's name and his public key is supposed to be known by everyone (e.g. thanks to a trusted Certificate Authority). If B replies to A in the second step, that means, he was able to decrypt a received message in the first step (as N_A is only known by A), and thus, he possesses a private key $\mathrm{priv}\left(K_B\right)$, i.e. it was exactly B who produced this message. On the other hand, if A was able to extract N_B to send it in the third step, then, using the same reasoning, B can be sure that the one who sent this message knows $\mathrm{priv}\left(K_A\right)$. Moreover, N_B can be used now as shared secret, since all messages are encrypted either by public key of A or of B, and thus, even if someone except A and B overheard the communication, cannot know N_B — data generated by B and received by A in the last step.

From this conversational specification and participant's initial knowledge, we can pass to a sequence of actions performed by every participant. This passage is called *projection*. For A this sequence may look like (where its initial knowledge is given in brackets):

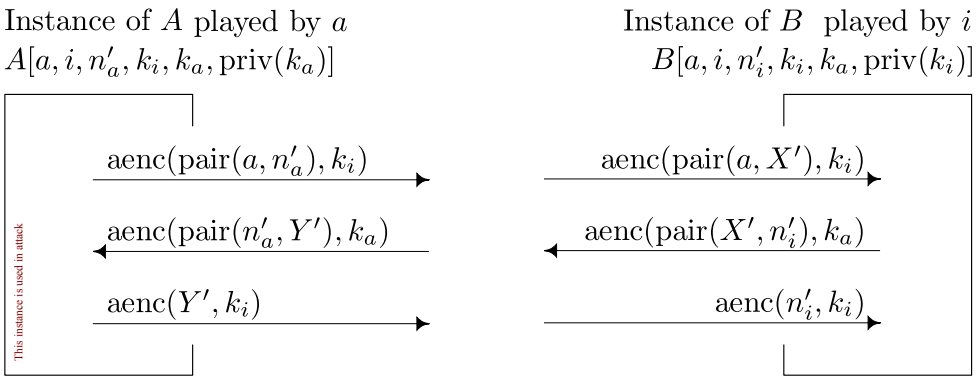
$$\begin{gathered} A[A, B, N_A, K_B, K_A, \mathrm{priv}\left(K_A\right)]: \\ !_B\, \mathrm{aenc}\left(\mathrm{pair}\left(A, N_A\right), K_B\right).\, ?_B\, \mathrm{aenc}\left(\mathrm{pair}\left(N_A, Y\right), K_A\right).\, !_B\, \mathrm{aenc}\left(Y, K_B\right), \end{gathered}$$

and for B:

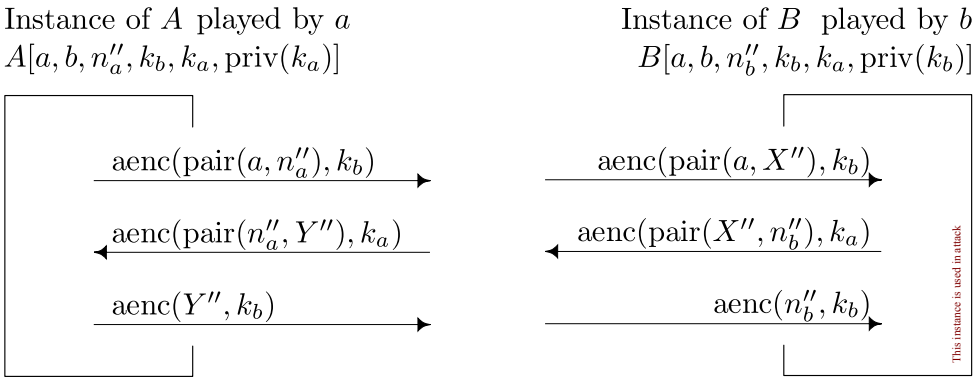
$$\begin{gathered} B[A, B, N_B, K_B, K_A, \mathrm{priv}\left(K_B\right)]: \\ ?_A\, \mathrm{aenc}\left(\mathrm{pair}\left(A, X\right), K_B\right).\, !_A\, \mathrm{aenc}\left(\mathrm{pair}\left(X, N_B\right), K_A\right).\, ?_A\, \mathrm{aenc}\left(N_B, K_B\right). \end{gathered}$$

These two sequences are called *protocol roles* [CDL+99], parametrized entities. For example, role for A can be instantiated with an agent who plays it (give a value to A), its public and private keys (value of K_A that implies value of $\mathrm{priv}\left(K_A\right)$), his interlocutor (value for B) and its public key (value for K_B); moreover, a fresh value for a nonce (N_A) can be also given (since it is a sequence and there is no need to dynamically generate its value during the execution of this role). Note that variables X and Y become known only during the execution of the protocol (A cannot know in advance the fresh value generated by B), and their values supposed to be $X = N_A$ and $Y = N_B$.

If we instantiate all roles of the protocol with the corresponding values (variables with the same name should take the same values in all the roles), we obtain a *protocol session* which consists of instances of protocol's roles.



(a) A session between a and i



(b) A session between a and b

Figure 5.1: Two sessions of NSPK protocol

As we mentioned above, we will consider a given finite set of instances of protocol roles to check for the insecurity problems. This is a research branch in protocol analysis in contrast to the methods that do not consider a bound on number of session one have to test. The advantage of this approach is that the assumed restrictions admit more decision procedures (that are correct, complete and terminate). However, it mostly should be used for *finding attacks* rather than *prove security properties.*

Let us return to NSPK and consider the following instances: an instance of A's role played by a of a session with some compromised entity i (which is an intruder who is also a legitimate network user) (Figure 5.1a), and an instance of B's role played by b of a session with an honest agent a (Figure 5.1b). These two instances are enough to perform an attack. Note that this is quite realistic scenario, where a malicious user i with whom the honest agent a initiated a session of the NSPK protocol, reuse the messages of this session in order to establish a new session with honest b where he impersonates a.

From the point of view of b (following the protocol steps), he is talking to a, and if the protocol finishes successfully, b is assured that he talks to a and moreover, n''_b is a secret value that is known only by him and a. After that, for example, n''_b can be used as a symmetric key to encrypt some secret data that must be sent to a.

Let us try to find out, whether in this scenario active Dolev-Yao intruder i can learn secret

$$\begin{cases} K_0 \cup \{\mathrm{aenc}(\mathrm{pair}(a, n'_a), k_i)\} \triangleright & \mathrm{aenc}(\mathrm{pair}(a, X''), k_b) & (5.1) \\ K_0 \cup \{\mathrm{aenc}(\mathrm{pair}(a, n'_a), k_i), \mathrm{aenc}(\mathrm{pair}(X'', n''_b), k_a)\} \triangleright & \mathrm{aenc}(\mathrm{pair}(n'_a, Y'), k_a) & (5.2) \\ K_0 \cup \{\mathrm{aenc}(\mathrm{pair}(a, n'_a), k_i), \mathrm{aenc}(\mathrm{pair}(X'', n''_b), k_a), \mathrm{aenc}(Y', k_i)\} \triangleright & \mathrm{aenc}(n''_b, k_b) & (5.3) \\ K_0 \cup \{\mathrm{aenc}(\mathrm{pair}(a, n'_a), k_i), \mathrm{aenc}(\mathrm{pair}(X'', n''_b), k_a), \mathrm{aenc}(Y', k_i)\} \triangleright & n''_b & (5.4) \end{cases}$$

Figure 5.2: A constraint system expressing a possible attack on NSPK protocol.

n''_b using a symbolic constraints approach[32]. We suppose that the intruder knows initially

$$K_0 = \{i, a, b, n_i, k_i, \mathrm{priv}(k_i), k_a, k_b\},$$

that is the initial knowledge of the compromised agent (see instance of B, Figure 5.1a) and names and public keys of a and b.

We have 3 actions for each considered instance. We must must try all possible ordering for the intruder in which he sends messages to the honest agents. Since this number is finite (in general thanks to bounded number of sessions) we can test them all. First we remark that if a player can send a message (like a in Figure 5.1a), he will do it immediately, and this message becomes a part of intruder's knowledge thanks to his ability to intercept the messages.

One of the four possibilities[33] is the following:

$$(a)\ !\,\mathrm{aenc}(\mathrm{pair}(a, n'_a), k_i); (b)\ ?\,\mathrm{aenc}(\mathrm{pair}(a, X''), k_b); (b)\ !\,\mathrm{aenc}(\mathrm{pair}(X'', n''_b), k_a);$$
$$(a)\ ?\,\mathrm{aenc}(\mathrm{pair}(n'_a, Y'), k_a); (a)\ !\,\mathrm{aenc}(Y', k_i); (b)\ ?\,\mathrm{aenc}(n''_b, k_b).$$

Following it, we build a system of deducibility constraints (Figure 5.2) such that if it is satisfiable, then the protocol admits an attack.

After reception of first message from a, the intruder i tries to produce a message that must be accepted by b (Constraint 5.1). Then he collect his response and tries to send a message to a (Constraint 5.2). Constraint 5.3 states that from the collected knowledge i tries to send a message to b in order to b completes his role's execution and thus thinks that he is communicating with a and shares with him a secret n''_b. Meanwhile, in Constraint 5.4 we check whether after these steps the intruder i can infer this secret value.

This constraint system is well-formed: left-hand sides of the constraints increases, and variables (here X'' and Y') appears first in right hand sides of some constraints.

From Part I as well as from, e.g., [MS01, RT01, CLS03, Tur06] we can see that the resolution of this constraint system is decidable, and we can find its solution $\sigma = \{X'' \mapsto n'_a, Y' \mapsto n''_b\}$. This solution gives us an attack depicted in Figure 5.3 (all operations performed by the intruder i are shown in gray text). That is, the intruder may use this scenario to get known a secret n''_b while b will think that he communicated with a and n''_b is known only by a and himself (b).

We have just shown an instance of procedure for reducing a secrecy problem of the given protocol sessions to the satisfiability of well-formed constraints. In the next section we will present an example, where the solving well-formed constraints is not enough and one must relax well-formedness property.

[32] We recall that the problem of protocol insecurity with bounded number of sessions is NP-complete [RT01].

[33] Note that one should consider also the case where the intruder does complete execution of a's instance, i.e. do not send anything to it

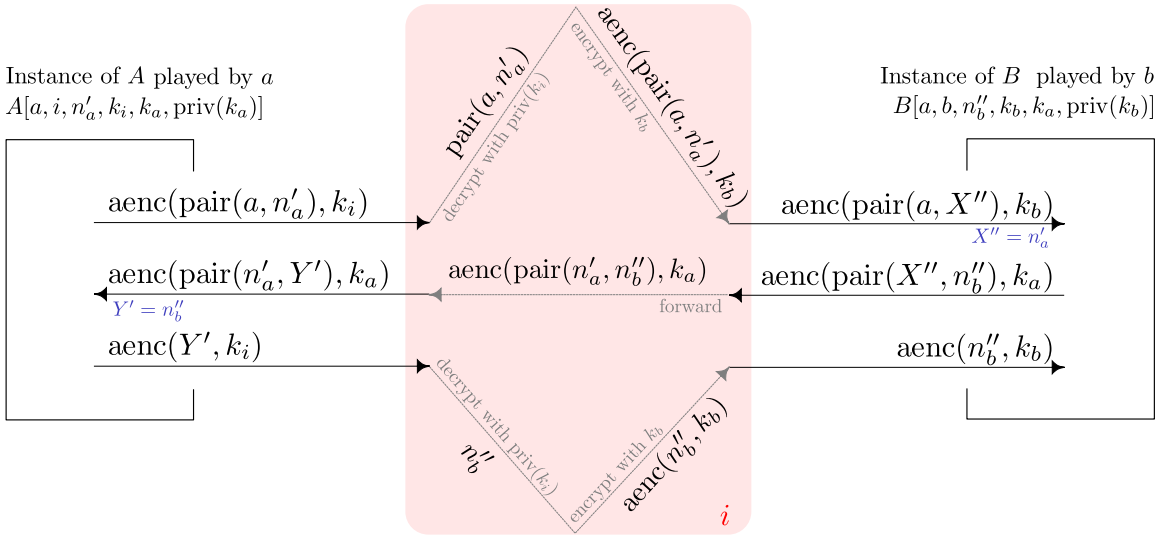


Figure 5.3: An attack to NSPK protocol

5.2 Motivation

In the domain of security protocol analysis Dolev-Yao model is widely used in spite of its limitations. We propose here to consider instead of a powerful intruder that controls the whole network, several non communicating intruders with smaller controlled domains. We give below an application of this model.

Suppose several agents ($A, B \ldots$, see Figure 5.4) execute a message exchange protocol (every agent has a finite list of actions in a send/receive format that is known to everybody). Due to their (long distance) layout they have to transmit data through routers ($1, 2, 3 \ldots$). The routing tables of all honest routers/agents are static (messages follow always the same path). Some routers ($2, 5, 7$) may be compromised: an intruder managed to install a device controlling input and output of the router or implanted there his malicious code. A message circulated via such an untrusted channel (e.g. DB) is consumed by the corresponding compromised device (*local intruder*) (7) thereby increasing his knowledge. Moreover, a local intruder can forge and emit to an endpoint (C, B, D) of any channel he controls (BD, DB, DC) any message he can build using the content of his memory and some available transformations specified by a deduction system. Because of the network topology malicious routers have no means to communicate (there is no links between them, neither direct nor via other routers), but at some point the intruder can gather the knowledge of all the compromised routers (by physically collecting devices and reading their memory).

More generally the problem input is a set of agents each given with a list of actions it is supposed to execute. Every agent has communication channels with other ones. We bind to every communication channel at most one active "local" intruder (several channels can be controlled by the same intruder, some can be free from intruders). This binding is also given and is normally conditioned by the feasibility of deploying local intruders into the network channels. Every local intruder has its initial knowledge. Intruder controlling a channel can intercept a message passed on it thereby increasing his knowledge, and forge a message (anything he can build from his current knowledge) to send to the endpoint of any channel controlled by him. Local intruders cannot exchange messages during protocol execution but only afterwards. The question posed is, given a set of secret data, whether there exists a sequence of message exchanges

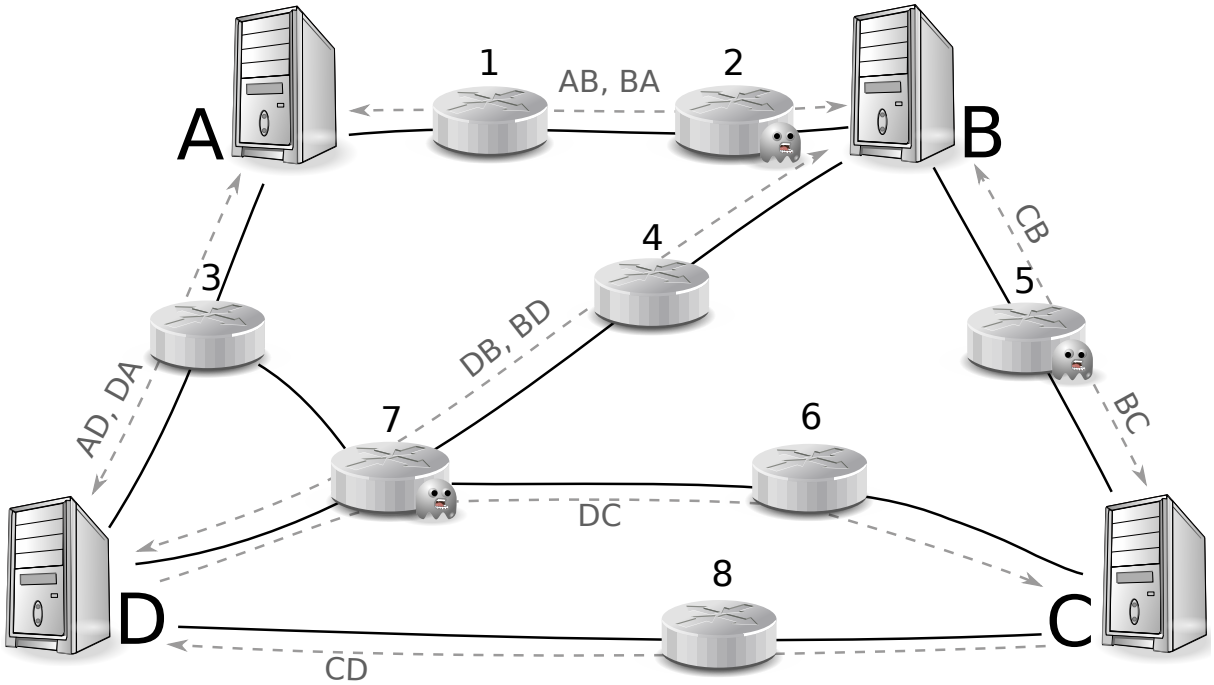


Figure 5.4: Untrusted routers

Distributed orchestration	Protocol insecurity with multiple intruders
Available services/Client	Protocol session (honest agents)
Mediator and Partners	Local intruders
Accessibility of a service	A channel being under control
Invoke a service	Intercept and/or emit a message on a channel

Table 5.1: Some analogies between formal models of Chapter 4 and 5.

between honest agents (which strictly follow the list of actions) and local intruders such that at some point, from common knowledge of all local intruders it is possible to deduce a secret data.

In this framework the *security problem* is to know whether it is possible to initially give instructions to compromised routers to force such an execution that honest agents (that strictly follow their list of actions) will reveal some secret data to the intruder (i.e. intruder can build this data from the gathered at the end knowledge of all local intruders).

5.3 Formal model

We follow the similar scheme as for *available services* presented in Chapter 4 (see page 73) to represent the behavior of the honest participants of the protocol session: an honest agent has a sequence of actions to execute. One may also draw a parallel between local intruders (compromised routers in the example given above) and partners of the Web Services distributed orchestration, since they have the similar capabilities to perform on messages; moreover, one may observe some kind of similarity between invocation of accessible available services and emitting or intercepting messages on the channels under control. These analogies are summarized in Table 5.1.

Agents

We will call communicating parties *agents*. Every agent is identified by his name. We denote a set of agent names as A. Every agent has a finite list of actions (see Agent behavior).

Channels

Between two any agents a and b there exists a communication channel which we denote as $a \rightharpoonup b$. We will suppose that channels are directed. The set of all channels is denoted as $\mathbb{C}$.

A channel supports a queue of messages: for example, if a sends sequentially two messages to b (via channel $a \rightharpoonup b$), then b cannot process the second message before the first one; messages are stored in queue to be processed in order of arrival.

Agents behavior

We define a protocol session $PS = \{\langle a_i, l_i\rangle\}_{i=1,\dots,k}$ as a set of pairs of an agent name and a finite list of actions to be executed by this agent[34]. We also suppose that $\mathrm{Vars}(l_i) \cap \mathrm{Vars}(l_j) = \varnothing$, for all $i \neq j$ (where $\mathrm{Vars}(\cdot)$ is naturally extended on lists of actions).

We recall a definition of action introduced in Chapter 4, see Definition 4.1.1 at page 73. Every action is of receiving type $?_f r$ or sending type $!_t s$ where

- f is an agent name, whom a message is to be received from;
- r is a term (a template for the message) expected to be received from f;
- t is an agent name, whom the message is expected to be sent to;
- s is a term (a template for the message) to be sent to t.

Let us take any agent $a \in A$ participating in the protocol session PS and let $\left\langle a, \{\rho_i\}_{i=1,\dots,k}\right\rangle \in PS$.

Case 1. If $\rho_1 = ?_{f_1} r_1$ then the first action agent a can do, is to accept a message m, admittedly from agent f_1 on channel $f_1 \rightharpoonup a$, matching the pattern r_1, i.e. such that $\ulcorner r_1\sigma \urcorner = \ulcorner m \urcorner$ for some normalized substitution σ[35] with $\mathrm{dom}(\sigma) = \mathrm{Vars}(r_1)$. Agent is blocked (does not execute any other actions) by awaiting a message. If a receives a message that does not match the expected pattern, then a terminates his participation in PS (the explanation is simple: a participant assumes he face an incident if a badly formed message is received). Note that no notification is sent to the sender, thus a sender continues his execution[36]. Once a has received message m matching the pattern r_1 with substitution σ, he instantiate $\mathrm{Vars}(r_1)$ with σ and execute his remaining actions using these values, i.e. a moves to a state where the list of actions to be executed is $\{\rho_i\sigma\}_{i=2,\dots,k}$.

[34] For simplicity, we suppose that for a protocol session, one agent can not have more than one list of actions to execute, but this restriction can be relaxed.

[35] Note here a non-deterministic behavior in the case where multiple unifiers σ are possible. As an example, for considered in this work ACI theory, if $r_1 = \cdot(\{X, Y\})$ and incoming message is $m = \cdot(\{a, b, c\})$, then 6 different possibilities exist for σ.

[36] A way to model another behavior, is to explicitly provide for every sending a succedent receive of an acknowledge message and for every receive a succedent send of an acknowledge message.

Case 2. If ρ_1 is $!_{t_1} s_1$ then the first action of agent a is sending message s_1 to agent t_1 (i.e. putting it to channel $a \rightharpoonup t_1$) and then, moving to a state where $\{\rho_i\}_{i=2,\dots,k}$ has to be executed.

We assume a usual that agents cannot have a sending pattern that contains variables not instantiated before, i.e. for any $\langle a, \rho_1 . \cdots . \rho_{k_a} \rangle \in PS$ if $\rho_i = !_t s$ then for any variable $x \in \mathrm{Vars}(s)$ there exists $j < i$ such that $\rho_j = ?_f r$ and $x \in \mathrm{Vars}(r)$.

Intruder model

We assume that some communication channels are controlled by N *local* intruders $\mathbb{I} = \{I_i\}_{i=1,\dots,N}$ and there is no channel controlled by more than one intruder. Still there can be channels free from any intruder.

We introduce an *intruders layout* represented by a function $\iota : \mathbb{C} \mapsto \mathbb{I} \cup \{\varnothing\}$ mapping every channel to the local intruder that controls it if there is one, and maps to $\varnothing$ otherwise.

Every intruder I has some initial knowledge K_I^0 that is a set of ground terms.

Once an agent sends a message via a channel controlled by an intruder, the intruder intercepts it, that is reads and blocks it. Reading the message means extending intruder's current knowledge with this message and blocking means preventing the message to achieve its initial recipient.

An intruder controlling a channel can generate a message from his knowledge using his deduction power and send it to the endpoint of this channel.

We specify the intruder's capability to forge messages as follows: local intruder I can send a message m, if $m \in \mathrm{Der}(K_I)$, where K_I is a current knowledge of intruder I.

Protocol session execution

Now, having a protocol session, intruders layout and their initial knowledges, we can present a course of a protocol execution. To do this, we will first introduce a *symbolic* execution, where data exchanged among the agents and intruders is not instantiated and represented as terms (possibly non-ground, i.e. that may contain variables). This execution is constrained by some conditions. Whenever these conditions are fulfilled with an appropriate ground instantiation of variables (i.e. by some ground substitution), we obtain a *concrete* execution (or simply, an *execution*). These conditions are defined by described earlier constraint systems (see § 1.2, p. 22).

Definition 5.3.1 A configuration Π of a protocol session PS is a quadruple $\langle PS, \mathcal{K}, \mathcal{Q}, \mathcal{S} \rangle$, where $\mathcal{K} = \{\langle I_i, K_i \rangle\}_{i=1,\dots,N}$ represents current knowledges of intruders, and $\mathcal{Q} = \{\langle c, m_c \rangle\}_{c \in \mathbb{C}}$ is a configuration of channels: for every channel c a queue of messages m_c is given. $\mathcal{S}$ is a deducibility constraint system.

Transitions on configurations are defined in Table[37] 5.2 and will be explained later. Transitions are written in form $\Pi_1 \xrightarrow{cond} \Pi_2$ and state that configuration Π_1 can evolve to a new configuration Π_2 if condition *cond* is satisfied.

Definition 5.3.2 A *symbolic execution* E_{PS}^S of protocol session PS (with intruders layout ι) is a sequence of configurations obtained by application of transitions to the initial configuration $\left\langle PS, \left\{\left\langle I, K_I^0 \right\rangle\right\}_{I \in \mathbb{I}}, \{\langle c, \varnothing \rangle\}_{c \in \mathcal{C}}, \varnothing \right\rangle$.

[37] $\uplus$ represents the union of two disjoint sets: $A \uplus B = A \cup B$ iff $A \cap B = \varnothing$.

1.	$\left\langle \left\{ \left\langle a, (?_f r).l_a \right\rangle \right\} \uplus PS, \{\langle I, K\rangle\} \uplus \mathcal{K}, \mathcal{Q}, \mathcal{S} \right\rangle \xrightarrow{\iota(f \rightarrowtail a)=I}$ $\left\langle \{\langle a, l_a\rangle\} \cup PS, \{\langle I, K\rangle\} \cup \mathcal{K}, \mathcal{Q}, \mathcal{S} \cup \{K \triangleright r\} \right\rangle$
2.	$\langle \{\langle a, (!_t s).l_a\rangle\} \uplus PS, \{\langle I, K\rangle\} \uplus \mathcal{K}, \mathcal{Q}, \mathcal{S}\rangle \xrightarrow{\iota(a \rightarrowtail t)=I}$ $\langle \{\langle a, l_a\rangle\} \cup PS, \{\langle I, K \cup s\rangle\} \cup \mathcal{K}, \mathcal{Q}, \mathcal{S}\rangle$
3.	$\langle \{\langle a, (!_t s).l_a\rangle\} \uplus PS, \mathcal{K}, \{\langle a \rightharpoonup t, m_{a \rightarrowtail t}\rangle\} \uplus \mathcal{Q}, \mathcal{S}\rangle \xrightarrow{\iota(a \rightarrowtail t)=\varnothing}$ $\langle \{\langle a, l_a\rangle\} \cup PS, \mathcal{K}, \{\langle a \rightharpoonup t, m_{a \rightarrowtail t}.s\rangle\} \cup \mathcal{Q}, \mathcal{S}\rangle$
4.	$\left\langle \left\{ \left\langle a, (?_f r).l_a \right\rangle \right\} \uplus PS, \mathcal{K}, \left\{ \left\langle f \rightharpoonup a, s.m_{f \rightarrowtail a} \right\rangle \right\} \uplus \mathcal{Q}, \mathcal{S} \right\rangle \xrightarrow{\iota(f \rightarrowtail a)=\varnothing}$ $\left\langle \{\langle a, l_a\rangle\} \cup PS, \mathcal{K}, \left\{ \left\langle f \rightharpoonup a, m_{f \rightarrowtail a} \right\rangle \right\} \cup \mathcal{Q}, \mathcal{S} \cup \{\{\mathrm{enc}(s,k)\} \triangleright \mathrm{enc}(r,k)\} \right\rangle$

Table 5.2: Configuration transitions

For a substitution σ and a configuration Π

$$\Pi = \left\langle \{\langle a_i, l_i\rangle\}_{i=1,\ldots,k}, \{\langle I_i, K_i\rangle\}_{i=1,\ldots,N}, \{\langle c, m_c\rangle\}_{c \in \mathbb{C}}, \{E_i \triangleright t_i\}_{i=1,\ldots,n} \right\rangle$$

we define $\Pi\sigma$ as

$$\Pi\sigma = \left\langle \{\langle a_i, l_i\sigma\rangle\}_{i=1,\ldots,k}, \{\langle I_i, K_i\sigma\rangle\}_{i=1,\ldots,N}, \{\langle c, m_c\sigma\rangle\}_{c \in \mathbb{C}}, \{E_i\sigma \triangleright t_i\sigma\}_{i=1,\ldots,n} \right\rangle,$$

where substitutions are applied to lists elementwise.

Definition 5.3.3 A (*concrete*) *execution* $E_{PS} = \{C_i\sigma\}_{i=1,\ldots,m}$ is an instance of a symbolic execution $\{C_i\}_{i=1,\ldots,m}$ (where $C_i = \langle PS_i, \mathcal{K}_i, \mathcal{Q}_i, \mathcal{S}_i\rangle$) such that all terms of $C_i\sigma$ are ground and $\mathcal{S}_m$ is satisfied by σ.

Now we describe the transitions of Table 5.2. Transition 1 expresses the possibility of intruder I controlling channel $f \rightharpoonup a$ to impersonate f and send to a some message compliant with the expected by a pattern r, if the current knowledge of I allows it. An intruder can also intercept messages sent on the channel that he controls (Transition 2). A message sent by an agent on the channel free from intruders is put to the end of the queue of this channel (Transition 3). Transition 4 represents the reading of a message by an agent from the queue of the channel.

Let us explain where the constraint $\{\mathrm{enc}(s,k)\} \triangleright \mathrm{enc}(r,k)$ comes from in the last transition. Agent a expects to read a message from the channel compatible with the pattern r. The first (possibly not yet instantiated) message in the queue is s. Thus, r and s must be unifiable (modulo considered equational theory), and even equivalent when we consider ground instances of the symbolic executions. Since we will be interested only in concrete executions, we can use this constraint to express equivalence between r and s (Lemma 22) at least for DY and DY+ACI theories.

Lemma 22. *For DY and DY+ACI theories, for terms t_1, t_2 and substitution σ, $\ulcorner t_1\sigma \urcorner = \ulcorner t_2\sigma \urcorner$ is true iff σ is a model of $\{\mathrm{enc}(t_1,k)\} \triangleright \mathrm{enc}(t_2,k)$ for any term k, i.e. $\ulcorner t_1\sigma \urcorner = \ulcorner t_2\sigma \urcorner$ iff $\ulcorner \mathrm{enc}(t_1,k)\,\sigma \urcorner \in \mathrm{Der}(\{\ulcorner \mathrm{enc}(t_2,k)\,\sigma \urcorner\})$.*

Offline communication

At some point the current knowledge of all local intruders can be shared to derive a secret which probably they cannot deduce separately. In some cases these offline interactions are time-consuming and may be detected. Therefore we consider reasonable that in the intruder strategy modeling they take place after the protocol is over.

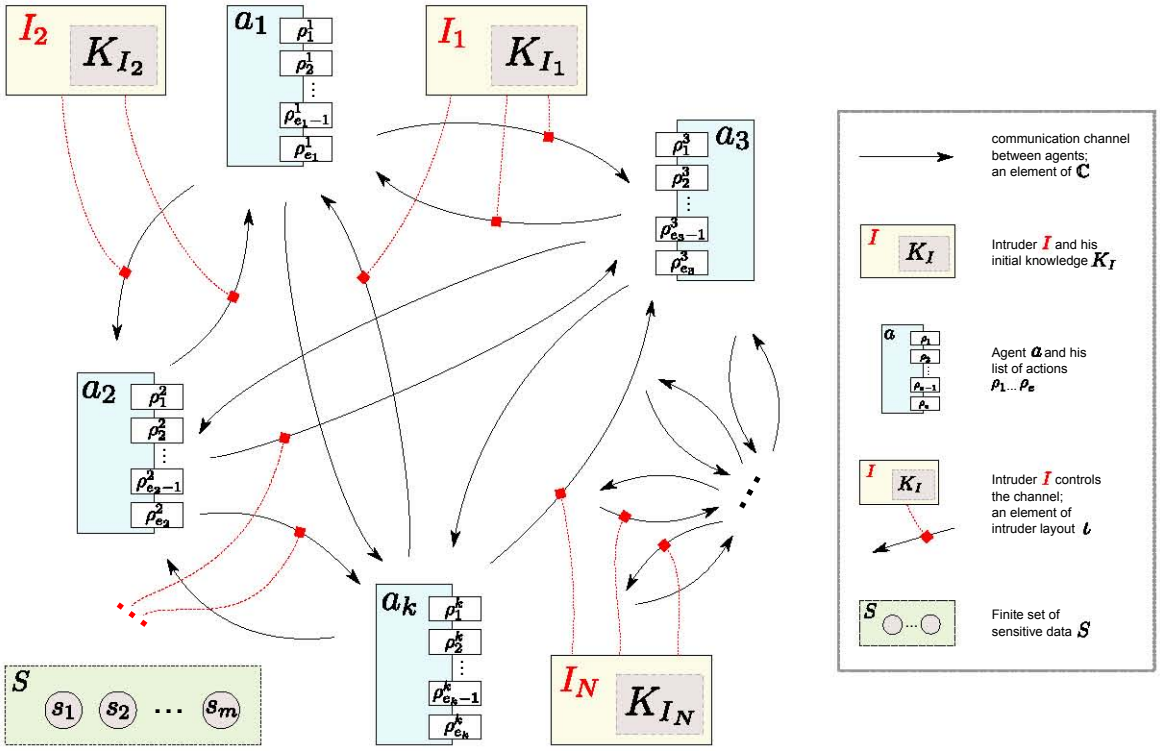


Figure 5.5: Generic input example for the coordinated attack problem

Coordinated attack problem

Now we can formally state the problem.
Input: A finite set of agents A and a protocol session $PS = \{\langle a_i, l_i\rangle\}_{i=1,\dots,k}$, a set of intruders $\mathbb{I} = \{I_i\}_{i=1,\dots,N}$ each with initial knowledge $K^0_{I_i}$, an intruder layout ι and some sensitive data given as a finite set of ground terms S.
Output: A pair $s \in S$ and an execution E_{PS} of protocol session PS with its last configuration $\langle PS, \mathcal{K}, \mathcal{Q}, \mathcal{S}\rangle$ such that $s \in \mathrm{Der}\left(\bigcup_{\langle K_I, I\rangle \in \mathcal{K}} K_I\right)$, if such a pair exists.

A scheme of the generic input for coordinated attack problem is illustrated in Figure 5.5.

5.4 Solution approach

As was announced before, and, being evident from the problem statement, we will reduce the coordinated attack problem to resolution of deducibility constraint systems. For this we propose a non-deterministic algorithm.

First, we can guess a sensitive value s on whose secrecy we will check a possibility of an attack. We can do it, since the set of sensitive data S is finite. Then, we can guess how many actions we need to be executed by agents in order to achieve an attack on the guessed sensitive value. This number, let's say, w, is bounded by the total number of actions for all agents of the given protocol session, i.e. $\sum_{\langle a,l\rangle \in PS} \mathrm{length}(l)$. Having w, we may guess a symbolic execution E^S of length w. Again, the number of possible symbolic executions with given initial state is finite, since every application of a rule from Table 5.2 decreases a total number of actions that all agents can perform. Note that a check whether a sequence of configurations of length l forms a symbolic execution can be done in polynomial time. Finally, once we have a guessed symbolic

execution with some constraint system $\mathcal{S}$ and the intruder knowledges $\mathcal{K}$ in its last configuration, we will solve constraint system $\mathcal{S} \cup \left\{\bigcup_{\langle K_I, I\rangle \in \mathcal{K}} K_I \rhd s\right\}$. Here S corresponds to the conditions that must be satisfied in order to make symbolic execution concrete, and $\bigcup_{\langle K_I, I\rangle \in \mathcal{K}} K_I \rhd s$ expresses the fact that secret data s can be derived from the total knowledge of all the local intruders on the moment they quit the protocol execution. If such constraint system is satisfiable with some σ, then $E^S\sigma$ is the sought execution.

Summing up, we proceed as follows:

1. Guess a sensitive datum s from S.
2. Guess a sequence of configurations that represent a symbolic execution E^S_{PS} of some length $\leq \sum_{\langle a, l\rangle \in PS} \text{length}(l) < \infty$.
3. For the last configuration $\langle PS, \{\langle K_I, I\rangle\}_{I \in \mathbb{I}}, \mathcal{Q}, \mathcal{S}\rangle$ of E^S_{PS}, we guess a normalized solution σ of constraint system $\mathcal{S}' = \mathcal{S} \cup \left\{\bigcup_{\langle K_I, I\rangle \in \mathcal{K}} K_I \rhd s\right\}$ such that $\text{size}(x\sigma) \leq 2 \times \text{size}(\mathcal{S}')$ for every $x \in \text{Vars}(\mathcal{S}')$.
4. Check that σ satisfies $\mathcal{S}'$, and if yes, then the protocol session is insecure and we return $E_{PS} = E^S_{PS}\sigma$.

The obtained execution gives a scenario of an attack against which the given protocol session is vulnerable.

Theorem 5. *Decidability of the coordinated attack problem is in NP.*

Proof idea. First, we note that we can check that a sequence of configurations forms symbolic execution in polynomial time. Second, we note that the DAG-size of a constraint system in the last configuration of E^S_{PS} (and thus $\mathcal{S}'$) is polynomially bounded by the size of the problem input. Finally, reusing the (reasoning for) Proposition 14 we have that the check whether the guessed σ satisfies $\mathcal{S}'$ can be done in polynomial time. Thus, the non-deterministic algorithm summarized above is in NP.

5.5 XML rewriting attacks

XML [W3C08] is a widely used format that became a base of hundreds languages. One of them is a message format used in SOAP protocol [Con07], a protocol used for Web Services communication. Since security standards for Web Services like WS-SecurityPolicy are applied to XML messages, it is reasonable to consider properties of XML representation in combination with security elements like encryption and signature.

While protocols that use XML can be vulnerable to the same attacks as classical ones, the new type of attacks called *XML rewriting attacks* [BFGP04, BFGO05] can be also applicable. As remarked in [CLR07], the XML format requires to consider set of terms to model message content rather than a term. Another point mentioned in the same work is that the parser of XML message may interfere since each partner may use its own implementation which can select different nodes as an answer to the same request. Thus, one should consider a non-deterministic behavior for XML parsing.

In our approach we take into account the mentioned points by considering ACI symbol as a set constructor, and since unification modulo ACI is in general not single-valued (e.g.

$\cdot(\{a,b,c\}) =_{ACI} \cdot(\{X,Y\})$ has 6 different solutions) the non-determinism of XML parsing is achieved.

Similarly to [CLR07] the presented technique can be considered in some sense as complementary to TulaFale tool [BFGP04] that uses ProVerif [Bla01] to verify security properties. While our approach is correct and complete assuming fixed sessions of a protocol, the ProVerif works with unbounded number of sessions, but may not terminate as well as may find false attacks.

Further, we present an example of a scenario vulnerable to an XML rewriting attack, a way the scenario can be modeled using our formalism in order to discovered the attack.

5.5.1 Example of discovering an attack exploiting XML format

Here we show how to model using our formalism attacks based on an XML-representation of messages. A different technique to handle this kind of attacks was presented in [CLR07].

Scenario

We consider an e-shop that accepts e-cheques, and we suppose that it is presented by a Web Service using SOAP protocol for exchanging messages.

It consists of two services:

- the first exposes the list of goods for sale with their prices and process orders by accepting payments,
- the second is a delivery service; it receives information from the first one about successfully paid orders, and sends the ordered goods to the buyer.

A simple scenario for ordering item is shown in Figure 5.6. First, a client sends an order using e-shop interface that consists of an item identifier, e-cheque, delivery address and some comments. Then, the first service of the e-shop checks whether the price of the ordered item corresponds to the received cheque. If it does, the service consumes the cheque and re-sends the order to the stock/delivery service (without the used e-cheque). Stock and delivery service prepare a parcel with ordered item and send it to given address. The comment is automatically printed on the parcel to give some complementary information about the delivery (like digital code to access the building, etc.).

Suppose, Alice has an e-cheque for 5€. She selected a simple pen (with ItemID *simple*) to buy, but she liked very much a more expensive top-quality gilded one (ItemID *gilded*). Unfortunately 5€ is too few. Can we help Alice to get what she wants for what she has?

Formalization

Let us formalize the behavior of scenario players (terms, normalization function and deduction system are defined as in § 2.1.1 except that we will write $(t_1 \cdot \ldots \cdot t_n)$ instead of $\cdot(\{t_1,\ldots,t_n\})$). Identifiers starting from a capital letter are considered as variables; numbers and identifier starting from lower-case letter are considered as constants.

We model a delivery of an item with some $ItemID$ to address $Addr$ with comments $Comm$ by the following message: $\mathrm{sig}((ItemID \cdot Addr \cdot Comm), \mathrm{priv}(k_s))$ — a signature produced by e-shop, where k_s its public key such that no one can construct this message except the shop. We abstract away from the procedure of checking price of the item and will suppose that Shop Interface expects 5€ e-cheque for Item "*simple*". For simplicity we assume only two items.

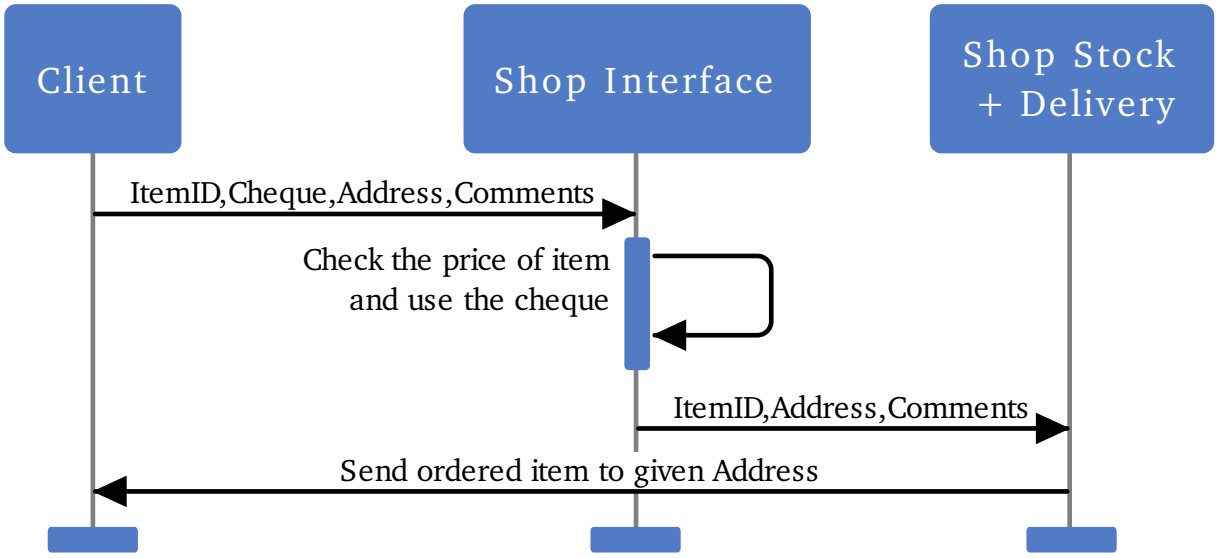


Figure 5.6: Ordering item scenario

For Shop Interface we have:

$$?_{Client}(simple \cdot cheque5 \cdot IAddr \cdot IComm);$$
$$!_{Delivery}(simple \cdot IAddr \cdot IComm).$$

For Shop Stock/Delivery we have:

$$?_{Interface}(DItemID \cdot DAddr \cdot DComm);$$
$$!_{Client}\,\mathrm{sig}\,((DItemID \cdot DAddr \cdot DComm), \mathrm{priv}\,(k_s))\,.$$

Alice initially has:

$simple, gilded$:	identifiers of items;
$cheque5$:	an e-cheque for 5€;
$addr$:	her address;
$cmnts$:	residence digital code;
k_s:	a public key of the shop.

Solution

Now we build a mixed constraint system (deducibility constraints and equations modulo ACI) to know, whether Alice can do what she wants (see Figure 5.7).

Constraint (5.5) shows that Alice can construct a message expected by the shop from a client. Equality (5.6) represents a request from the first to the second service of the shop: left-hand side is a message sent by the interface service, and right-hand side is a message expected by stock/delivery service. The last constraint shows that from the received values Alice can build a message that models a delivery of item with ItemID *gilded*.

Note that Equality (5.6) can be encoded as a deducibility constraint (see Lemma 22, p. 127) and then we obtain a system of deducibility constraint within DY+ACI theory considered in the first part of the thesis.

$$\left\{\begin{array}{ll} \{gilded, simple, cheque5, addr, cmnts, k_s\} \ \triangleright & \\ \qquad\qquad (simple \cdot cheque5 \cdot IAddr \cdot IComm) & (5.5) \\ (simple \cdot IAddr \cdot IComm) =_{ACI} \quad (DItemID \cdot DAddr \cdot DComm) & (5.6) \\ \{gilded, simple, cheque5, addr, cmnts, k_s, & \\ \mathrm{sig}\,((DItemID \cdot DAddr \cdot DComm), \mathrm{priv}\,(k_s))\} \ \triangleright & \\ \qquad\qquad \mathrm{sig}\,((gilded \cdot addr \cdot DComm), \mathrm{priv}\,(k_s)) & (5.7) \end{array}\right.$$

Figure 5.7: A constraint system describing possible vulnerability in E-shop scenario

Then we can find a solution for the obtained constraint system; one of them is:

$IAddr$	$\mapsto addr$	$IComm$	$\mapsto (gilded \cdot cmnts)$
$DItemID$	$\mapsto gilded$	$DAddr$	$\mapsto addr$
		$DComm$	$\mapsto (simple \cdot cmnts)$

From this solution we see that Alice can send unexpected comments (that presents two XML-nodes), and Delivery service parser can choose an entry with ID *gilded*, while a parser of the first layer can return a value of the first occurrence of ItemID, i.e. *simple*. An attack-request can look like this:

```
<ItemID>simple</ItemID>
<Cheque>cheque5</Cheque>
<Address>addr</Address>
<Comments>cmnts</Comments>
<ItemID>gilded</ItemID>
```

This attack is possible, if Alice constructs a request "by hand", but a similar attack is probably feasible using so-called XML-injection: Alice when filling a request form enters instead of her comments the following string:

```
cmnts</Comments>
<ItemID>gilded</ItemID><Comments>
```

and if the used application fails to make an appropriate data validation, then in the resulting request we get:

```
<ItemID>simple</ItemID>
<Cheque>cheque5</Cheque>
<Address>addr</Address>
<Comments>cmnts</Comments>
<ItemID>gilded</ItemID><Comments>
</Comments>
```

Of course, it is possible only if the input is not well sanitized. This kind of XML-injection attacks was described in [Fou08].

Thus, we can see how our formalism may help to detect vulnerabilities of communication systems whose messaging is based on XML.

5.6 Conclusions

In this chapter we considered a coordinated attack problem: a problem of protocol sessions insecurity under the new attacker model which assumes multiple non-communicating active local intruders each may control a set of communication channels used by honest agents — the participants of some protocol session. In this model at some point the knowledge is gathered from the local intruders in order to deduce a sensitive value.

We showed that the coordinated attack problem can be reduced to satisfiability of general deducibility constraint systems and thus, decidable.

We also showed how ACI symbol may be used for modeling a set of XML nodes in order to capture a class of attacks on protocols which use XML format for messaging, so-called XML-rewriting attacks.

Critics The presented attack on the assumed online shop exploits some non-uniform implementation of XML parsers. It is a hard assumption to rely on non-standard implementation of software. For example, if both services use different implementations of XPath standard [Wor07] for getting values of the necessary nodes, there should not be such problems.

Research directions One of the research directions is to study how the general constraints may be used in order to check security properties other than secrecy for the protocols.

Other possibility is to consider different algebraic properties, like XOR or modular exponentiation. While this extension mostly affects the resolution of deducibility constraint systems, we think it worth to be mentioned here.

Conclusion

Summary

In this thesis we studied one symbolic technique to deal with cryptographic primitives in the context of communication systems. We are talking about satisfiability of so-called *deducibility constraint systems*.

This technique has been deeply studied for the past 10 years in the context of cryptographic protocol analysis with bounded number of sessions. In this setting a natural, non-restrictive assumption is usually considered: the constraint system must be in some special form, more precisely, such constraint systems must be *well-formed*. This property holds every time one builds a constraint system to check an insecurity of a given protocol session in the presence of an active Dolev-Yao intruder.

A lot of works were done on relaxing the perfect cryptography assumption and considering algebraic properties of operations used to build messages, for example, XOR, etc.

But these techniques work only when considering a classical Dolev-Yao intruder model. If we switch to another model where multiple non-communicating intruders are attacking a protocol execution, then the system of deducibility constraints that is build for modeling this case is not anymore well-formed.

This motivated us to consider general constraints, where "well-formedness" is relaxed. In Part I we assumed such constraint systems and showed that satisfiability in this setting is decidable and the problem is NP-complete.

Summing up

Satisfiability of general deducibility constraint systems is NP-complete problem for DY and DY+ACI.

Moreover, we also considered an ACI symbol (that is, associative, commutative and idempotent) as a constructor for sets. These properties come to the rescue if one wants to model a set of XML nodes. It may be extremely useful for the protocols whose messages are based on XML language. For example, the majority of Web Services use SOAP protocol which rely on XML. Thus, the extension can be employed for reasoning about security of Web Services, where each Web Service is seen as an agent playing some protocol role.

We presented an example of how our technique can be used to model an attack based on XML-representation of messages and its processing. It is not a novelty [CLR07], but proposes another way to deal with this kind of attacks. Meanwhile, our technique admits to consider XML-rewriting attacks in a novel setting (multiple intruders), that may be important for checking the security of Web Services communication with untrusted peers.

Summing up

The technique allows one to model (and search for) XML-rewriting attacks.

As we mentioned above, general constraint systems may be used to reason about security of protocol sessions in presence of multiple active DY intruders, each of them controls only some communication channels but not the entire network. Moreover, they cannot communicate during the execution of the protocol, but only when they quit it. In this setting we showed that the problem of secrecy is decidable for a given protocol session. We called it a *coordinated attack problem*.

The other domain where we applied the technique of satisfying general deducibility constraint systems is a composition of Web Services under constraint of security policies. We considered a distributed orchestration model that covers both choreography and orchestration approaches in Web Services composition.

This model allows an automatic composition technique we also described. As for our knowledge there is not much works on fully automatic composition of Web Services that take into account security policies of the available services.

Summing up

We presented a new model assuming multiple non-communicating intruders and gave within this model a decision procedure for checking insecurity of protocol sessions.

Moreover, our model restrict the communication between partners on which the orchestration is distributed in a way that no sensitive data of one partner leaks to another.

The automatic composition in these settings can be also reduced to the satisfiability of deducibility constraint systems but under *non-disclosure condition*. This condition as we presented it in the thesis does not affect the decidability. Thus, we showed that the problem of automatic distributed orchestration of Web Services under non-disclosure condition that takes into account security policies of the available Web Services is decidable.

Summing up

We showed that the automatic distributed orchestration of security-aware Web Services under non-disclosure policy is decidable.

A particular case of the presented model is an orchestration of security-aware Web Services. If we apply our approach to this case, there is no need of extended techniques for solving deducibility constraints. To decide satisfiability of well-formed ones is enough. Thus, we are able to reuse existing tools that are able to cope with this problem. One such tool we can find in cryptographic protocol analysis domain.

Summing up

We reported an implementation for the non-distributed case.

Basing on this tool we can build a software that can solve the problem of automatic orchestration of Web Services that takes into account security policies. We reported such an implementation called AVANTSSAR Orchestrator. We shortly discussed its input language and architecture and showed how it is used in the context of AVANTSSAR Validation Platform, that allows one to generate orchestration satisfying given security requirements. We also briefly reported the experimental results that were run for this tool.

Perspectives

There are several perspectives concerning satisfiability of general constraints. The presented technique is far from being perfect and all-embracing. First, one can see that the public keys considered in the signature can have only atomic values. While using atomic public keys we may cover quite a lot of useful cases, this unpleasant restriction can probably be removed.

Second, considering more symbols with particular algebraic properties, for example, even XOR or modular exponentiation, is a plus, since the applications of such extended technique may better approximate the real systems.

Third, extending general deducibility constraints with negation (allowing negative constraints) would give us, for example, some pretty procedures for Web Services composition under different constraints. Moreover, using this mixed constraint systems we may express fairness property for the protocols.

A related extension is allowing disequalities of terms as additional constraints. While it is weaker one than the previous for Dolev-Yao based theories, it allows one, for example, to express and check a violation of authentication property when reason about protocol insecurity within multiple non-communicating intruder model. For now we deal only with secrecy.

Finally, we lack an effective implementation of the proposed decision procedure. Without it, the presented approach may have only theoretical value. Thus, investigation of heuristic methods for satisfying general deducibility constraint systems, which is NP-complete, is also a way to go.

Concerning the applications of the decision procedure presented in this thesis, it is hard to define further extensions which is completely independent of the underlying technique discussed above. However, we may point out the necessity of thorough study of Web Services industrial standards and its correspondence to elements of our model.

On the other hand, if we don't stick to the deducibility constraints as the underlying layer, we may mention as a longer-term goal an extended behavior of Web Services allowing, for example, loops. The composition in these settings would possibly require a mixture of automata-based techniques which in general allow reasoning on looping behaviors, but usually do not take into account message structure, especially cryptographic primitives, and the presented approach, suffering from poor behavior expressiveness but which supports reasoning on message structure in the presence of cryptographic elements. Nevertheless, the completeness of such combination approaches might be unachievable.

Bibliography

[AARS08] Humberto Abdelnur, Tigran Avanesov, Michael Rusinowitch, and Radu State. Abusing sip authentication. In *IAS '08: Proceedings of the 2008 The Fourth International Conference on Information Assurance and Security*, pages 237–242, Washington, DC, USA, 2008. IEEE Computer Society.

[AARS09] Humberto Abdelnur, Tigran Avanesov, Michael Rusinowitch, and Radu State. Abusing SIP authentication. *Journal of Information Assurance and Security*, 4:311–318, 2009.

[ABCC11] Wihem Arsac, Giampaolo Bella, Xavier Chantry, and Luca Compagna. Multi-attacker protocol validation. *Journal of Automated Reasoning*, 46:353–388, 2011. 10.1007/s10817-010-9185-y.

[AC08] Alessandro Armando and Luca Compagna. SAT-based Model-Checking for Security Protocols Analysis. *International Journal of Information Security*, 7(1):3–32, 2008.

[ACC07] Alessandro Armando, Roberto Carbone, and Luca Compagna. LTL Model Checking for Security Protocols. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF20)*. IEEE Computer Society Press, 2007.

[ACC+08] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, and Llanos Tobarra Abad. Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps. In *Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*. ACM Press, 2008.

[ACRT10] Tigran Avanesov, Yannick Chevalier, Michael Rusinowitch, and Mathieu Turuani. Satisfiability of General Intruder Constraints with and without a Set Constructor. Research Report RR-7276, INRIA, 05 2010.

[ACRT11] Tigran Avanesov, Yannick Chevalier, Michaël Rusinowitch, and Mathieu Turuani. Satisfiability of general intruder constraints with and without a set constructor. *CoRR*, abs/1103.0220, 2011.

[ACRTar] Tigran Avanesov, Yannick Chevalier, Michael Rusinowitch, and Mathieu Turuani. Satisfiability of General Intruder Constraints with a Set Constructor. In *The Fifth International Conference on Risks and Security of Internet and Systems (CRiSIS2010)*. IEEExplore, To appear.

[AT09] Charu Arora and Mathieu Turuani. Validating integrity for the ephemerizer's protocol with cl-atse. In *Formal to Practical Security: Papers Issued from the 2005-*

2008 French-Japanese Collaboration, pages 21–32, Berlin, Heidelberg, 2009. Springer-Verlag.

[avaa] Automated Validation of Trust and Security of Service-Oriented Architectures, AVANTSSAR project. `http://www.avantssar.eu`.

[avab] AVANTSSAR orchestration and validation platform on Loria-INRIA CASSIS team page. `http://cassis.loria.fr/wiki/Wiki.jsp?page=Avantssar`.

[AVA08] AVANTSSAR. Deliverable 2.1: Requirements for modelling and ASLan v.1. Available at `http://www.avantssar.eu`, 2008.

[AVA10] AVANTSSAR. Deliverable 4.2: AVANTSSAR Validation Platform v.2. `www.avantssar.eu`, 2010.

[AVA11] AVANTSSAR. Deliverable 2.3 (update): ASLan++ specification and tutorial. Available at `http://www.avantssar.eu`, 2011.

[AVI03a] AVISPA. Deliverable 2.1: The High-Level Protocol Specification Language. `http://www.avispa-project.org`, 2003.

[AVI03b] AVISPA. Deliverable 2.3: The Intermediate Format. `http://www.avispa-project.org`, 2003.

[Bau05] Mathieu Baudet. Deciding security of protocols against off-line guessing attacks. In *ACM Conference on Computer and Communications Security*, pages 16–25, 2005.

[BCD+03] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic Composition of e-Services that export their Behavior. In *Proc. 1st Int. Conf. on Service Oriented Computing, ICSOC 2003*, volume 2910, 2003.

[BCD+05] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, and M. Mecella. Automatic Composition of Transition-based semantic Web Services with Messaging. In *Proc. 31st Int. Conf. Very Large Data Bases, VLDB 2005*, pages 613–624, 2005.

[BCD+09] K. Bhargavan, R. Corin, P.M. Deniélou, C. Fournet, and J.J. Leifer. Cryptographic protocol synthesis and verification for multiparty sessions. In *2009 22nd IEEE Computer Security Foundations Symposium*, page 124–140. IEEE, 2009.

[BCF08] Philippe Balbiani, Fahima Cheikh, and Guillaume Feuillade. Composition of interactive web services based on controller synthesis. *2nd International Workshop on Web Service Composition and Adaptation (WSCA 08)*, 0(0):521–528, 2008.

[BCGP08] Daniela Berardi, Fahima Cheikh, Giuseppe De Giacomo, and Fabio Patrizi. Automatic Service Composition via Simulation. *International Journal of Foundations of Computer Science*, 19(2):429–452, 2008.

[BCK96] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology—CRYPTO'96*, page 1–15. Springer, 1996.

[BCLD07] Sergiu Bursuc, Hubert Comon-Lundh, and Stéphanie Delaune. Associative-commutative deducibility constraints. In *Proceedings of the 24th Annual Symposium on Theoretical Aspects of Computer Science (STACS'07), volume 4393 of Lecture Notes in Computer Science*, pages 634–645. Springer, 2007.

[BDC09] Sergiu Bursuc, Stéphanie Delaune, and Hubert Comon-Lundh. Deducibility constraints. In Anupam Datta, editor, *Proceedings of the 13th Asian Computing Science Conference (ASIAN'09)*, volume 5913 of *Lecture Notes in Computer Science*, page 24–38, Seoul, Korea, December 2009. Springer.

[BFGO05] K. Bhargavan, C. Fournet, A.D. Gordon, and G. O'Shea. An advisor for web services security policies. In *Proceedings of the 2005 workshop on Secure web services*, page 1–9. ACM, 2005.

[BFGP04] K. Bhargavan, C. Fournet, A.D. Gordon, and R. Pucella. TulaFale: A security tool for web services. In *Formal Methods for Components and Objects*, page 197–222. Springer, 2004.

[BFHS03] Tevfik Bultan, Xiang Fu, Richard Hull, and Jianwen Su. Conversation specification: a new approach to design and analysis of e-service composition. In *WWW*, pages 403–410, 2003.

[BG06] Antonio Bucchiarone and Stefania Gnesi. A survey on services composition languages and models. In *in Proceedings of International Workshop on Web Services Modeling and Testing (WS-MaTe2006)*, pages 51–63, 2006.

[BHK05] Yohan Boichut, Pierre-Cyrille Héam, and Olga Kouchnarenko. Automatic Verification of Security Protocols Using Approximations. Rapport de recherche RR-5727, INRIA, 2005.

[BHM+07] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web Services Architecture. 2007. `http://www.w3.org/TR/ws-arch/`.

[BIPT09] Antonia Bertolino, Paola Inverardi, Patrizio Pelliccione, and Massimo Tivoli. Automatic synthesis of behavior protocols for composable web-services. In *ESEC/FSE '09: Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 141–150, New York, NY, USA, 2009. ACM.

[BKL01] D. Beringer, H. Kuno, and M. Lemon. Using WSCL in a UDDI Registry 1.02. UDDI Working Draft Technical Note Document, May 5, 2001. 2001.

[Bla01] B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *csfw01*, pages 82–96. ieeecoso, 2001.

[BLP03] Liana Bozga, Yassine Lakhnech, and Michaël Périn. Hermes: An automatic tool for verification of secrecy in security protocols. In Rebecca Wright, editor, *Financial Cryptography*, volume 2742 of *Lecture Notes in Computer Science*, pages 219–222. Springer Berlin / Heidelberg, 2003.

[BMM06] Luciano Baresi, Andrea Maurino, and Stefano Modafferi. Towards distributed bpel orchestrations. *ECEASST*, 3, 2006.

[BMar] Achim Brucker and Sebastian Mödersheim. Integrating automated and interactive protocol verification. In *Proceedings of FAST 2009*, Lecture Notes in Computer Science. Springer-Verlag, 2009 (To appear). Extended version available as IBM Research Report RZ3750.

[BMV05a] David Basin, Sebastian Mödersheim, and Luca Viganò. Algebraic intruder deductions. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, pages 549–564, 2005.

[BMV05b] David Basin, Sebastian Mödersheim, and Luca Viganò. OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3):181–208, 2005.

[BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, New York, NY, USA, 1998.

[BO97] J. Bull and D.J. Otway. The authentication protocol. *Technical Report DRA/CIS3/PROJ/CORBA/SC/1/CSM/436-04/03, Defense Research Agency*, 1997.

[Bor01] Michele Boreale. Symbolic trace analysis of cryptographic protocols. In *Proceedings of the 28th International Conference on Automata, Language and Programming: ICALP'01*, LNCS 2076, pages 667–681. Springer-Verlag, Berlin, 2001.

[CDK+02] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. *Internet Computing, IEEE*, 6(2):86–93, 2002.

[CDL+99] I. Cervesato, N. A. Durgin, P.D. Lincoln, J. C. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *Computer Security Foundations Workshop, 1999. Proceedings of the 12th IEEE*, page 55–69. IEEE Computer Society Press, 1999.

[CDL06] Véronique Cortier, Stéphanie Delaune, and Pascal Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.

[CDM11] Hubert Comon-Lundh, Stéphanie Delaune, and Jonathan Millen. Constraint solving techniques and enriching the model with equational theories. In Véronique Cortier and Steve Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, volume 5 of *Cryptology and Information Security Series*, page 35–61. IOS Press, 2011. To appear.

[CGH+05] S. Creese, M. Goldsmith, R. Harrison, B. Roscoe, P. Whittaker, and I. Zakiuddin. Exploiting empirical engagement in authentication protocol design. *Security in Pervasive Computing*, page 119–133, 2005.

[CGL+08] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Massimo Mecella, and Fabio Patrizi. Automatic service composition and synthesis: the roman model. *IEEE Data Eng. Bull.*, 31(3):18–22, 2008.

[CHY07] Marco Carbone, Kohei Honda, and Nobuko Yoshida. Structured communication-centred programming for web services. In Rocco De Nicola, editor, *Programming Languages and Systems*, volume 4421 of *Lecture Notes in Computer Science*, page 2–17. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-71316-6_2.

[CK07] Y. Chevalier and M. Kourjieh. Key substitution in the symbolic analysis of cryptographic protocols. *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science*, page 121–132, 2007.

[CKR+03] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, and L. Vigneron. Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents. In *Proceedings of FSTTCS'2003*, Lecture Notes in Computer Science. Springer-Verlag, 2003.

[CKRT03a] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *Proceedings of the Logic In Computer Science Conference, LICS'03*, pages 261–270, 2003.

[CKRT03b] Y. Chevalier, R. Küsters, M. Rusinowitch, and M Turuani. Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents. In *Proceedings of the Foundations of Software Technology and Theoretical Computer Science, FST TCS'03*, LNCS 2914. Springer-Verlag, 2003. `http://www.avispa-project.org`.

[CKRT04] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the Security of Protocols with Commuting Public Key Encryption. In *Workshop on Automated Reasoning for Security Protocol Analysis - ARSPA'2004*, Electronic Notes in Theoretical Computer Science - ENTCS, Cork, Ireland, Jul 2004.

[CKRT05] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. *Theoretical Computer Science*, 338(1-3):247–274, June 2005.

[CLC03] H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. Technical Report LSV-03-3, Laboratoire Specification and Verification, ENS de Cachan, Cachan, France, January 2003.

[CLR07] Yannick Chevalier, Denis Lugiez, and Michaël Rusinowitch. Towards an automatic analysis of web service security. In *FroCoS 2007, Liverpool, UK, September 10-12*, volume 4720 of *Lecture Notes in Computer Science*, pages 133–147. Springer, 2007.

[CLS03] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Logic in Computer Science, 2003. Proceedings. 18th Annual IEEE Symposium on*, pages 271–280. IEEE, IEEE Comp. Soc. Press, 2003.

[CLT04] H. Comon-Lundh and R. Treinen. Easy intruder deductions. *Verification: Theory and Practice*, page 182–184, 2004.

[CMR08] Y. Chevalier, M.A. Mekki, and M. Rusinowitch. Automatic composition of services with security policies. In *Proceedings of the 2008 IEEE Congress on Services - Part I*, SERVICES '08, page 529–537, Washington, DC, USA, 2008. IEEE.

[CMR09] Y. Chevalier, M.A. Mekki, and M. Rusinowitch. Orchestration under security constraints. In *Sixth International Workshop on Formal Aspects in Security and Trust (FAST2009) Eindhoven, the Netherlands, November 5-6, 2009*, 2009.

[CMS+09] J. Camara, J.A. Martin, G. Salaun, J. Cubo, M. Ouederni, C. Canal, and E. Pimentel. Itaca: An integrated toolbox for the automatic composition and adaptation of web services. *Software Engineering, International Conference on*, 0:627–630, 2009.

[Con07] The World Wide Web Consortium. Simple Object Access Protocol 1.2. `http://www.w3.org/TR/soap12-part1`, Apr 2007.

[CR10a] Y. Chevalier and M. Rusinowitch. Compiling and securing cryptographic protocols. *Information Processing Letters*, 110(3):116–122, 2010.

[CR10b] Yannick Chevalier and Michael Rusinowitch. Symbolic protocol analysis in the union of disjoint intruder theories: Combining decision procedures. *Theoretical Computer Science*, 411(10):1261 – 1282, 2010. ICALP 2005 - Track C: Security and Cryptography Foundations.

[Cre08] C.J.F. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, USA, Proc.*, volume 5123/2008 of *Lecture Notes in Computer Science*, pages 414–418. Springer, 2008.

[CV01] Yannick Chevalier and Laurent Vigneron. A tool for lazy verification of security protocols. In *ASE*, page 373–376. IEEE Computer Society, 2001.

[Del06a] S. Delaune. Easy intruder deduction problems with homomorphisms. *Information Processing Letters*, 97(6):213–218, 2006.

[Del06b] Stéphanie Delaune. An undecidability result for agh. *Theoretical Computer Science*, 368(1-2):161–167, 2006.

[DH76] W. Diffie and M. Helman. New directions in cryptography. *IEEE Transactions on Information Society*, 22(6):644–654, november 1976.

[DLHBH+02] G. Della-Libera, P. Hallam-Baker, M. Hondo, T. Janczuk, C. Kaler, H. Maruyama, N. Nagaratnam, A. Nash, R. Philpott, H. Prafullchandra, et al. Web services security policy language (WS-SecurityPolicy). 2002.

[DLLT06] Stéphanie Delaune, Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Symbolic protocol analysis in presence of a homomorphism operator and exclusive or. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, page 132–143. Springer, 2006.

[DLLT08] Stéphanie Delaune, Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Symbolic protocol analysis for monoidal equational theories. *Information and Computation*, 206(2-4):312–351, February-April 2008.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[DY83] D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.

[EG83] Shimon Even and Oded Goldreich. On the security of multi-party ping pong protocols. In *Proceedings of 24th IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 1983.

[ElG85] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31:469–472, 1985.

[EMM09] S. Escobar, C. Meadows, and J. Meseguer. Maude-npa: Cryptographic protocol analysis modulo equational properties. *Foundations of Security Analysis and Design V*, pages 1–50, 2009.

[FHG98] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why a security protocol is correct? In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, pages 160–171. IEEE Computer Society Press, New York, May 1998.

[FHG99] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7:191–230, 1999.

[FJHG99] F. Javier Thayer Fabrega, F. Javier, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct, 1999.

[Fou08] OWASP Foundation. OWASP-DV-008, OWASP Testing Guide, v3.0. `http://www.owasp.org/index.php/Testing_for_XML_Injection_(OWASP-DV-008)`, 2008.

[GFD09] Patrick Gallagher, Deputy Director Foreword, and Cita Furlani Director. Fips pub 186-3 federal information processing standards publication digital signature standard (dss), 2009.

[Gro06] Open Group. SOA Definition. `http://opengroup.org/projects/soa/doc.tpl?gdid=10632`, 2006.

[GS07] Prateek Gupta and Vitaly Shmatikov. Security analysis of voice-over-ip protocols. *Computer Security Foundations Symposium, IEEE*, 0:49 – 63, 2007.

[GTC+04] Joshua D. Guttman, F. Javier Thayer, Jay A. Carlson, Jonathan C. Herzog, John D. Ramsdell, and Brian T. Sniffen. Trust management in strand spaces: A rely-guarantee method. In *In Proc. of the European Symposium on Programming (ESOP '04), LNCS*, page 325–339. Springer-Verlag, 2004.

[Guo08] LI Guoqiang. *On-the-fly Model Checking of Security Protocols*. PhD thesis, Japan Advanced Institute of Science and Technology, 2008.

[Haa02] Hugo Haas. Web service usage scenario: Travel reservation. `http://www.w3.org/2002/04/17-ws-usecase`, 2002.

[IIS10] P. Inverardi, V. Issarny, and R. Spalazzese. A theory of mediators for eternal connectors. *Leveraging Applications of Formal Methods, Verification, and Validation*, page 236–250, 2010.

[JK03] J. Jonsson and B. Kaliski. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447, Internet Engineering Task Force, February 2003.

[Kel76] R.M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, 1976.

[KNW03] Deepak Kapur, Paliath Narendran, and Lida Wang. An e-unification algorithm for analyzing protocols that use modular exponentiation. In *Proceedings of the 14th international conference on Rewriting techniques and applications*, RTA'03, pages 165–179, Berlin, Heidelberg, 2003. Springer-Verlag.

[Kø05] G.M. Køien. Privacy enhanced cellular access security. In *Proceedings of the 4th ACM workshop on Wireless security*, page 57–66. ACM, 2005.

[LLT05] Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Intruder deduction for c-like equational theories with homomorphisms. In Jürgen Giesl, editor, *RTA*, LNCS 3467, pages 308–322. Springer, 2005.

[LM96] Steven Low and Nicholas Maxemchuk. Modeling cryptographic protocols and their collusion analysis. In Ross Anderson, editor, *Information Hiding*, volume 1174 of *Lecture Notes in Computer Science*, page 169–184. Springer Berlin / Heidelberg, 1996.

[Low95] Gavin Lowe. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Information Processing Letters*, 56(3):131–136, november 1995.

[LW10] Niels Lohmann and Karsten Wolf. Artifact-centric choreographies. In Paul Maglio, Mathias Weske, Jian Yang, and Marcelo Fantinato, editors, *8th International Conference on Service Oriented Computing, ICSOC 2010, December 7-10, 2010, San Francisco, California, USA, Proceedings*, volume 6470 of *Lecture Notes in Computer Science*, page 32–46. Springer-Verlag, December 2010.

[Maz05] Laurent Mazaré. Satisfiability of Dolev-Yao Constraints. *Electronic Notes in Theoretical Computer Science*, 125(1):109–124, 2005.

[Maz06] Laurent Mazaré. *Computational Soundness of Symbolic Models for Cryptographic Protocols*. PhD thesis, Institut National Polytechnique de Grenoble, October 2006.

[Men02] D.A. Menasce. QoS issues in Web services. *Internet Computing, IEEE*, 6(6):72–75, 2002.

[MFP11] J.A. Martín, Martinelli. F., and E. Pimentel. Synthesis of secure adaptors. In *to appear in JLAP*, 2011.

[MK08] Jay A. McCarthy and Shriram Krishnamurthi. Cryptographic protocol explication and end-point projection. In *Proceedings of ESORICS'08*, LNCS 5283, page 533–547. Springer, 2008.

[MS01] Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the ACM Conference on Computer and Communications Security CCS'01*, pages 166–175, 2001.

[MS03] Jonathan K. Millen and Vitaly Shmatikov. Symbolic protocol analysis with products and diffie-hellman exponentiation. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop (CSFW'03)*, pages 47–61. IEEE Computer Society Press, 2003.

[MS05] J. Millen and V. Shmatikov. Symbolic protocol analysis with an abelian group operator or Diffie-Hellman exponentiation. *Journal of Computer Security*, 13(3):515–564, 2005.

[MTH90] Robin Milner, Mads Tofte, and Robert Harper. *The Definition of Standard ML*. MIT Press, Cambridge, MA, USA, 1990.

[MV09] Sebastian Mödersheim and Luca Viganò. The Open-source Fixed-point Model Checker for Symbolic Analysis of Security Protocols. In *Fosad 2007-2008-2009*, LNCS 5705, pages 166–194. Springer-Verlag, 2009.

[MVBar] Sebastian Mödersheim, Luca Viganò, and David Basin. Constraint Differentiation: Search-Space Reduction for the Constraint-Based Analysis of Security Protocols. *Journal of Computer Security*, 2009 (to appear).

[NH87] R. De Nicola and M. Hennessy. CCS without τ's. In *TAPSOFT'87*, page 138–152. Springer, 1987.

[NH06] S. Nanz and C. Hankin. A framework for security analysis of mobile wireless networks. *Theoretical Computer Science*, 367(1-2):203–227, 2006.

[NS78] Roger Needham and Michael Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), december 1978.

[OASa] OASIS. ebXML standards. `http://www.ebxml.org/`.

[OASb] OASIS. OASIS SOA Reference Model TC. `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm`.

[OAS05] OASIS. Security Assertion Markup Language (SAML) v2.0. Available at `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security`, April 2005.

[Oas06] Oasis Consortium. Web Services Business Process Execution Language Version 2.0. `http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsbpel`, 23 January, 2006.

[Oas07] Oasis Technical Comittee on Secure Exchange. Ws-securitypolicy 1.2. `http://doc.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-cd-02.pdf`, 2007.

[Ohl02] Enno Ohlebusch. *Advanced Topics in Term Rewriting*. Springer, New York, 2002.

[Pad08] Luca Padovani. Contract-Directed Synthesis of Simple Orchestrators. In *Proceedings of the 19th International Conference on Concurrency Theory (CONCUR'08)*, volume LNCS 5201, page 131–146. Springer, 2008.

[Pat09] Fabio Patrizi. An introduction to simulation-based techniques for automated service composition. In *Proceedings Fourth European Young Researchers Workshop on Service Oriented Computing, (YR-SOC 2009), Pisa, Italy, 17-19th June 2009*, volume 2 of *EPTCS*, pages 37–49, 2009.

[Pau97] Lawrence C. Paulson. Mechanized proofs for a recursive authentication protocol. In *10th Computer Security Foundations Workshop*, pages 84–95. IEEE Computer Society Press, 1997. `http://citeseer.nj.nec.com/article/paulson97mechanized.html`.

[PE09] Gabriel Pedraza and Jacky Estublier. Distributed orchestration versus choreography: The focas approach. In *ICSP '09: Proceedings of the International Conference on Software Process*, pages 75–86, Berlin, Heidelberg, 2009. Springer-Verlag.

[Pel03a] Chris Peltz. Web Services Orchestration, 2003. HP white paper.

[Pel03b] Chris Peltz. Web services orchestration and choreography. *Computer*, 36:46–52, 2003.

[PQ01] Olivier Pereira and Jean-Jacques Quisquater. Security analysis of the Cliques protocols suites. In *Proceedings of CSFW'01*. IEEE Computer Society Press, 2001.

[QBHG09] S. Quinton, I. Ben-Hafaiedh, and S. Graf. From orchestration to choreography: Memoryless and distributed orchestrators. In *Proc. of FLACOS'09*, 2009.

[rH06] D. Eastlake 3rd and T. Hansen. US Secure Hash Algorithms (SHA and HMAC-SHA). RFC 4634, Internet Engineering Task Force, July 2006.

[Riv92] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, Internet Engineering Task Force, April 1992.

[RS98] P. Ryan and S. Schneider. An attack on a recursive authentication protocol: A cautionary tale. *Information Processing Letters*, 65(1):7–10, 1998.

[RSA78] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[RSC+02] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, Internet Engineering Task Force, June 2002.

[RT01] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proceedings of CSFW'01*, pages 174–190. IEEE Computer Society Press, 2001.

[RT03] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with a finite number of sessions, composed keys is np-complete. *Theor. Comput. Sci.*, 1-3(299):451–475, 2003.

[Sch94] B. Schneier. Description of a new variable-length key, 64-bit block cipher (Blowfish). In *Fast Software Encryption*, page 191–204. Springer, 1994.

[Shm04] Vitali Shmatikov. Decidable analysis of cryptographic protocols with products and modular exponentiation. In *Proceedings of the 13th European Symposium on Programming (ESOP '04)*, LNCS 2986, pages 355–369. Springer-Verlag, 2004.

[SK03] B. Srivastava and J. Koehler. Web service composition-current solutions and open problems. In *ICAPS 2003 Workshop on Planning for Web Services*, volume 35. Citeseer, 2003.

[SMC00] Paul Syverson, Catherine Meadows, and Iliano Cervesato. Dolev-Yao is no better than Machiavelli. In *First Workshop on Issues in the Theory of Security — WITS'00*, pages 87–92, 2000.

[TBG07] Ter, Antonio Bucchiarone, and Stefania Gnesi. Web service composition approaches: From industrial standards to formal methods. In *Internet and Web Applications and Services, 2007. ICIW '07. Second International Conference on*, page 15, 2007.

[TPC+ro] Michele Trainotti, Marco Pistore, Gaetano Calabrese, Gabriele Zacco, Gigi Lucchese, Fabio Barbon, Piergiorgio Bertoli, and Paolo Traverso. ASTRO: Supporting Composition and Execution of Web Services. In *Service-Oriented Computing - ICSOC 2005: Third International Conference*, LNCS 3826, pages 495–501. Springer Verlag, 2005. URL of the ASTRO project: `http://sra.itc.it/projects/astro/`.

[Tur03] M. Turuani. *Sécurité des Protocoles Cryptographiques: Décidabilité et Complexité*. Thèse de doctorat, Université Henri Poincaré, Nancy, décembre 2003.

[Tur06] Mathieu Turuani. The CL-Atse Protocol Analyser. In *Term Rewriting and Applications (RTA)*, LNCS 4098, pages 277–286, 2006.

[UDD04] UDDI. Introduction to UDDI: Important Features and Functional Concepts. `http://uddi.org/pubs/uddi-tech-wp.pdf`, 2004.

[vRTC+04] SAP Claus von Riegen, I. Trickovic, L. Clément, A. Hately, and T. Bellwood. Using BPEL4WS in a UDDI registry. `http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-bpel-20040725.pdf`, 2004.

[W3C08] W3C. Extensible Markup Language (XML) 1.0. `http://www.w3.org/TR/REC-xml/`, 2008.

[Wir97] LAN Wireless. medium access control (MAC) and physical layer (PHY) specifications. *IEEE std*, 802(1), 1997.

[Wor01] World Wide Web Consortium. Web Services Description Language (WSDL) 1.1. `http://www.w3.org/TR/wsdl`, 15 March, 2001.

[Wor07] World Wide Web Consortium. XML Path Language (XPath) 2.0. `http://www.w3.org/TR/xpath20/`, 23 January, 2007.

Résumé

Les contraintes de déductibilité ont été introduites pour vérifier les protocoles cryptographiques. Cette thèse présente une procédure de décision pour le problème de satisfaisabilité des systèmes généraux de contraintes de déductibilité. Deux cas sont envisagés: la théorie de Dolev-Yao standard et son extension par un opérateur associatif, commutatif, idempotent. Le résultat est utilisé pour résoudre le problème de l'orchestration automatique et distribué de services Web sécurisés. Comme seconde application nous proposons une procédure pour décider la sécurité d'un nombre fini de sessions de protocole en présence de plusieurs intrus qui ne communiquent pas. Nous montrons également comment il est possible de détecter certaines attaques par réécriture qui exploitent le format des messages en XML.

Mots-clés: contraintes de déductibilité, ACI, composition de services Web, orchestration distribuée, protocoles

Abstract

Deducibility constraints have been introduced to verify cryptographic protocols. This thesis gives a decision procedure for the satisfiability problem of general deducibility constraints. Two cases are considered: the standard Dolev-Yao theory and its extension with an associative, commutative idempotent operator. The result is applied to solve the automated distributed orchestration problem for secured Web services. As a second application we give a procedure to decide the security of a cryptographic protocol in presence of several non-communicating intruders. We also show how it is possible to detect some XML rewriting attacks on Web services.

Keywords: Deducibility constraints, ACI, Web Services composition, Distributed orchestration, Protocols