VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

préparée dans le cadre d'une cotutelle entre l'*Université de Grenoble* et *Vysoké učení technické v Brně*

Spécialité : **Mathematiques, Sciences et Technologie de l'Information, Informatique**

Arrêté ministériel : le 6 janvier 2005 - 7 août 2006

Présentée par

« Filip KONEČNÝ »

Thèse dirigée par **« Tomáš VOJNAR »** et **« Yassine LAKHNECH »** codirigée par **« Radu IOSIF »**

préparée au sein de **Laboratoire VERIMAG**

dans **l'École Doctorale Mathematiques, Sciences et Technologie de l'Information, Informatique**

Vérification relationnelle pour des programmes avec des données entières

Thèse soutenue publiquement le **« 29/10/2012 »**,
devant le jury composé de :

professeur Ahmed BOUAJJANI
Rapporteur
professeur Petr JANČAR
Rapporteur
professeur Parosh ABDULLA
Membre
professeur Mojmír KŘETÍNSKÝ
Membre
professeur agrégé David MONNIAUX
Membre
professeur Milan ČESKA
Président

Université Joseph Fourier / Université Pierre Mendès France / Université Stendhal / Université de Savoie / Grenoble INP

Relational Verification of Programs with Integer Data

Filip Konečný

October 29, 2012

Abstract

This work presents novel methods for verification of reachability and termination properties of programs that manipulate unbounded integer data. Most of these methods are based on acceleration techniques which compute transitive closures of program loops.

We first present an algorithm that accelerates several classes of integer relations and show that the new method performs up to four orders of magnitude better than the previous ones. On the theoretical side, our framework provides a common solution to the acceleration problem by proving that the considered classes of relations are periodic.

Subsequently, we introduce a semi-algorithmic reachability analysis technique that tracks relations between variables of integer programs and applies the proposed acceleration algorithm to compute summaries of procedures in a modular way. Next, we present an alternative approach to reachability analysis that integrates predicate abstraction with our acceleration techniques to increase the likelihood of convergence of the algorithm. We evaluate these algorithms and show that they can handle a number of complex integer programs where previous approaches failed.

Finally, we study the termination problem for several classes of program loops and show that it is decidable. Moreover, for some of these classes, we design a polynomial time algorithm that computes the exact set of program configurations from which non-terminating runs exist. We further integrate this algorithm into a semi-algorithmic method that analyzes termination of integer programs, and show that the resulting technique can verify termination properties of several non-trivial integer programs.

Keywords

programs with integers, counter automata, reachability analysis, termination analysis, acceleration, transitive closures, procedure summaries, recurrent sets, termination preconditions

Abstrakt

Tato práce představuje nové metody pro verifikaci programů pracujících s neomezenými celočíslenými proměnnými, konkrétně metody pro analýzu dosažitelnosti a konečnosti. Většina těchto metod je založena na akceleračních technikách, které počítají tranzitivní uzávěry cyklů programu.

V práci je nejprve představen algoritmus pro akceleraci několika tříd celočíselných relací. Tento algoritmus je až o čtyři řády rychlejší než existující techniky. Z teoretického hlediska práce dokazuje, že uvažované třídy relací jsou periodické a poskytuje tudíž jednotné řešení prolému akcelerace.

Práce dále představuje semi-algoritmus pro analýzu dosažitelnosti celočíselných programů, který sleduje relace mezi proměnnými programu a aplikuje akcelerační techniky za účelem modulárního výpočtu souhrnů procedur. Dále je v práci navržen alternativní algoritmus pro analýzu dosažitelnosti, který integruje predikátovou abstrakci s accelerací s cílem zvýšit pravděpodobnost konvergence výpočtu. Provedené experimenty ukazují, že oba algoritmy lze úspěšně aplikovat k verifikaci programů, na kterých předchozí metody selhávaly.

Práce se rovněž zabývá problémem konečnosti běhu programů a dokazuje, že tento problém je rozhodnutelný pro několik tříd celočíselných relací. Pro některé z těchto tříd relací je v práci navržen algoritmus, který v polynomiálním čase vypočítá množinu všech konfigurací programu, z nichž existuje nekonečný běh. Tento algoritmus je integrován do metody, která analyzuje konečnost běhů celočíselných programů. Efektivnost této metody je demonstrována na několika netriviálních celočíselných programech.

Klíčová slova

programy s celočíselnými daty, čítačové automaty, analýza dosažitelnosti, analýza konečnosti, akcelerace, tranzitivní uzávěry, souhrny procedur, rekurentní množiny, vstupní podmínky pro konečnost

Résumé

Les travaux présentés dans cette thèse sont lies aux problèmes de vérification de l'atteignabilité et de la terminaison de programmes qui manipulent des données entières non-bornées. On décrit une nouvelle méthode de vérification basée sur une technique d'accélération de boucle, qui calcule, de manière exacte, la clôture transitive d'une relation arithmétique.

D'abord, on introduit un algorithme d'accélération de boucle qui peut calculer, en quelques secondes, des clôtures transitives pour des relations de l'ordre d'une centaine de variables. Ensuite, on présente une méthode d'analyse de l'atteignabilité, qui manipule des relations entre les variables entières d'un programme, et applique l'accélération pour le calcul des relations entrée-sortie des procédures, de façon modulaire.

Une approche alternative pour l'analyse de l'atteignabilité, présentée également dans cette thèse, intègre l'accélération avec l'abstraction par prédicats, afin de traiter le problème de divergence de cette dernière. Ces deux méthodes ont été évaluées de manière pratique, sur un nombre important d'exemples, qui étaient, jusqu'a présent, hors de la portée des outils d'analyse existants.

Dernièrement, on a étudié le problème de la terminaison pour certaines classes de boucles de programme, et on a montré la décidabilité pour les relations étudiées. Pour ces classes de relations arithmétiques, on présente un algorithme qui s'exécute en temps au plus polynomial, et qui calcule l'ensemble d'états qui peuvent générer une exécution infinie. Ensuite on a intégré cet algorithme dans une méthode d'analyse de la terminaison pour des programmes qui manipulent des données entières.

Mots clés

programmes entiers, automates a compteurs, analyse d'atteignabilite, analyse de terminaison, accélération, clôture transitive, sommaire de procédure, ensembles récurrents, preconditions de terminaison

Acknowledgements

I would like to thank everyone who contributed to the successful completion of this work. First and foremost, I would like to thank my supervisors Tomáš Vojnar and Radu Iosif for their guidance, cooperation, invaluable suggestions and support they offered during the research. Next, I would also like to thank my research partners Marius Bozga, Peter Habermehl, Hossein Hojjat, and Viktor Kuncak for great cooperation on our joint publications. Finally, I would like to thank my family for their continual moral support.

The work presented in this thesis was supported by the Czech Science Foundation (projects P103/10/0306, 102/09/H042, 201/09/P531), the Czech Ministry of Education (project COST OC10009, Czech-French Barrande project MEB021023, the long-term institutional project MSM0021630528), the European Science Foundation (ESF COST action IC0901), the EU/Czech IT4Innovations Centre of Excellence (project ED1.1.00/02.0070), the French National Research Agency (project ANR-09-SEGI-016 VERIDYC), and the Brno University of Technology (projects FIT-S-10-1, FIT-S-11-1, FIT-S-12-1).

Contents

1 Introduction

Formal verification is the task of analyzing *correctness* of a computer system with respect to a given *specification*. A majority of formal verification methods is *sound*, meaning that when a method finds no counterexample to the specification, then the inspected system is guaranteed to be correct. This is in contrast with traditional testing since the fact that no error manifests in a given set of test cases does not imply correctness of the system under inspection. Therefore, formal verification can be seen as a technique that is complementary to traditional testing. The experience of the computer systems industry in the last two decades shows that both techniques are needed to ensure reliability of computer systems. This need arises from increasing complexity and pervasiveness of computer systems. Indeed, more and more sophisticated software systems of growing sizes run on an increasingly complex hardware. Moreover, many computer systems are safety-critical since they operate aircraft, medical devices, nuclear systems, etc., and their malfunction may have serious consequences. Both hardware manufacturers and software developers aim to deliver reliable systems. Hence the growing need for more sophisticated formal verification methods that can cope with the increasing complexity of computer systems.

Model checking [CE82, QS82] is one of the main branches of formal verification. It is an approach of automated checking whether a *model* of a computer system (where the model can sometimes be identical to the system) satisfies a certain correctness specification by systematically exploring the state space of the system being verified. Model checking can be usually fully automated and moreover, in cases when a model violates a given specification, a model checking method can provide a counter-example which demonstrates the cause of the violation. Traditionally, model checking has been applied to systems with a *finite state space*, especially in hardware. A major success has been achieved when explicit representation of states has been replaced with a *symbolic* one [BCM$^+$92]. This is because a symbolic state represents a potentially large set of concrete states which can then be manipulated simultaneously. More recently, model checking has been extended to deal with *infinite* state spaces which arise, e.g., in software that manipulates dynamic data structures or integers. In certain cases, infinite state verification can be reduced to finite state verification by applying e.g. cut-off techniques [BHV03]. When such reduction is not possible, a model checking method needs to use a symbolic representation for possibly infinite sets of states that need to be explored. Many such representations have been proposed including those based on various kinds of *automata* and *logics* [BHV04, BHRV06]. Since an automaton recognizes a potentially infinite set of traces and a logical formula may have a potentially infinite set of models, they can be used to encode potentially infinite sets of states.

In our work, we address model checking of sequential non-recursive programs with *unbounded integer data* (also known as counter automata, counter systems, or counter machines). The interest for integer programs comes from the fact that they can encode various classes of systems with unbounded (or very large) data domains, such as hardware circuits, cache memories, or software systems with variables of non-primitive types, such as integer arrays, pointers and/or recursive data structures. This comes with no surprise since, in theory, any Turing-complete class of systems can be simulated by integer programs, as shown by Minsky [Min67]. For practical purposes, however, a number of recent works have revealed cost-effective reductions of verification problems for several classes of complex systems to decision problems phrased in terms of integer programs. Examples of such systems that can be effectively verified by means of integer programs, include: specifications of hardware components [SV07], programs with singly-linked lists [BBH+06, FLS07, BI07, BIP08], trees [HIRV07], and integer arrays [HIV08b, HIV08a, BHI+09]. Hence the growing interest for analysis tools working on integer programs.

Our work contributes to the field of model checking of programs with integer data by developing methods for reachability and termination analysis. Given an integer program, a set of initial states, and a set of error states, the *reachability problem* asks whether the program has a computation starting in a state from the initial set which leads to a state from the error set. The *termination problem* asks whether each computation of a program that starts in a state from the initial set eventually halts. A slightly more general problem is the *conditional termination problem* which asks to compute the set of initial program states from which every computation halts.

1.1 State of the Art

Since the result of Minsky [Min67], proving Turing-completeness of 2-counter machines with increment, decrement and zero test, research on analysis and verification of integer programs has been pursued in two orthogonal directions. The first one is defining subclasses of systems for which various decision problems, such as reachability or termination, are found to be decidable. Based on these results, model checking algorithms that perform precise analysis have been proposed. The second direction is concerned with finding sound (but not necessarily complete) answers to decision problems in a cost-effective way, by abstracting the system under inspection.

Decidable Classes of Integer Programs

Examples of classes of integer programs where the reachability problem is decidable include *reversal-bounded counter machines* [Iba78], *Petri nets* and *vector addition systems* [Reu90], or *flat counter automata* [CJ98, LS05]. These classes pose certain restrictions on the syntax of transition labels (vector addition systems, flat counter automata), control structure (flat counter automata), or semantics (reversal-bounded counter machines) of integer programs. Often, decidability of various problems is proved by defining the set of reachable states in a decidable logic, such as Presburger arithmetic [Pre29].

The termination problem was studied for Petri nets where it was shown undecidable when strong fairness is considered [Car87] and decidable for weak fairness assumptions [Jan90]. A direct consequence of a result on the liveness problem from [DIP01] is that the termination problem is decidable for reversal-bounded counter automata. Concerning flat counter automata, the decidability status of the termination problem remains open.

Precise Model Checking Methods

A closely related line of work consists in attempts to apply model checking techniques to the verification of integer programs. To solve the reachability problem, one typically proceeds by computing the set of states that are reachable from the initial set I via the transition relation R of the inspected system, and checking the emptiness of the intersection with the set of error states. The set of reachable states can be defined as the post-image of the initial set I via the transitive closure R^* of the transition relation R, formally as $R^*(I)$. The computation of $R^*(I)$ is usually done by computing successive under-approximations of the set of reachable states. Consider a naive technique that computes the set of reachable states S by first initializing S_0 with I ($S_0 \leftarrow I$) and then iteratively computing $S_{i+1} \leftarrow S_i \cup R(S_i)$, where $R(S_i)$ denotes the post-image of S_i via the transition relation R. The computation terminates if $S_k = S_{k+1}$ for some $k \geq 0$. Termination is guaranteed only if the set of reachable states is finite. Therefore, early methods for verification of infinite state systems like FIFO-channel systems [BH99, BG99] attempted to increase the likelihood of convergence of the computation by applying *acceleration* techniques. Essentially, given a loop L of a system and a reachability set S_i computed so far, an acceleration technique aims to compute effects of executing the loop L arbitrarily many times, formally, to compute $S_{i+1} \leftarrow S_i \cup L^+(S_i)$ where L^+ denotes the transitive closure of L. As a result, S_{i+1} may potentially contain infinitely more states than S_i.

To this end, it is important to know for which classes of arithmetic relations it is possible to compute the transitive closure precisely. To the best of our knowledge, the three main classes of integer relations for which transitive closures can be computed effectively and precisely are: (1) difference bounds constraints [CJ98, BIL09], (2) octagons [BGI09], and (3) finite monoid affine transformations [Boi99, FL02]. For these three classes, the transitive closures can be moreover defined in Presburger arithmetic.

The model checking methods from [Boi99, FL02] are based on computation of transitive closures of finite monoid affine relations. For certain restricted classes of systems, one can prove termination of the method from [FL02] as reported in [BFLS05]. These classes are typically equivalent, from a semantical point of view, to *flat* systems. A system is *flat* if it has no nested loops and, moreover, each loop in the system can be accelerated. Another model checking method from [AAB00] is based on an extrapolation heuristics that guesses an effect of iterating a loop infinitely many times and then checks whether this guess is correct.

The above methods lack modularity, which is one of the keys to scalability. Since larger programs are usually organized in many small functions, a modular verification approach aims at running an analysis per function (in isolation), computing a *function*

summary which is a relation between the input and output valuations of its parameters, and combining the results in a final verification condition.

On what concerns the existing acceleration algorithms for difference bounds and octagonal relations, they suffer from poor scalability in the number of variables. As reported in [BGI09], these methods can only handle difference bounds relations with no more than 10 variables and octagonal relations with no more than 5 variables.

Methods Based on Abstractions

Another, orthogonal, direction of work is concerned with finding sound (but not necessarily complete) answers to decision problems in a cost-effective way. Such approaches, based on the theory of *abstract interpretation* [CC77], use abstract domains (such as, e.g., polyhedra [CH78], octagons [Min06], etc.) and compute fixed points of the transfer functions, which are over-approximations of the sets of reachable states. A drawback of the methods based solely on abstract interpretation is the inability to deal with spurious counterexamples (false positives), i.e., errors caused by the use of a too coarse abstract domain. To ensure termination of state space exploration, abstract interpretation applies *widening*. In contrast to acceleration, the widening can be seen as an operator that computes the set of reachable states S_{i+1} as an extrapolation of S_i and $R(S_i)$ such that $S_{i+1} \supseteq S_i \cup R(S_i)$.

The method of *predicate abstraction* [GS97] combines ideas from abstract interpretation and model checking in order to compute program invariants in a goal-driven fashion. The underlying idea is to verify a program by reasoning about its *abstraction* that is easier to analyze, and is defined with respect to a set of predicates [GS97]. The set of predicates is refined, by adding new predicates, to achieve the precision needed to prove the absence or the presence of errors [BMMR01]. This technique is also known as Counter Example-based Abstraction Refinement (CEGAR) [CGJ$^+$03]. Typically, predicate abstraction computes an over-approximation of the transition system generated by a program and verifies whether an error state is reachable in the abstract system. If no error occurs in the abstract system, the algorithm reports that the original system is safe. Otherwise, if a path to an error state (counterexample) has been found in the abstract system, the corresponding concrete path is checked. If this latter path corresponds to a real execution of the system, then a real error has been found. Otherwise, the counterexample is spurious, the abstraction is refined in order to exclude the counterexample, and the procedure continues.

A key difficulty in this approach is to automatically find, during the refinement phase, predicates that make the abstraction sufficiently precise [BPR02]. A breakthrough technique is to generate predicates based on *Craig interpolants* [Cra57] derived from the proof of infeasibility of a spurious trace [HJMM04]. Adding the interpolant to the set of predicates prevents the spurious trace from reappearing in the refined abstract model. Cutting edge CEGAR tools, such as, e.g., ARMC [PR07], BLAST [HJMS03] or CPAchecker [BK11], are empirically successful on a variety of domains and are quite effective in finding bugs and certifying correctness of real-life systems (device drivers, web servers, subsystems of operating system kernels). However, abstraction refinement using

interpolants suffers from unpredictability of interpolants computed by provers, which can cause the verification process to diverge and never discover a sufficient set of predicates (even in cases such predicates exist). The failure of such a refinement approach manifests in a sequence of predicates that rule out longer and longer counterexamples, but still fails to discover more general inductive invariants that would rule out multiple spurious counterexamples during the refinement step.

1.2 Goals of the Thesis

The purpose of this thesis is to develop efficient methods for reachability and termination analysis of programs with integer data. This involves a design of more efficient algorithms for computing transitive closures of classes of relations that can be expressed in decidable logical fragments such as Presburger arithmetic. As discussed previously, the known classes satisfying this criterion are difference bounds, octagonal, and finite monoid affine relations. Another goal is to develop modular methods for reachability analysis of non-recursive integer programs that are based on acceleration algorithms. Further, we aim to resolve the decidability status of the (conditional) termination problem for difference bounds, octagonal, and finite monoid affine relations and to develop techniques for termination analysis of programs with integer data. Last but not least, we aim to find solutions to the divergence problem in predicate abstraction.

1.3 An Overview of the Contributions

This section outlines the main contributions of this thesis.

Precise Acceleration of Integer Relations

We consider the acceleration problem and inspect the classes of difference bounds, octagonal, and finite monoid affine relations, for which the transitive closure is known to be Presburger definable and effectively computable [CJ98, BGI09, Boi99]. We present a general theoretical framework for computing transitive closures of certain relations, called periodic. We define a notion of periodicity on classes of relations that can be naturally represented as matrices and show that the sequence of powers $\{R^k\}_{k=0}^{\infty}$ can be finitely represented for periodic relations. We study the three classes of arithmetic relations mentioned previously and show that they are periodic. On the theoretical side, this provides a unifying view and shorter proofs to the fact that the transitive closures for these classes are Presburger definable and that they can be effectively computed. We prove that our algorithm computing transitive closures of periodic relations runs in EXPTIME for all three classes. On the practical side, despite its asymptotic complexity, the algorithm computes the transitive closure of difference bounds and octagonal relations up to four orders of magnitude faster than the methods from [BIL09, BGI09] and also scales much better in the number of variables.

Techniques to Compute Weakest (Non-)termination Sets

We study the conditional termination problem, which is that of defining the set of initial states from which a given program terminates, for difference bounds, octagonal, and finite monoid affine relations. We define the dual set called *weakest non-termination set* of initial states from which a non-terminating execution exists, as the greatest fixpoint of the pre-image of the transition relation. Next, we show that this set can be defined as the limit of the Kleene sequence for the above classes of relations. This allows to effectively compute and represent the weakest non-termination set in Presburger arithmetic. Since Presburger arithmetic is decidable, the termination problem is thus decidable too, for all three classes. For difference bounds and octagonal relations, we next present a PTIME algorithm that computes the weakest non-termination set. Moreover, we investigate the existence of linear ranking functions and prove that for each well-founded difference bounds or octagonal relations, there exists an effectively computable witness relation, i.e., a well-founded relation which has equal weakest non-termination set and which has a linear ranking function. We also study the class of linear affine relations and give a method of under-approximating the termination precondition for a non-trivial subclass of affine relations, called polynomially bounded affine relations.

Reachability Analysis for Integer Programs

We give a semi-algorithmic method for reachability analysis of non-recursive integer programs which tracks relations instead of sets of reachable states, computes procedure summaries and is therefore modular. The algorithm builds the summary relation of a procedure incrementally, by eliminating control states and composing incoming with outgoing relations. The main difficulty here is the elimination of states with several self-loops. We address this issue by first computing the transitive closures of the self-loops individually, and then exploring all interleavings between them until no new relations are produced. Interleaving of transitive closures instead of self-loops themselves increases the likelihood of termination. We implemented these techniques in the FLATA tool which successfully verifies many non-trivial integer programs and outperforms other tools for model checking integer programs on many of the considered benchmarks.

Termination Analysis of Integer Programs

We present a semi-algorithmic method computing termination preconditions for non-recursive integer programs, which rests on computation of a *transition invariant* of a program and on the algorithm computing the weakest non-termination sets mentioned previously. Further, we show that this method is guaranteed to compute the weakest non-termination set for flat integer programs, which renders the termination problem decidable for this class of programs. We have implemented this technique and have verified (non-)termination of several benchmarks.

Coping with the Divergence Problem in Predicate Abstraction

We address the divergence problem that appears in predicate abstraction and observe that this problem can be alleviated by applying acceleration. We present Counterexample-Guided *Accelerated* Abstraction Refinement (CEGAAR), an interpolation-based predicate abstraction algorithm that applies acceleration algorithms for particular classes of loops to rule out an infinite family of counterexamples during the abstraction refinement phase. An essential ingredient of this approach are interpolants that not only rule out one spurious path, but are also *inductive* with respect to loops along this path and which hence rule out potentially infinitely many spurious paths. We observe that inductive interpolants can be computed from classical Craig interpolants and transitive closures of loops. We present an implementation of CEGAAR that verifies integer transition systems and show that the resulting implementation robustly handles a number of difficult transition systems that cannot be handled using interpolation-based predicate abstraction or acceleration alone.

1.4 Thesis Roadmap

Chapter 2 presents notation and basic definitions used in the rest of the thesis. Chapters 3, 4, and 5 present our results concerning the acceleration problem. Chapter 3 describes the general framework for computation of transitive closures of periodic relations which is then instantiated in Chapter 4 for difference bounds, octagonal, and finite monoid affine relations. The complexity of the transitive closure algorithm is studied in Chapter 5. Next, Chapter 6 presents a solution of the conditional termination problem for difference bounds, octagonal, and finite monoid affine relations. Chapter 7 builds upon the results of Chapters 3–6 and presents methods contributing to the analysis of programs with integer data: a modular reachability analysis algorithm based on procedure summaries, a conditional termination analysis method based on transition invariants, and finally, a method that integrates predicate abstraction and acceleration. All experiments that we performed are described in Chapter 8. Finally, Chapter 9 summarizes the obtained results, gives an overview of publications, and outlines possible future work.

2 Background

This chapter presents notions that will be used throughout this thesis. Section 2.1 gives basic notions on relations, formulae, and graphs. Next, Section 2.2 defines the syntax and the semantics of programs with integer data and then defines the reachability and conditional termination problems. Finally, Sections 2.3, 2.4, and 2.5 present basic definitions and several known results results on the three classes of relations we study in this thesis: difference bounds, octagonal, and finite monoid affine relations, respectively.

2.1 Basic Notions

Integer Sets

We denote by $\mathbb{Z}$, $\mathbb{N}$ and $\mathbb{N}_+$ the sets of integers, positive (including zero) and strictly positive integers, respectively. We denote by $\mathbb{Z}_\infty$ and $\mathbb{Z}_{-\infty}$ the sets $\mathbb{Z} \cup \{\infty\}$ and $\mathbb{Z} \cup \{-\infty\}$, respectively. In the rest of this paper we will fix the set of variables $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$, for some $N > 0$. The set of *primed* variables is $\mathbf{x}' = \{x'_1, x'_2, \ldots, x'_N\}$. These variables are assumed to be ranging over $\mathbb{Z}$, unless otherwise specified. For a set $S \in \mathbb{Z}$ of integers, we denote by $\max S$ the largest integer $m \in S$, if one exists and by $\min S$ the smallest integer $m \in S$, if one exists. By $\sup S$ we denote the smallest value $m \in \mathbb{Z}_\infty$ such that $s \leq m$, for all $s \in S$. By $\inf S$ we denote the largest value $m \in \mathbb{Z}_{-\infty}$ such that $s \geq m$, for all $s \in S$.

Presburger Arithmetic

A *linear term* t over a set of variables in $\mathbf{x}$ is a linear combination of the form $a_0 + \sum_{i=1}^{n} a_i x_i$, where $a_0, a_1, \ldots, a_n \in \mathbb{Z}$. An *atomic proposition* is a predicate of the form either $t \leq 0$, or $t \equiv_c 0$, where t is a linear term, $c \in \mathbb{N}_+$ is a strictly positive integer, and $\equiv_c$ is the equivalence relation modulo c. Given an integer $c \in \mathbb{N}_+$ and a linear term t, we write $c \mid t$ to denote the predicate $t \equiv_c 0$. *Presburger arithmetic* is the first-order logic over atomic propositions of the form either $t \leq 0$ or $t \equiv_c 0$. Presburger arithmetic has quantifier elimination and is decidable [Pre29]. For simplicity we consider only formulas in Presburger arithmetic in this thesis.

Arithmetic Formulae

Let φ be an arithmetic formula. By $AP(\varphi)$, we denote the set of atomic propositions in φ. By $FV(\varphi)$, we denote the set of free variables in φ, i.e. variables not bound by a quantifier. By writing $\varphi(\mathbf{x})$, we intend that $FV(\varphi) \subseteq \mathbf{x}$. We write $\bot$ and $\top$ for the boolean constants false and true. We use the symbols $\Rightarrow$ and $\Leftrightarrow$ to denote logical

implication and equivalence, respectively. By $\models \varphi$ we denote the fact that φ is valid, i.e. logically equivalent to $\top$. For a formula $\varphi(\mathbf{x})$, we denote by $\varphi[t_1/x_1, \ldots, t_N/x_N]$ (or equivalently, by $\varphi[t_i/x_i]_{i=1}^N$) the formula obtained from φ by syntactically replacing each free occurrence of x_i with the term t_i, $1 \leq i \leq N$. For a formula $\varphi(\mathbf{x})$ and a valuation $\nu : \mathbf{x} \to \mathbb{Z}$, we denote by $\nu \models \varphi(\mathbf{x})$ the fact that the formula obtained by replacing each free occurrence of $x \in \mathbf{x}$ with $\nu(x)$ is valid, formally $\models \varphi(\mathbf{x})[\nu(x)/x]_{x \in \mathbf{x}}$. A formula $\varphi(\mathbf{x})$ is said to be *consistent* if there exists a valuation $\nu : \mathbf{x} \to \mathbb{Z}$ such that $\nu \models \varphi(\mathbf{x})$. By $[\![\varphi]\!]$, we denote the set of all valuations satisfying $\varphi(\mathbf{x})$, formally $[\![\varphi]\!] = \{\nu : \mathbf{x} \to \mathbb{Z} \mid \nu \models \varphi(\mathbf{x})\}$.

Integer Relations

Let $R_1, R_2 \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$ be integer relations. Relational composition is defined as $R_1 \circ R_2 = \{(s, s') \in \mathbb{Z}^N \times \mathbb{Z}^N \mid \exists s'' \in \mathbb{Z}^N \ . \ (s, s'') \in R_1 \ \wedge \ (s'', s') \in R_2\}$. For any relation $R \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$, we consider R^0 to be the identity relation $\mathcal{I} = \{(s, s) \mid s \in \mathbb{Z}^N\}$ and we define $R^{i+1} = R^i \circ R$, for all $i \geq 0$. We say that R^i is the *i-th power* of R. With these notations, $R^+ = \bigcup_{i=1}^{\infty} R^i$ denotes the *transitive closure* of R, and $R^* = R^+ \cup \mathcal{I}$ denotes the *reflexive and transitive closure* of R. The inverse of R is defined as $R^{-1} = \{(s', s) \mid (s, s') \in R\}$. Further, we define $R^{-m} = (R^m)^{-1}$ for each $m \geq 1$. For any relation $R \in \mathbb{Z}^N \times \mathbb{Z}^N$ and set $S \subseteq \mathbb{Z}^N$, we define the post-image of S via R as $R(S) = \{s' \mid \exists s \in S \ \wedge \ (s, s') \in R\}$. Then, the pre-image of S via R is defined as $R^{-1}(S)$. The pre-image function of R, denoted as $\mathrm{pre}_R : \mathbb{Z}^N \to \mathbb{Z}^N$ maps each set $S \subseteq \mathbb{Z}^N$ to $R^{-1}(S)$. It is easy to show that $\mathrm{pre}_R^m = \mathrm{pre}_{R^m}$ for all $m \geq 0$ and that $\mathrm{pre}_R^m(\mathbb{Z}^N) = R^{-m}(\mathbb{Z}^N)$ for all $m \geq 0$. We say that a relation $R \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$ is *deterministic* if and only if $(s, s') \in R \wedge (s, s'') \in R \Rightarrow s' = s''$ for all $s, s', s'' \in \mathbb{Z}^N$.

An integer set $S \subseteq \mathbb{Z}^N$ is typically defined by an arithmetic formula $S(\mathbf{x})$. Similarly, an integer relation $R \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$ is typically defined by an arithmetic formula $R(\mathbf{x}, \mathbf{x}')$. Intuitively, $\mathbf{x}$ represent input variables and $\mathbf{x}'$ represents output variables. Note that for each $m \geq 0$, $\mathrm{pre}_R^m(\mathbb{Z}^N)$ can be defined by $\exists \mathbf{x}'.R^m(\mathbf{x}, \mathbf{x}')$. For a relation $R(\mathbf{x}, \mathbf{x}')$ and valuations $\nu, \nu' : \mathbf{x} \to \mathbb{Z}$, we denote by $\nu, \nu' \models R(\mathbf{x}, \mathbf{x}')$ the fact that the formula obtained by replacing each occurrence of $x \in \mathbf{x}$ with $\nu(x)$ and each occurrence of $x' \in \mathbf{x}'$ with $\nu'(x)$ is logically valid, formally $\models R(\mathbf{x}, \mathbf{x}')[\nu(x)/x, \nu'(x)/x']_{x \in \mathbf{x}}$.

A relation $R \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$ defined by $R(\mathbf{x}, \mathbf{x}')$ is *consistent* if and only if there exist valuations $\nu, \nu' : \mathbf{x} \to \mathbb{Z}$ such that $\nu, \nu' \models R(\mathbf{x}, \mathbf{x}')$. Clearly, $R(\mathbf{x}, \mathbf{x}')$ is consistent if and only if $R \neq \emptyset$. $R(\mathbf{x}, \mathbf{x}')$ is said to be $*$*-consistent* if and only if $R^m(\mathbf{x}, \mathbf{x}')$ is consistent for all $m \geq 0$. $R(\mathbf{x}, \mathbf{x}')$ is said to be *well-founded* if and only if there exists no infinite sequence of valuations $\{\nu_i : \mathbf{x} \to \mathbb{Z}\}_{i \geq 0}$ such that $(\nu_i, \nu_{i+1}) \models R(\mathbf{x}, \mathbf{x}')$ for all $i \geq 0$. When R represents the transition relation of a program, we sometimes say that valuations ν, ν' are program states. We sometimes use the same symbols to denote a relation and its defining formula or a set and its defining formula.

For a set of N-tuples $S \subseteq \mathbb{Z}^N$ and a relation $R \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$, let $post(S, R) \stackrel{def}{=} R(S)$ denote the *strongest postcondition* of S via R, and $wpre(S, R) = \{s \in \mathbb{Z}^N \mid \forall s' \ . \ (s, s') \in R \Rightarrow v \in S\}$ denote the *weakest precondition* of S with respect to R. We use *post* and *wpre* for sets and relations, as well as for logical formulae defining them.

If $\nu : \mathbf{x} \to \mathbb{Z}$ is a valuation, we write $\nu(\mathbf{x})$ to denote the N-tuple $(\nu(x_1), \ldots, \nu(x_N))$. Thus, if $\nu, \nu' : \mathbf{x} \to \mathbb{Z}$ are valuations and $R \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$ is a relation defined by $R(\mathbf{x}, \mathbf{x}')$, then $\nu, \nu' \models R(\mathbf{x}, \mathbf{x}')$ if and only if $(\nu(\mathbf{x}), \nu'(\mathbf{x})) \in R$. We denote by $\overrightarrow{\mathbf{y}} = (y_1, \ldots, y_k)$ a k-tuple of variables. We denote by $|\overrightarrow{\mathbf{y}}| = k$ the length of $\overrightarrow{\mathbf{y}}$. If $\mathbf{x}$ is a set of variables, we write $\overrightarrow{\mathbf{y}} \subseteq \mathbf{x}$ if all elements of $\overrightarrow{\mathbf{y}}$ are in $\mathbf{x}$. If $\overrightarrow{\mathbf{y}} = (y_1, \ldots, y_k)$, $\overrightarrow{\mathbf{y}} \subseteq \mathbf{x}$ is an ordered sequence of variables and $\nu : \mathbf{x} \to \mathbb{Z}$ is a valuation, we denote by $\nu(\overrightarrow{\mathbf{y}})$ the k-tuple of integers $(\nu(y_1), \ldots, \nu(y_k))$.

Weighted Digraphs

Let $G = \langle V, E, w \rangle$ be a *weighted digraph*, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, and $w : E \to \mathbb{Z}$ is a weight function. When G is clear from the context, we denote by $u \xrightarrow{n} v$ the fact that $(u, v) \in E$ and $w(\langle u, v \rangle) = n$. Let $\mu(G) = \max\{|n| \mid u \xrightarrow{n} v \text{ in } G\}$ denote the maximum absolute value of all weights in G.

A *path* in G is a sequence $\pi : v_0 \xrightarrow{w_1} v_1 \xrightarrow{w_2} v_2 \ldots v_{p-1} \xrightarrow{w_p} v_p$ where $v_{i-1} \xrightarrow{w_i} v_i$ is an edge in E with weight w_i, for each $1 \leq i \leq p$. For a path π of the above form, we denote the length of π by $|\pi| = p$, and the weight of π by $w(\pi) = \sum_{i=1}^{p} w_i$. The *average weight* of π is defined as $\overline{w}(\pi) = \frac{w(\pi)}{|\pi|}$. A path is *elementary* if all vertices, except for the first and the last one, are pairwise distinct. A *cycle* is a path where the first and the last vertex are equal. By $\pi_{i \ldots j}$, where $0 \leq i < j \leq p$, we denote the subpath $v_i \xrightarrow{w_{i+1}} v_{i+1} \xrightarrow{w_{i+2}} \ldots \xrightarrow{w_j} v_j$.

If $W \subseteq V$ is a set of vertices from G, the subgraph induced by W, denoted $G_{[W]}$, is defined by the restriction of E and w to the vertices in W. A subgraph $G_{[W]}$ is *strongly connected* if there exists a path between any two distinct vertices $u, v \in W$. $G_{[W]}$ is a *strongly connected component* if it is a maximal strongly connected subgraph of G.

2.2 Programs with Integer Data

2.2.1 Syntax

We define integer programs as collections of procedures. We abstract from specific programming language constructs and assume that each procedure is a control flow graph whose edges are labeled by Presburger arithmetic relations. In addition, certain edges correspond to calls between procedures, and the parameters and results are passed on by values. Formally, an *integer program* is a tuple $\mathcal{P} = \langle \mathbf{x}_g, \{P_1, \ldots, P_n\}, P_m \rangle$ where $\mathbf{x}_g$ are *global* variables, $P_1, \ldots, P_n$ are *procedures*, P_m is the *main* procedure, for some $m = 1, \ldots, n$, and each procedure is a tuple $P_i = \langle \mathbf{x}_i, \overrightarrow{\mathbf{x}}_i^{in}, \overrightarrow{\mathbf{x}}_i^{out}, Q_i, q_{0,i}, q_{f,i}, q_{e,i}, \Delta_i \rangle$, where:

- $\mathbf{x}_i$ are the *local* variables of P_i. We require that $\mathbf{x}_i \cap \mathbf{x}_g = \emptyset$ and that $\mathbf{x}_i \cap \mathbf{x}_j = \emptyset$ for all indices $i \neq j$, i.e. the global variables are the only variables shared by any two procedures.

- $\overrightarrow{\mathbf{x}}_i^{in} \subseteq \mathbf{x}_i$ and $\overrightarrow{\mathbf{x}}_i^{out} \subseteq \mathbf{x}_i$ are the *input* and *output* variables of P_i. Intuitively, input variables are used to pass the arguments, and output variables are used to retrieve the resulting values from a procedure.

- Q_i are the *control states* of P_i. We require that the sets of control states are pairwise disjoint, i.e. $Q_i \cap Q_j = \emptyset$, for all $i \neq j$.

- Δ_i is a set of *transition rules* of the form, either:

  1. $q \xrightarrow{R(\mathbf{x}_i \cup \mathbf{x}_g, \mathbf{x}'_i \cup \mathbf{x}'_g)} q'$ is an *internal transition*, where $q, q' \in Q_i$ are the source and destination state, and $R(\mathbf{x}_i \cup \mathbf{x}_g, \mathbf{x}'_i \cup \mathbf{x}'_g)$ is a Presburger arithmetic relation
  2. $q \xrightarrow{\overrightarrow{\mathbf{z}}' = call_j(\overrightarrow{\mathbf{t}})} q'$ is a *call transition*, where $q, q' \in Q_i$ are the source and destination control states, respectively, $j = 1 \dots n$ is the index of the callee procedure, $\overrightarrow{\mathbf{t}}$ is a sequence of linear terms over $\mathbf{x}_g \cup \mathbf{x}_i$, called *parameters*, and $\overrightarrow{\mathbf{z}} \subseteq \mathbf{x}_g \cup \mathbf{x}_i$ is a sequence of variables, called *return variables*. We require that $|\overrightarrow{\mathbf{t}}| = |\overrightarrow{\mathbf{x}}_j^{in}|$ and $|\overrightarrow{\mathbf{z}}| = |\overrightarrow{\mathbf{x}}_j^{out}|$, i.e. the numbers of parameters and return variables of the call transition match the numbers of input and output variables of the callee, respectively.

- $q_{0,i}, q_{f,i}, q_{e,i} \in Q_i$ are the *initial*, *final* and *error* control states of P_i. We require that these states are pairwise disjoint, that $q_{0,i}$ has no incoming transition rules, and that $q_{f,i}$ and $q_{e,i}$ have no outgoing transition rules.

For a program $\mathcal{P} = \langle P_1, \dots, P_n \rangle$ the *call graph* of $\mathcal{P}$, denoted $CG(\mathcal{P}) = \langle \mathcal{P}, \hookrightarrow \rangle$, is a graph whose vertices are procedures, and edges $P_i \hookrightarrow P_j$ represent calls of P_i to P_j. The program $\mathcal{P}$ is said to be *recursive* if and only if $CG(\mathcal{P})$ has cycles. In this thesis, we proceed under the assumption that the considered program is not recursive.

2.2.2 Semantics

A *configuration* of a procedure $P_i = \langle \mathbf{x}_i, \overrightarrow{\mathbf{x}}_i^{in}, \overrightarrow{\mathbf{x}}_i^{out}, Q_i, q_{0,i}, q_{f,i}, q_{e,i}, \Delta_i \rangle$ is a pair $\langle q, \nu \rangle$, where $q \in Q_i$ is a control state and $\nu : \mathbf{x}_i \cup \mathbf{x}_g \to \mathbb{Z}$ is a valuation of the variables visible in P_i. For each procedure P_i, we define the set of valuations of variables visible in P_i as $\mathcal{V}_i = \mathbb{Z}^{\mathbf{x}_i \cup \mathbf{x}_g}$. Next, we define predicates MatchCall and MatchCallRet which are later used to define compatibility of valuations at call (return) sites with initial (final) valuations of called procedures.

$$\textsc{MatchCall}(q \xrightarrow{\overrightarrow{\mathbf{z}}' = call_j(\overrightarrow{\mathbf{t}})} q' \in \Delta_i, \nu \in \mathcal{V}_i, \nu_1 \in \mathcal{V}_j) \stackrel{def}{\equiv}$$

$$\bigwedge \begin{cases} \nu(x) = \nu_1(x) \text{ for all } x \in \mathbf{x}_g & \text{(values of global variables match)} \\ \nu((\overrightarrow{\mathbf{t}})_k) = \nu_1((\overrightarrow{\mathbf{x}}_i^{in})_k) \text{ for all } 1 \leq k \leq |\overrightarrow{\mathbf{t}}| & \text{(input values match)} \end{cases}$$

$$\textsc{MatchCallRet}(q \xrightarrow{\overrightarrow{\mathbf{z}}'=call_j(\overrightarrow{\mathbf{t}})} q' \in \Delta_i, \nu, \nu' \in \mathcal{V}_i, \nu_1, \nu_2 \in \mathcal{V}_j) \stackrel{def}{\equiv}$$

$$\bigwedge \begin{cases} \textsc{MatchCall}(q \xrightarrow{\overrightarrow{\mathbf{z}}'=call_j(\overrightarrow{\mathbf{t}})} q', \nu, \nu_1) \\ \nu'(x) = \nu_2(x) \text{ for all } x \in \mathbf{x}_g & \text{(values of global variables match)} \\ \nu'((\overrightarrow{\mathbf{z}})_k) = \nu_2((\overrightarrow{\mathbf{x}}_i^{out})_k) \text{ for all } 1 \leq k \leq |\overrightarrow{\mathbf{z}}| & \text{(output values match)} \\ \nu(x) = \nu'(x) \text{ for all } x \in \mathbf{x}_i \setminus \overrightarrow{\mathbf{z}} & \text{(frame rule)} \end{cases}$$

Informally, MatchCall evaluates to true if a valuation ν at a call site of a procedure P_i is compatible with a valuation ν_1 of a called procedure P_j. MatchCallRet moreover requires that a valuation ν' at a return site of a procedure P_i is compatible with a valuation ν_2 of a called procedure P_j and that the frame rule is respected.

Program Summaries

For each procedure $P_i = \langle \mathbf{x}_i, \overrightarrow{\mathbf{x}}_i^{in}, \overrightarrow{\mathbf{x}}_i^{out}, Q_i, q_{0,i}, q_{f,i}, q_{e,i}, \Delta_i \rangle$, we define the set of summaries compatible with P_i as $\mathcal{S}_i = \{S_i : Q_i \times Q_i \to \mathcal{V}_i \times \mathcal{V}_i\}$. Intuitively, a summary S_i of a procedure P_i maps each pair of control states $(q, q') \in Q_i \times Q_i$ to a relation $S_i(q, q') \in \mathcal{V}_i \times \mathcal{V}_i$ between valuations in q and q' that are feasible by an execution of P_i that starts in q and ends in q'. For a program $\mathcal{P} = \langle \mathbf{x}_g, \{P_1, \ldots, P_n\}, P_m \rangle$, the set of summaries compatible with $\mathcal{P}$ is defined as $\mathcal{S} = \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$.

Given two configurations $\langle q, \nu \rangle$ and $\langle q', \nu' \rangle$ of a procedure P_i, the configuration $\langle q', \nu' \rangle$ is said to be an *immediate successor* of $\langle q, \nu \rangle$, with respect to a program summary $S = (S_1, \ldots, S_n) \in \mathcal{S}$, if and only if either:

- $q \xrightarrow{R} q' \in \Delta_i$ and $\nu, \nu' \models R$ (internal action)
- $q \xrightarrow{\overrightarrow{\mathbf{z}}=call_j(\overrightarrow{\mathbf{t}})} q' \in \Delta_i$ and $\textsc{MatchCallRet}(q \xrightarrow{\overrightarrow{\mathbf{z}}=call_j(\overrightarrow{\mathbf{t}})} q', \nu, \nu', \nu_1, \nu_2)$ for some $(\nu_1, \nu_2) \in S_j(q_{0,j}, q_{f,j})$ (successful call)
- $q' = q_{e,i}$, $q \xrightarrow{\overrightarrow{\mathbf{z}}=call_j(\overrightarrow{\mathbf{t}})} q'' \in \Delta_i$ and $\textsc{MatchCallRet}(q \xrightarrow{\overrightarrow{\mathbf{z}}=call_j(\overrightarrow{\mathbf{t}})} q'', \nu, \nu', \nu_1, \nu_2)$ for some $q'' \in Q_i$ and for some $(\nu_1, \nu_2) \in S_j(q_{0,j}, q_{e,j})$ (erroneous call)

A *finite run* of length k of a procedure P_i from q to q', under a program summary $S \in \mathcal{S}$, is a finite sequence $\langle q_0, \nu_0 \rangle \to \langle q_1, \nu_1 \rangle \to \ldots \to \langle q_k, \nu_k \rangle$, such that $q = q_0$, $q' = q_k$, and $\langle q_{i+1}, \nu_{i+1} \rangle$ is an immediate successor of $\langle q_i, \nu_i \rangle$ with respect to S, for all $0 \leq i < k$. An *infinite run* of a procedure P_i, from a control state q, under a program summary $S \in \mathcal{S}$, is an infinite sequence $\langle q_0, \nu_0 \rangle \to \langle q_1, \nu_1 \rangle \to \ldots$ such that $q = q_0$ and $\langle q_{i+1}, \nu_{i+1} \rangle$ is an immediate successor of $\langle q_i, \nu_i \rangle$ with respect to S, for all $i \geq 0$.

The *summary of a procedure* P_i under a program summary $S \in \mathcal{S}$ is a mapping $[\![P_i]\!]_S \in \mathcal{S}_i$ defined for each $q, q' \in Q_i$ as

$$[\![P_i]\!]_S(q, q') \stackrel{def}{=} \{(\nu, \nu') \mid (q, \nu) \to \ldots \to (q', \nu') \text{ is a finite run of } P_i \text{ of length } k \geq 1 \text{ under } S\}$$

The *summary of a program* $\mathcal{P}$, denoted by $[\![\mathcal{P}]\!]$, is defined as the least fixpoint of the function

$$\begin{array}{rcl} \mathcal{S} & \to & \mathcal{S} \\ S \in \mathcal{S} & \mapsto & ([\![P_1]\!]_S, \ldots, [\![P_n]\!]_S) \end{array}$$

We denote the components of the program summary $[\![\mathcal{P}]\!]$ as $[\![\mathcal{P}]\!] = ([\![P_1]\!], \ldots, [\![P_n]\!])$. Intuitively, $[\![P_i]\!]$ represents the reachability relation between two arbitrary control states of P_i in a finite and non-zero number of steps. Analogically, we define $[\![\mathcal{P}]\!]^* = ([\![P_1]\!]^*, \ldots, [\![P_n]\!]^*)$ to represent the reachability relation in arbitrary finite number of steps, including zero, by referring to

$$[\![P_i]\!]^*_S(q, q') \stackrel{def}{=} \{(\nu, \nu') \mid (q, \nu) \to \ldots \to (q', \nu') \text{ is a finite run of } P_i \text{ of length } k \geq 0 \text{ under } S\}$$

Further, we write $[\![\mathcal{P}]\!]^f = ([\![P_1]\!]^f, \ldots, [\![P_n]\!]^f)$, where $[\![P_i]\!]^f \stackrel{def}{=} [\![P_i]\!](q_{0,i}, q_{f,i})$, to denote only the summaries from the initial to the final control state. Further, we define the *final state summary* as $[\![\mathcal{P}]\!]^f_\exists = ([\![P_1]\!]^f_\exists, \ldots, [\![P_n]\!]^f_\exists)$ where

$$[\![P_i]\!]^f_\exists \stackrel{def}{=} \exists(\mathbf{x}_i \cup \mathbf{x}'_i) \setminus (\overrightarrow{\mathbf{x}}_i^{in} \cup \overrightarrow{\mathbf{x}}_i'^{out}) \,.\, [\![P_i]\!]^f$$

Intuitively, $[\![\mathcal{P}]\!]^f_\exists$ is computed from $[\![\mathcal{P}]\!]^f$ by eliminating variables that are not in the signature of P_i. Similarly, we define $[\![\mathcal{P}]\!]^e = ([\![P_1]\!]^e, \ldots, [\![P_n]\!]^e)$ and the *error state summary* $[\![\mathcal{P}]\!]^e_\exists = ([\![P_1]\!]^e_\exists, \ldots, [\![P_n]\!]^e_\exists)$, where $[\![P_i]\!]^e \stackrel{def}{=} [\![P_i]\!](q_{0,i}, q_{e,i})$ and

$$[\![P_i]\!]^e_\exists \stackrel{def}{=} \exists(\mathbf{x}_i \cup \mathbf{x}'_i) \setminus (\overrightarrow{\mathbf{x}}_i^{in} \cup \overrightarrow{\mathbf{x}}_i'^{out}) \,.\, [\![P_i]\!]^e$$

Weakest (Non-)Termination Sets

We proceed with a definition of the weakest non-termination set for *non-recursive* programs. The *non-termination set* of a procedure P_i with respect to a non-termination valuation $\mathcal{N} = (\mathcal{N}_1, \ldots, \mathcal{N}_n) \in \mathcal{V}_1 \times \ldots \times \mathcal{V}_n$ is denoted as $[\![P_i]\!]^{nt}_\mathcal{N}$ and is defined as follows. For each valuation $\nu \in \mathcal{V}_i$, $\nu \in [\![P_i]\!]^{nt}_\mathcal{N}$ if and only if either

1. there exists an infinite run in P_i under $[\![\mathcal{P}]\!]$ that starts in $\langle q_{0,i}, \nu \rangle$
2. there exists a finite run of length $k \geq 0$ in P_i under $[\![\mathcal{P}]\!]$ that starts in $\langle q_{0,i}, \nu \rangle$, ends in $\langle q', \nu' \rangle$ for some $q' \in Q_i, \nu' \in \mathcal{V}_i$ and moreover, there exists a transition $q' \xrightarrow{\overrightarrow{\mathbf{z}} = call_j(\overrightarrow{\mathbf{t}})} q'' \in \Delta_i$ and a valuation $\nu'' \in \mathcal{N}_j$ such that

$$\textsc{MatchCall}(q' \xrightarrow{\overrightarrow{\mathbf{z}} = call_j(\overrightarrow{\mathbf{t}})} q'', \nu', \nu'') \Leftrightarrow \top$$

Intuitively, the first point above captures all initial configurations from which there exists an infinite run in P_i on which some control state of P_i appears infinitely often. The second point captures all initial configurations from which there exists a run which

enters one of the called procedures P_j and moreover, P_j is entered in a configuration from which a non-terminating run, captured by the non-termination valuation $\mathcal{N}$, exists.

Since each non-terminating run in a non-recursive program must loop infinitely in one of the procedures, the weakest non-termination set of a non-recursive program $\mathcal{P}$, denoted as $[\![\mathcal{P}]\!]^{nt}$, can be defined as the least fixpoint of the following function, where $\mathcal{V} = \mathcal{V}_1 \times \ldots \times \mathcal{V}_n$

$$\begin{array}{rcl} \mathcal{V} & \rightarrow & \mathcal{V} \\ \mathcal{N} \in \mathcal{V} & \mapsto & ([\![P_1]\!]^{nt}_{\mathcal{N}}, \ldots, [\![P_n]\!]^{nt}_{\mathcal{N}}) \end{array}$$

Notice that since $\mathcal{P}$ is non-recursive, the fixpoint is reached after at most n steps. After the first step, $\mathcal{N}_i$ contains exactly the valuations from which a run that loops infinitely in P_i exists (this corresponds to the point 1 above). In at most $n-1$ following iterations, the fixpoint is reached by propagating the sets $\mathcal{N}_i$, computed in the first step, to the calling procedures (this corresponds to the point 2 above).

We denote the components of $[\![\mathcal{P}]\!]^{nt}$ as $[\![\mathcal{P}]\!]^{nt} = ([\![P_1]\!]^{nt}, \ldots, [\![P_n]\!]^{nt})$. Finally, we define $[\![\mathcal{P}]\!]^{nt}_{\exists} = ([\![P_1]\!]^{nt}_{\exists}, \ldots, [\![P_n]\!]^{nt}_{\exists})$, where

$$[\![P_i]\!]^{nt}_{\exists} \stackrel{def}{=} \exists(\mathbf{x}_i \setminus \overrightarrow{\mathbf{x}}^{in}_i) \,.\, [\![P_i]\!]^{nt}$$

Intuitively, $[\![P_i]\!]^{nt}_{\exists}$ is obtained from $[\![P_i]\!]^{nt}$ by eliminating variables that are not in the signature of P_i.

Finally, we define the *weakest termination set* $[\![\mathcal{P}]\!]^{t}$, a dual of $[\![\mathcal{P}]\!]^{nt}$, as $[\![\mathcal{P}]\!]^{t} = ([\![P_1]\!]^{t}, \ldots, [\![P_n]\!]^{t})$, where $[\![P_i]\!]^{t} = \mathcal{V}_i \setminus [\![P_i]\!]^{nt}$. Similarly, $[\![\mathcal{P}]\!]^{t}_{\exists}$ is defined as $[\![\mathcal{P}]\!]^{t}_{\exists} = ([\![P_1]\!]^{t}_{\exists}, \ldots, [\![P_n]\!]^{t}_{\exists})$, where $[\![P_i]\!]^{t}_{\exists} \stackrel{def}{=} \mathcal{V}^{in}_i \setminus [\![P_i]\!]^{nt}_{\exists}$ and $\mathcal{V}^{in}_i = \mathbb{Z}^{\overrightarrow{\mathbf{x}}^{in}_i \cup \mathbf{x}_g}$.

2.2.3 Reachability and Termination Problems

Reachability Problem

Informally, the *reachability problem* asks whether a program has a run that starts in the initial control state and ends in the error control state. If no such run exists, the program is said to be *safe*. The reachability problem can be defined by referring to the summary semantics defined in Section 2.2.1.

Definition 2.1 *Let $\mathcal{P} = \langle \mathbf{x}_g, \{P_1, \ldots, P_n\}, P_m \rangle$ be a program and $[\![\mathcal{P}]\!] = ([\![P_1]\!], \ldots, [\![P_n]\!])$ be its summary. $\mathcal{P}$ is said to be* safe *if and only if $[\![P_m]\!]^{e}_{\exists} = \emptyset$. The* reachability problem *asks whether $\mathcal{P}$ is safe.*

Termination Problem

Informally, the *termination problem* asks whether a given program stops for every possible input configuration. The problem of determining the set of configurations from which a program terminates on all paths is called the *conditional termination problem*. This set is called the *weakest termination set*. We next formalize these problems for non-recursive programs.

Definition 2.2 *Let* $\mathcal{P} = \langle \mathbf{x}_g, \{P_1, \ldots, P_n\}, P_m \rangle$ *be a non-recursive program and let* $[\![\mathcal{P}]\!]^t = ([\![P_1]\!]^t, \ldots, [\![P_n]\!]^t)$ *be its weakest termination set. The* conditional termination problem *asks to compute* $[\![P_m]\!]^t_\exists$. *Program* $\mathcal{P}$ *is* terminating *if and only if* $[\![P_m]\!]^t_\exists = \mathcal{V}_i^{in}$. *The* termination problem *asks whether* $\mathcal{P}$ *is terminating.*

Since $[\![P_i]\!]^t_\exists \stackrel{def}{=} \mathcal{V}_i^{in} \setminus [\![P_i]\!]^{nt}_\exists$, it follows that a program $\mathcal{P}$ is terminating if and only if $[\![P_m]\!]^{nt}_\exists = \emptyset$.

2.3 Difference Bounds Relations

Difference bounds constraints are known as *zones* in the context of timed automata verification [AD91] and abstract interpretation [Min06]. They are defined syntactically as conjunctions of atomic propositions of the form $x - y \leq c$, where x and y are variables and c is an integer constant. Difference bounds constraints can be represented as matrices and graphs. Moreover, their canonical form, useful for efficient inclusion checks, can be computed by the classical Floyd-Warshall algorithm. We report on these results in Section 2.3.1.

Difference bounds relations are defined as difference bounds constraints where variables can be also primed (e.g. $x - x' \leq 0$). Intuitively, primed variables denote future values of variables and unprimed variables denote current values of variables. Difference bounds relations have been studied by Comon and Jurski who showed, in [CJ98], that their transitive closure is Presburger definable. Their proof was subsequently simplified in [BIL09], using the notion of *zigzag* automata. Intuitively, zigzag automaton corresponding to a difference bounds relation R is a finite weighted automaton that encodes m-th power of R by minimal runs of length $m + 2$. As we show in Chapter 4, zigzag automata can be also used in proving *periodicity* of difference bounds relations, which allows to compute R^+ efficiently, using the algorithm presented in Chapter 3. Furthermore, they also play a crucial role in Chapter 6 in designing a PTIME algorithm computing the weakest termination sets and in proving existence of linear ranking functions for difference bounds and octagonal relations. We give definitions of difference bounds relations and zigzag automata in Section 2.3.2 and Section 2.3.3, respectively. In the rest of this section, let $\mathbf{x} = \{x_1, x_2, ..., x_N\}$ be a set of variables ranging over $\mathbb{Z}$.

2.3.1 Difference Bounds Constraints

The following definition formalizes the notion of a difference bounds constraint.

Definition 2.3 *A formula* $\phi(\mathbf{x})$ *is a* difference bounds constraint *if it is equivalent to a finite conjunction of atomic propositions of the form* $x_i - x_j \leq a_{ij},\ 1 \leq i, j \leq N, i \neq j$, *where* $a_{ij} \in \mathbb{Z}$.

For instance, $x - y = 5$ is a difference bounds constraint, as it is equivalent to $x - y \leq 5 \ \wedge \ y - x \leq -5$. In practice, difference bounds constraints are represented either as matrices or as graphs:

Definition 2.4 *Let* $\mathbf{x} = \{x_1, x_2, ..., x_N\}$ *be a set of variables ranging over* $\mathbb{Z}$ *and* $\phi(\mathbf{x})$ *be a difference bounds constraint. Then a* difference bounds matrix *(DBM) representing* ϕ *is an* $N \times N$ *matrix* M_ϕ *such that:*

$$(M_\phi)_{i,j} = \begin{cases} \alpha_{i,j} & \text{if } (x_i - x_j \leq \alpha_{i,j}) \in AP(\phi) \\ \infty & \text{otherwise} \end{cases}$$

Definition 2.5 *Let* $\mathbf{x} = \{x_1, x_2, ..., x_N\}$ *be a set of variables ranging over* $\mathbb{Z}$ *and* $\phi(\mathbf{x})$ *be a difference bounds constraint. Then* ϕ *can be represented as a weighted graph* $\mathcal{G}_\phi = (\mathbf{x}, \rightarrow)$*, where each vertex corresponds to a variable, and there is an edge* $x_i \xrightarrow{a_{ij}} x_j$ *in* $\mathcal{G}_\phi$ *if and only if there exists a constraint* $x_i - x_j \leq a_{ij}$ *in* ϕ*. This graph is also called a* constraint graph*.*

Clearly, M_ϕ is the incidence matrix of $\mathcal{G}_\phi$. If $M \in \mathbb{Z}_\infty^{N \times N}$ is a DBM, the corresponding difference bounds constraint is defined as $\Phi_M \Leftrightarrow \bigwedge_{M_{ij} < \infty} x_i - x_j \leq M_{ij}$. We denote by $\|\phi\| = \sum_{(x_i - x_j \leq a_{ij}) \in AP(\phi)} |a_{ij}|$ the sum of absolute values of all coefficients of ϕ. The restriction of a DBM M_ϕ to variables $\mathbf{z} \subseteq \mathbf{x}$, denoted as $(M_\phi)_{\downarrow \mathbf{z}}$, is a matrix obtained by erasing the rows and columns of M_ϕ which encode constraints that involve variables $\mathbf{z} \setminus \mathbf{x}$. For two difference bounds matrices M_1, M_2, we write $M_1 = M_2$ if and only if $(M_1)_{ij} = (M_2)_{ij}$ for all $1 \leq i, j \leq N$ and $M_1 \leq M_2$ if and only if $(M_1)_{ij} \leq (M_2)_{ij}$ for all $1 \leq i, j \leq N$.

A DBM M is said to be *consistent* if and only if its corresponding constraint ϕ_M is consistent. The following proposition relates the consistency of ϕ to the existence of an elementary negative weight cycle of $\mathcal{G}_\phi$.

Proposition 2.6 *Let* ϕ *be a difference bounds constraint and* $\mathcal{G}_\phi$ *be the constraint graph of* ϕ*. Then, the following statements are equivalent:*

- ϕ *is consistent*
- $\mathcal{G}_\phi$ *contains an elementary negative weight cycle*

Proof: See e.g. [CSRL01], §25.5. □

The next definition gives a canonical form for consistent DBMs.

Definition 2.7 *A consistent DBM* $M \in \mathbb{Z}_\infty^{N \times N}$ *is said to be* closed *if and only if* $M_{ii} = 0$ *and* $M_{ij} \leq M_{ik} + M_{kj}$*, for all* $1 \leq i, j, k \leq N$*.*

Given a consistent DBM $M \in \mathbb{Z}^N \times \mathbb{Z}^N$, we denote the (unique) closed DBM by M^*. It is well-known that, if M is consistent, then M^* is unique, and can be computed from M in time $\mathcal{O}(N^3)$, by the classical Floyd-Warshall algorithm [CSRL01]. Consistency of M can be checked by the Floyd-Warshall algorithm too. By Proposition 2.6, it amounts to checking whether $M^*_{ii} < 0$ for some $1 \leq i \leq N$. The closed form is needed to check the equivalence and entailment of two difference bounds constraints.

Proposition 2.8 ([Min06]) *Let ϕ_1 and ϕ_2 be consistent difference bounds constraints. Then,*

- $\phi_1 \Leftrightarrow \phi_2$ *if and only if* $M^*_{\phi_1} = M^*_{\phi_2}$,
- $\phi_1 \Rightarrow \phi_2$ *if and only if* $M^*_{\phi_1} \leq M^*_{\phi_2}$.

The following proposition shows that given a difference bounds constraint $\phi(\mathbf{x})$, the formula $\exists x_k.\phi$ is a difference bounds constraint as well, and its closed DBM is effectively computable from M^*_ϕ.

Proposition 2.9 *Let $\phi(\mathbf{x})$, $\mathbf{x} = \{x_1, \ldots, x_N\}$, be a consistent difference bounds constraint. Further, let $1 \leq k \leq N$ and M' be the restriction of M^*_ϕ to $\mathbf{x} \setminus \{x_k\}$. Then, M' is closed and $\Phi(M') \Leftrightarrow \exists x_k.\phi(\mathbf{x})$. Moreover, the constraint graph $\mathcal{G}'$ corresponding to $\Phi(M')$ is obtained by erasing the vertex x_k together with the incident arcs from the graph $\mathcal{G}_{\Phi(M^*_\phi)}$.*

Proof: We use the following notation: for a $N \times N$ DBM M we denote by $\gamma(M) = \{\mathbf{v} \in \mathbb{Z}^N \mid v_i - v_j \leq m_{i,j}, 1 \leq i, j \leq N\}$ the set of *concretizations* of M. Notice that a DBM M is consistent if and only if $\gamma(M) \neq \emptyset$.

Clearly, M' is the incidence matrix of $\mathcal{G}'$. Without loss of generality, we assume that $k = N$. It is sufficient to show that:

$$\gamma(M') = \{(v_1, v_2, \ldots, v_{N-1}) \mid (v_1, v_2, \ldots, v_{N-1}, v) \in \gamma(M^*_\phi) \text{ for some } v \in \mathbb{Z}\}$$

The "$\supseteq$" direction is obvious, since M' is the restriction of M^*_ϕ to $\{x_1, x_2, \ldots, x_{N-1}\}$. For the "$\subseteq$" direction, we must show that there exists $v \in \mathbb{Z}$ such that $v_i - v \leq (M^*_\phi)_{i,N}$ and $v - v_j \leq (M^*_\phi)_{N,j}$, for all $1 \leq i, j \leq N$. But this amounts to $v_i - (M^*_\phi)_{i,N} \leq (M^*_\phi)_{N,j} + v_j$, for all $1 \leq i, j \leq N$. Since $(v_1, v_2, \ldots, v_{N-1}) \in \gamma(M')$ we have $v_i - v_j \leq (M^*_\phi)_{i,j}$, for all $1 \leq i, j \leq n$. Since M^*_ϕ is closed, $(M^*_\phi)_{i,j} \leq (M^*_\phi)_{i,N} + (M^*_\phi)_{N,j}$, which leads to the conclusion. DBM M' is closed as a direct consequence of the fact that M^*_ϕ is closed. □

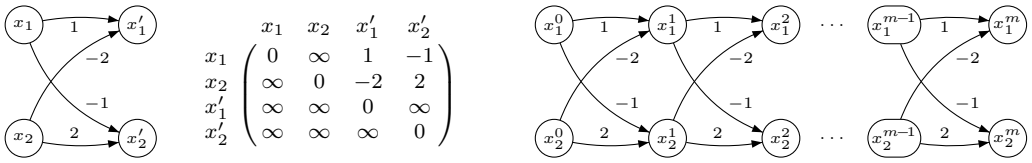
2.3.2 Difference Bounds Relations and Their Powers

We first define difference bounds relations.

Definition 2.10 *Let $\mathbf{x} = \{x_1, \ldots, x_N\}$ be a set of variables. A relation $R \in \mathbb{Z}^N \times \mathbb{Z}^N$ is a* difference bounds relation *if it can be defined by a difference bounds constraint $R(\mathbf{x}, \mathbf{x}')$.*

The class of relations defined by difference bounds constraints over the variables $\mathbf{x} \cup \mathbf{x}'$ is denoted $\mathcal{R}_{db}$ in the following. A consequence of Proposition 2.9 is that $\mathcal{R}_{db}$ is closed under composition.

Proposition 2.11 *$\mathcal{R}_{db}$ is closed under intersection and composition.*



(a) $\mathcal{G}_R$ (b) M_R^* (c) $\mathcal{G}_R^m$

Figure 2.1: Graph and matrix representation of a relation. Graph unfolding.

Proof: Let $R_1(\mathbf{x}, \mathbf{x}')$, $R_2(\mathbf{x}, \mathbf{x}')$ be difference bounds constraints defining difference bounds relations. By Definition 2.3, the conjunction $R_1(\mathbf{x}, \mathbf{x}') \wedge R_2(\mathbf{x}, \mathbf{x}')$ is a difference bounds constraint too. The composition of relations $R_1 \circ R_2$ can be defined as $\exists \mathbf{x}'' \,.\, (R_1(\mathbf{x}, \mathbf{x}'') \wedge R_2(\mathbf{x}'', \mathbf{x}'))$ which is again a difference bounds constraint, by Definition 2.3 and Proposition 2.9. □

Example 2.12 *Let* $R(x_1, x_2, x'_1, x'_2) \Leftrightarrow x_1 - x'_1 \leq 1 \wedge x_1 - x'_2 \leq -1 \wedge x_2 - x'_1 \leq -2 \wedge x_2 - x'_2 \leq 2$ *be a difference bounds relation. Figure 2.1a shows the graph representation* $\mathcal{G}_R$ *and Figure 2.1b the closed DBM representation of* R. □

Given a difference bounds relation $R(\mathbf{x}, \mathbf{x}')$, we define the m-times concatenation of $\mathcal{G}_R$ with itself.

Definition 2.13 *Let* $R(\mathbf{x}, \mathbf{x}')$, $x = \{x_1, \ldots, x_N\}$, *be a difference bounds relation and* $\mathcal{G}_R$ *be its constraint graph. The* m-times unfolding *of* $\mathcal{G}_R$ *is defined as* $\mathcal{G}_R^m = (\bigcup_{k=0}^{N} \mathbf{x}^{(k)}, \rightarrow)$, *where* $\mathbf{x}^{(k)} = \{x_i^{(k)} \mid 0 \leq i \leq N\}$ *and for all* $0 \leq k < N$,

- $(x_i^{(k)} \xrightarrow{c} x_j^{(k)}) \in \rightarrow$ *if and only if* $(x_i - x_j \leq c) \in AP(\phi)$
- $(x_i^{(k)} \xrightarrow{c} x_j^{(k+1)}) \in \rightarrow$ *if and only if* $(x_i - x'_j \leq c) \in AP(\phi)$
- $(x_i^{(k+1)} \xrightarrow{c} x_j^{(k)}) \in \rightarrow$ *if and only if* $(x'_i - x_j \leq c) \in AP(\phi)$
- $(x_i^{(k+1)} \xrightarrow{c} x_j^{(k+1)}) \in \rightarrow$ *if and only if* $(x'_i - x'_j \leq c) \in AP(\phi)$

Each constraint in R^m corresponds to a path between extremal points in $\mathcal{G}_R^m$. Notice that, since $\mathcal{R}_{db}$ is closed under relational composition, then $R^m \in \mathcal{R}_{db}$ for any $m > 0$. Then we have:

$$R^m \Leftrightarrow \bigwedge_{1 \leq i,j \leq N} \begin{array}{l} x_i - x_j \leq \min\{x_i^0 \rightarrow x_j^0\} \ \wedge \ x'_i - x'_j \leq \min\{x_i^m \rightarrow x_j^m\} \wedge \\ x_i - x'_j \leq \min\{x_i^0 \rightarrow x_j^m\} \ \wedge \ x'_i - x_j \leq \min\{x_i^m \rightarrow x_j^0\} \end{array}$$

where $\min\{x_i^p \rightarrow x_j^q\}$ is the minimal weight between all paths among the extremal vertices x_i^p and x_j^q in $\mathcal{G}_R^m$, for $p, q \in \{0, m\}$.

Example 2.14 *Figure 2.1c depicts the m-times unfolding of $\mathcal{G}_R$ for the relation $R \Leftrightarrow x_1 - x'_1 \leq 1 \wedge x_1 - x'_2 \leq -1 \wedge x_2 - x'_1 \leq -2 \wedge x_2 - x'_2 \leq 2$.* □

The set of paths between any two extremal points in $\mathcal{G}_R^m$ can be seen as words over the finite alphabet of subgraphs of $\mathcal{G}_R^m$ that are accepted by a finite weighted automaton called *zigzag automaton* [BIL09]. In the following section, we give the definition of these automata.

2.3.3 Zigzag Automata

This section defines zigzag automata, which can seen as recognizers of powers of difference bounds relations. Intuitively, a zigzag automaton corresponding to a difference bounds relation R is a finite weighted automaton that encodes m-th power of R by minimal runs of length $m + 2$.

Alphabet and Words

Without losing generality, in the following we work with a simplified (yet equivalent) form of difference bounds relations: all constraints of the form $x - y \leq \alpha$ are replaced by $x - t' \leq \alpha \ \wedge \ t' - y \leq 0$, and all constraints of the form $x' - y' \leq \alpha$ are replaced by $x' - t \leq \alpha \ \wedge \ t - y' \leq 0$, by introducing fresh variables $t \notin \mathbf{x}$. In other words, we can assume that the constraint graph $\mathcal{G}_R$ corresponding to R is *bipartite*, i.e. it does only contain edges from $\mathbf{x}$ to $\mathbf{x}'$ and vice versa.

A path π in $\mathcal{G}_R^m$ between, say, x^0 and y^m, with $x, y \in \mathbf{x}$ is represented by a word $w = w_1 \dots w_m$ of length m, as follows: the w_i symbol represents *simultaneously* all edges of π that involve only nodes from $\mathbf{x}^{i-1} \cup \mathbf{x}^i$, $1 \leq i \leq m$. Since we assumed that $\mathcal{G}_R^m$ is bipartite, it is easy to see that, for a path from x^0 to y^m, coded by a word w, the number of times the w_i symbol is traversed by the path is odd, whereas for a path from x^0 to y^0, or from x^m to y^m, this number is even. Hence the names of *even* and *odd automata*.

Given a difference bounds relation R, the *even alphabet* of R, denoted as Σ_R^e, is the set of all graphs satisfying the following conditions, for each $G \in \Sigma_R^e$:

1. the set of nodes of G is $\mathbf{x} \cup \mathbf{x}'$
2. for any $x, y \in \mathbf{x} \cup \mathbf{x}'$, there is an edge labeled with $\alpha \in \mathbb{Z}$ from x to y, only if the constraint $x - y \leq \alpha$ occurs in ϕ
3. the in-degree and out-degree of each node are at most one
4. the number of edges from $\mathbf{x}$ to $\mathbf{x}'$ equals the number of edges from $\mathbf{x}'$ to $\mathbf{x}$

Notice that the number of edges in all symbols of Σ_R^e is even.

The *odd alphabet* of R, denoted by Σ_R^o, is defined in the same way, with the exception of the last condition, which becomes:

4. the difference between the number of edges from $\mathbf{x}$ to $\mathbf{x}'$ and the number of edges from $\mathbf{x}'$ to $\mathbf{x}$ is either 1 or -1

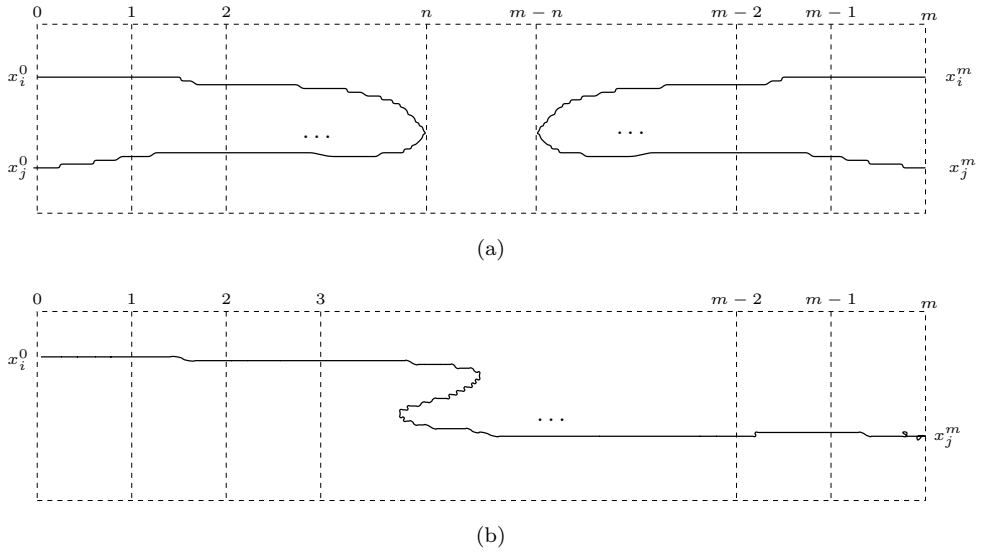


Figure 2.2: Runs of Even and Odd Automata

Notice that the number of edges in all symbols of Σ_R^o is odd.

Let $\Sigma_R = \Sigma_R^e \cup \Sigma_R^o \cup \{\epsilon\}$ be the alphabet of the zigzag automaton for R, where ϵ is a special symbol of weight 0. The weight of any symbol $G \in \Sigma_R^e \cup \Sigma_R^o$, denoted $\omega(G)$, is the sum of the weights that occur on its edges. For a word $w = w_1 w_2 \dots w_n \in \Sigma_R^*$, we define its weight as $\omega(w) = \sum_{i=1}^{n} \omega(w_i)$.

Construction of Zigzag Automata

We are now ready for the definition of automata recognizing words that represent encodings of paths from $\mathcal{G}_R^m$. The *even automaton* recognizes paths that start and end on the same side of $\mathcal{G}_R^m$ i.e., either paths from x_i^0 to x_j^0, or from x_i^m to x_j^m, for some $1 \le i, j \le N$, respectively. We call the automata recognizing paths from x_i^0 to x_j^0 *forward* even automata, and the ones recognizing paths from x_i^m to x_j^m *backward* even automata (Figure 2.2 (a)). The *odd automata* recognize paths from one side of $\mathcal{G}_R^m$ to another. The automata recognizing paths from x_i^0 to x_j^m are called *forward* odd automata, whereas the ones recognizing paths from x_i^m to x_j^0 are called *backward* odd automata (Figure 2.2 (b)).

The even and odd automata share the same alphabet and transition table, while the differences are in the sets of initial and final states. The common transition table is defined as $T_R = \langle Q, \Delta, w \rangle$, where Q is the set of control states defined as:

$$
\begin{array}{rcl}
Q & = & Q_g \cup \bigcup_{1 \le i,j \le N} (Q_{ij}^{ef} \cup Q_{ij}^{eb} \cup Q_{ij}^{of} \cup Q_{ij}^{ob}) \text{ where} \\
Q_g & = & \{l, r, lr, rl, \perp\}^N \\
Q_{ij}^{ef} & = & \{I_{ij}^{ef}, F^{ef}\} \qquad Q_{ij}^{eb} = \{I^{eb}, F_{ij}^{eb}\} \\
Q_{ij}^{of} & = & \{I_i^{of}, F_j^{of}\} \qquad Q_{ij}^{ob} = \{I_i^{ob}, F_j^{ob}\}
\end{array}
$$

The $\{l, r, lr, rl, \perp\}$ components of states in Q_g capture the direction of incoming and outgoing edges (l for a path traversing from right to left, r for a path traversing from left to right, lr for a right incoming and right outgoing path, rl for a left incoming and left outgoing path, and $\perp$ when there are no incoming nor outgoing edges from that node.). Given $1 \leq i, j \leq N$, the sets $Q_{ij}^{ef}, Q_{ij}^{eb}, Q_{ij}^{of}, Q_{ij}^{ob}$ contain the initial and the final state in even forward (ef), even backward (eb), odd forward (of), and odd backward (ob) zigzag automaton corresponding to i, j, respectively. The four automata recognize paths from $x_i^{(0)}$ to $x_j^{(0)}$ (ef), from $x_i^{(0)}$ to $x_j^{(0)}$ (eb), from $x_i^{(0)}$ to $x_j^{(m)}$ (of), and from $x_i^{(m)}$ to $x_j^{(0)}$ (ob) in $\mathcal{G}_R^m$, respectively.

The set of transitions Δ is defined as:

$$\Delta \quad = \quad \Delta_g \cup \Delta_l \bigcup_{1 \leq i,j \leq N} (\Delta_{ij}^{ef} \cup \Delta_{ij}^{eb} \cup \Delta_{ij}^{of} \cup \Delta_{ij}^{ob})$$

There is a transition

$$\langle q_1 \dots q_N \rangle \xrightarrow{G} \langle q'_1, \dots, q'_N \rangle$$

in Δ_g if and only if the following conditions hold, for all $1 \leq i \leq N$:

- $q_i = l$ iff G has one edge whose destination is x_i, and no other edge involving x_i.
- $q'_i = l$ iff G has one edge whose source is x'_i, and no other edge involving x'_i.
- $q_i = r$ iff G has one edge whose source is x_i, and no other edge involving x_i.
- $q'_i = r$ iff G has one edge whose destination is x'_i, and no other edge involving x'_i.
- $q_i = lr$ iff G has exactly two edges involving x_i, one having x_i as source, and another as destination.
- $q'_i = rl$ iff G has exactly two edges involving x'_i, one having x'_i as source, and another as destination.
- $q'_i \in \{lr, \perp\}$ iff G has no edge involving x'_i.
- $q_i \in \{rl, \perp\}$ iff G has no edge involving x_i.

Some even paths in $\mathcal{G}_R^m$ may be of length strictly less than m. Since we want to recognize these path by runs of length $m+2$, we need several zero weight self-loop transitions:

$$\Delta_l = \{F^{ef} \xrightarrow{\epsilon} F^{ef}, I^{eb} \xrightarrow{\epsilon} I^{eb}\}$$

Finally, we define for each $q \leq i, j \leq N$ and each of the four zigzag automata (ef, eb, of, ob), the set of transitions that are incident with an initial or a final control state of the respective automaton:

$$\Delta_{ij}^{ef} \quad = \quad \begin{cases} \{I_{ij}^{ef} \xrightarrow{\epsilon} q \mid q_i = r,\, q_j = l \text{ and } q_h \in \{lr, \perp\},\, 1 \leq h \leq N,\, h \notin \{i, j\}\} & \text{if } i \neq j \\ \{I_{ij}^{ef} \xrightarrow{\epsilon} q \mid q_i = q_j = lr \text{ and } q_h \in \{lr, \perp\},\, 1 \leq h \leq N,\, h \neq i\} & \text{if } i = j \end{cases}$$

$$\bigcup \{q \xrightarrow{\epsilon} F^{ef} \mid q \in \{rl, \perp\}^N\}$$

$$\Delta_{ij}^{eb} = \begin{cases} \{q \xrightarrow{\epsilon} F_{ij}^{eb} \mid q_i = l,\ q_j = r \text{ and } q_h \in \{lr, \bot\},\ 1 \le h \le N,\ h \notin \{i,j\}\} & \text{if } i \neq j \\ \{q \xrightarrow{\epsilon} F_{ij}^{eb} \mid q_i = q_j = lr \text{ and } q_h \in \{lr, \bot\},\ 1 \le h \le N,\ h \neq i\} & \text{if } i = j \end{cases}$$

$$\bigcup \{I^{eb} \xrightarrow{\epsilon} q \mid q \in \{rl, \bot\}^N\}$$

$$\begin{aligned} \Delta_{ij}^{of} &= \{I_i^{of} \xrightarrow{\epsilon} q \mid q_i = r \text{ and } q_h \in \{lr, \bot\},\ 1 \le h \le N,\ h \neq i\} \\ &\quad \bigcup \{q \xrightarrow{\epsilon} F_j^{of} \mid q_j = r \text{ and } q_h \in \{rl, \bot\},\ 1 \le h \le N,\ h \neq j\} \end{aligned}$$

$$\begin{aligned} \Delta_{ij}^{ob} &= \{I_i^{ob} \xrightarrow{\epsilon} q \mid q_i = l \text{ and } q_h \in \{lr, \bot\},\ 1 \le h \le N,\ h \neq i\} \\ &\quad \bigcup \{q \xrightarrow{\epsilon} F_j^{ob} \mid q_j = l \text{ and } q_h \in \{rl, \bot\},\ 1 \le h \le N,\ h \neq j\} \end{aligned}$$

The weight function w maps each transition $q \xrightarrow{a} q' \in \Delta$, $q, q' \in Q, a \in \Sigma_R$ to $w(a)$. Finally, for each $1 \le i, j \le N$, we define four zigzag automata

$$\begin{array}{lll} A_{ij}^{ef} = \langle Q, \Delta, w, I_{i,j}^{ef}, F^{ef} \rangle & & A_{ij}^{of} = \langle Q, \Delta, w, I_i^{of}, F_j^{of} \rangle \\ A_{ij}^{eb} = \langle Q, \Delta, w, I^{eb}, F_{i,j}^{eb} \rangle & & A_{ij}^{ob} = \langle Q, \Delta, w, I_i^{ob}, F_j^{ob} \rangle \end{array}$$

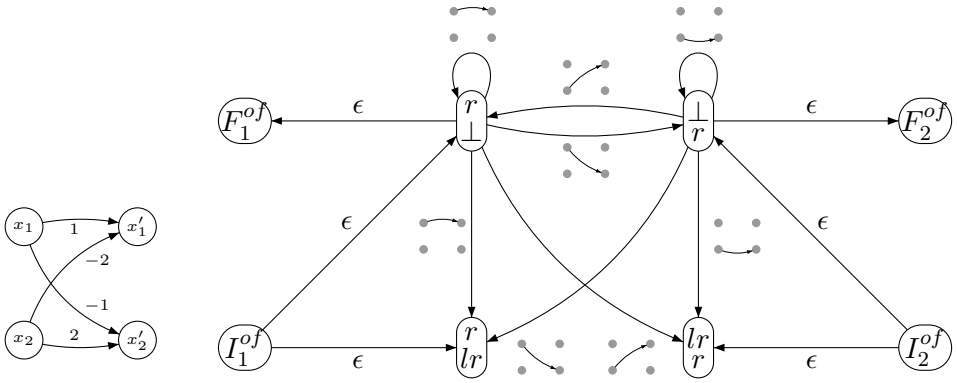
Notice that these automata share the same states and transitions, and the number of states is at most $5^N + 2N^2 + 4N + 2$, where N is the number of variables in $\mathbf{x}$.

Language of Zigzag Automata

Recall that $\mathcal{G}_R^m$ denotes the constraint graph corresponding to R^m, obtained by concatenating the constraint graph of R to itself $m > 0$ times. We say that a path in $\mathcal{G}_R^m$ *stretches between k and l*, for some $k \le l$, if the path contains at least one node from $\mathbf{x}^i$, for each $k \le i \le l$ and contains no node from $\mathbf{x}^i$, for each i such that $i < k$ or $i > l$. Intuitively, all paths from x_i^0 to x_j^0 in $\mathcal{G}_R^m$ are recognized by the automaton A_{ij}^{ef}, paths from x_i^m to x_j^m by A_{ij}^{eb} (Figure 2.2 (a)), paths from x_i^0 to x_j^m by A_{ij}^{of}, and paths from x_i^m to x_j^0 by A_{ij}^{ob} (Figure 2.2 (b)). The following lemma makes the relationship between between paths in $\mathcal{G}_R^m$ and runs in zigzag automata of length $m + 2$ precise.

Lemma 2.15 ([BIL09]) *Suppose that $\mathcal{G}_R^m$ does not have cycles of negative weight, for some $m > 0$. Then, for any $1 \le i, j \le N$, $i \neq j$, the following hold:*

1. *A_{ij}^{ef} has an accepting run of length $m + 2$ if and only if there exists a path in $\mathcal{G}_R^m$, from x_i^0 to x_j^0, that stretches between 0 and n, for some $0 \le n \le m$. Moreover, the minimal weight among all paths from x_i^0 to x_j^0 in $\mathcal{G}_R^m$, stretching from 0 to n, for some $0 \le n \le m$, equals the minimal weight among all accepting runs of A_{ij}^{ef} of length $m + 2$.*



(a) $\mathcal{G}_R$ (b) T_R

Figure 2.3: Zigzag automaton

2. *A_{ij}^{eb} has an accepting run of length $m+2$ if and only if there exists a path in $\mathcal{G}_R^m$, from x_i^m to x_j^m, that stretches between n and m, for some $0 \leq n \leq m$. Moreover, the minimal weight among all paths from x_i^m to x_j^m in $\mathcal{G}_R^m$, stretching from n to m, for some $0 \leq n \leq m$, equals the minimal weight among all accepting runs of A_{ij}^{eb}, of length $m+2$.*

3. *A_{ij}^{of} has an accepting run of length $m+2$ if and only if there exists a path in $\mathcal{G}_R^m$, from x_i^0 to x_j^m. Moreover, the minimal weight among all paths from x_i^0 to x_j^m in $\mathcal{G}_R^m$ equals the minimal weight among all accepting runs of length $m+2$.*

4. *A_{ij}^{ob} has an accepting run of length $m+2$ if and only if there exists a path in $\mathcal{G}_R^m$, from x_i^m to x_j^0. Moreover, the minimal weight among all paths from x_i^m to x_j^0 in $\mathcal{G}_R^m$ equals the minimal weight among all accepting runs of length $m+2$.*

Proof: See [BIL09], Lemmas 4.3, 4.4, 4.6 and 4.7. □

Example 2.16 *Let us show the construction of the zigzag automaton for the relation $R \Leftrightarrow x_1 - x'_1 \leq 1 \wedge x_1 - x'_2 \leq -1 \wedge x_2 - x'_1 \leq -2 \wedge x_2 - x'_2 \leq 2$. Figures 2.2(a) and (b) depict $\mathcal{G}_R$ and M_R^*. Notice that there are only forward odd paths, i.e. paths from $\mathbf{x_0}$ to $\mathbf{x}_m$ in $\mathcal{G}_R^m$ for any $m \geq 1$. The transition table $T_R = \langle Q, \Delta, w \rangle$ of the zigzag automaton is depicted in Figure 2.3 (isolated states, such as (r, l), have been removed). For instance, the automaton $A_{xy}^{ef} = \langle T_R, I_x^{of}, F_x^{of} \rangle$ recognizes a run of length $m+2$ with weight w if and only if there is a path from x_0 to x^m in $\mathcal{G}_R^m$ of length m and with weight w. There are four such paths in $\mathcal{G}_R^3$ and the Figure 2.4 shows the corresponding runs of the zigzag automaton. The second and the third runs have minimal weight.* □

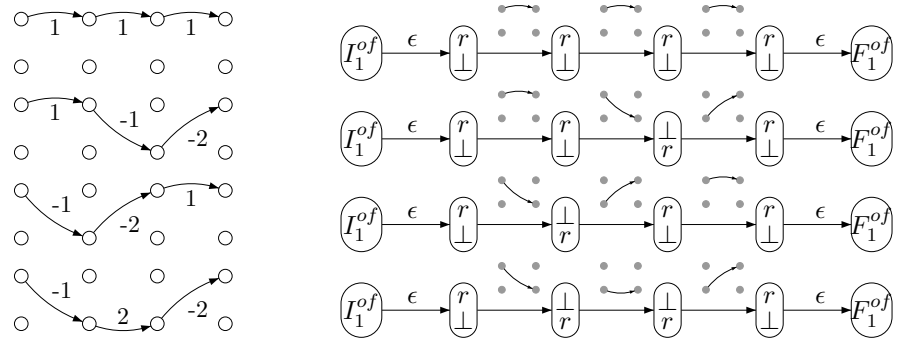


Figure 2.4: Runs

2.4 Octagonal Relations

Octagonal constraints (also known as Unit Two Variables Per Inequality or UTVPI, for short) appear in the context of abstract interpretation where they have been extensively studied as an abstract domain [Min06]. They are defined syntactically as a conjunctions of atomic propositions of the form $\pm x \pm y \leq c$, where x and y are variables and c is an integer constant. Thus, they can be seen as a generalization of difference bounds constraints. We adopt the classical representation of octagonal constraints (or octagons, for short) $\phi(x_1, \ldots, x_N)$ as difference bounds constraints $\phi(y_1, \ldots, y_{2N})$, where y_{2i-1} stands for $+x_i$ and y_{2i} stands for $-x_i$ with an implicit condition $y_{2i-1} = -y_{2i}$, for each $1 \leq i \leq N$. With this convention, [BHZ08] provides an algorithm for computing the canonical form of an octagon, by first computing the canonical form of the corresponding difference bounds constraint and subsequently *tightening* the difference bounds constraints $y_i - y_j \leq c$. We present these results in Section 2.4.1.

Octagonal relations are defined as octagonal constraints where variables can be also primed. Octagonal relations were studied in [BGI09] where it was shown that the transitive closure is Presburger definable. The core result of [BGI09] is that the canonical form of the m-th power of an octagonal relation R can be computed directly from the m-th power of a difference bounds relation that represents R. We present these results in Section 2.4.2.

2.4.1 Octagonal Constraints

Let $\mathbf{x} = \{x_1, x_2, ..., x_N\}$ be a set of variables ranging over $\mathbb{Z}$. The class of integer octagonal constraints is defined as follows:

Definition 2.17 *A formula $\phi(\mathbf{x})$ is an* octagonal constraint *if it is equivalent to a finite conjunction of terms of the form $x_i - x_j \leq a_{ij}$, $x_i + x_j \leq b_{ij}$ or $-x_i - x_j \leq c_{ij}$ where $a_{ij}, b_{ij}, c_{ij} \in \mathbb{Z}$, for all $1 \leq i, j \leq N$.*

We represent octagons as difference bounds constraints over the dual set of variables $\mathbf{y} = \{y_1, y_2, \ldots, y_{2N}\}$, with the convention that y_{2i-1} stands for x_i and y_{2i} for $-x_i$,

respectively. For example, the octagonal constraint $x_1+x_2 = 3$ is represented as $y_1-y_4 \leq 3 \wedge y_2-y_3 \leq -3$. In order to handle the $\mathbf{y}$ variables in the following, we define $\bar{i} = i-1$, if i is even, and $\bar{i} = i+1$ if i is odd. Obviously, we have $\bar{\bar{i}} = i$, for all $i \in \mathbb{Z},\ i \geq 0$. We denote by $\overline{\phi}(\mathbf{y})$ the difference bounds constraint over $\mathbf{y}$ that represents $\phi(\mathbf{x})$ and which is defined as follows:

Definition 2.18 *Given an octagonal constraint $\phi(\mathbf{x})$, $\mathbf{x} = \{x_1, \ldots, x_N\}$, its difference bounds representation $\overline{\phi}(\mathbf{y})$, $\mathbf{y} = \{y_1, \ldots, y_{2N}\}$ is a conjunction of the following difference bounds constraints where $1 \leq i \neq j \leq N$, $c \in \mathbb{Z}$.*

$$\begin{array}{lcl}
(x_i - x_j \leq c) \in AP(\phi) & \Leftrightarrow & (y_{2i-1} - y_{2j-1} \leq c), (y_{2j} - y_{2i} \leq c) \in AP(\overline{\phi}) \\
(-x_i + x_j \leq c) \in AP(\phi) & \Leftrightarrow & (y_{2j-1} - y_{2i-1} \leq c), (y_{2i} - y_{2j} \leq c) \in AP(\overline{\phi}) \\
(-x_i - x_j \leq c) \in AP(\phi) & \Leftrightarrow & (y_{2i} - y_{2j-1} \leq c), (y_{2j} - y_{2i-1} \leq c) \in AP(\overline{\phi}) \\
(x_i + x_j \leq c) \in AP(\phi) & \Leftrightarrow & (y_{2i-1} - y_{2j} \leq c), (y_{2j-1} - y_{2i} \leq c) \in AP(\overline{\phi}) \\
(2x_i \leq c) \in AP(\phi) & \Leftrightarrow & (y_{2i-1} - y_{2i} \leq c) \in AP(\overline{\phi}) \\
(-2x_i \leq c) \in AP(\phi) & \Leftrightarrow & (y_{2i} - y_{2i-1} \leq c) \in AP(\overline{\phi})
\end{array}$$

The following equivalence relates ϕ and $\overline{\phi}$:

$$\phi(\mathbf{x}) \Leftrightarrow (\exists y_2, y_4, \ldots, y_{2N} \,.\, \overline{\phi} \wedge \bigwedge_{i=1}^{N} y_{2i-1} = -y_{2i})[x_i/y_{2i-1}]_{i=1}^{N} \tag{2.1}$$

An octagonal constraint ϕ is equivalently represented by the DBM $M_{\overline{\phi}} \in \mathbb{Z}_\infty^{2N \times 2N}$, corresponding to $\overline{\phi}$. We sometimes write M_ϕ instead of $M_{\overline{\phi}}$. We say that a DBM $M \in \mathbb{Z}_\infty^{2N \times 2N}$ is *coherent* iff $M_{ij} = M_{\overline{j}\overline{i}}$ for all $1 \leq i, j \leq 2N$. This property is needed since e.g. an atomic proposition $x_i - x_j \leq a_{ij}$, $1 \leq i, j \leq N$, can be represented as both $y_{2i-1} - y_{2j-1} \leq a_{ij}$ and $y_{2j} - y_{2i} \leq a_{ij}$. Dually, a coherent DBM $M \in \mathbb{Z}_\infty^{2N \times 2N}$ corresponds to the octagonal constraint:

$$\Omega_M \Leftrightarrow \bigwedge_{1 \leq i,j \leq N} (x_i - x_j \leq M_{2i-1,2j-1} \wedge x_i + x_j \leq M_{2i-1,2j} \wedge -x_i - x_j \leq M_{2i,2j-1}) \tag{2.2}$$

A coherent DBM M is said to be *octagonal-consistent* if and only if Ω_M is consistent. Similar to the case of difference bounds constraints, for an octagonal constraint ϕ, we define $\|\phi\|$ as $\|\phi\| \stackrel{def}{=} \|\overline{\phi}\|$, where $\|\overline{\phi}\|$ is the maximal absolute value of all coefficients of $\overline{\phi}$ defined in Section 2.3.

Definition 2.19 *An octagonal-consistent coherent DBM $M \in \mathbb{Z}_\infty^{2N \times 2N}$ is said to be* tightly closed *if and only if the following hold, for all $1 \leq i, j, k \leq 2N$:*

1. $M_{ii} = 0$
2. $M_{i\bar{i}}$ *is even*
3. $M_{ij} \leq M_{ik} + M_{kj}$
4. $M_{ij} \leq \lfloor \frac{M_{i\bar{i}}}{2} \rfloor + \lfloor \frac{M_{\bar{j}j}}{2} \rfloor$

Given an octagonal-consistent coherent DBM $M \in \mathbb{Z}^{2N} \times \mathbb{Z}^{2N}$, we denote the (unique) tightly closed DBM by M^t . The following theorem from [BHZ08] provides an effective way of testing octagonal-consistency and computing the tight closure of a coherent DBM. Moreover, it shows that the tight closure of a given DBM is unique and can also be computed in time $\mathcal{O}(N^3)$.

Theorem 2.20 *[BHZ08] Let $M \in \mathbb{Z}_\infty^{2N \times 2N}$ be a coherent DBM. Then M is octagonal-consistent if and only if M is consistent and $\lfloor \frac{M^*_{i\bar{i}}}{2} \rfloor + \lfloor \frac{M^*_{\bar{i}i}}{2} \rfloor \geq 0$, for all $1 \leq i \leq 2N$. Moreover, if M is octagonal-consistent, the tight closure of M is the DBM $M^t \in \mathbb{Z}_\infty^{2N \times 2N}$ defined as:*

$$M^t_{ij} = \min \left\{ M^*_{ij}, \left\lfloor \frac{M^*_{i\bar{i}}}{2} \right\rfloor + \left\lfloor \frac{M^*_{\bar{j}j}}{2} \right\rfloor \right\}$$

for all $1 \leq i, j \leq 2N$ where $M^* \in \mathbb{Z}_\infty^{2N \times 2N}$ is the closure of M.

The tight closure of DBMs is needed for checking equivalence and entailment between octagonal constraints.

Proposition 2.21 ([Min06]) *Let ϕ_1 and ϕ_2 be octagonal-consistent octagonal constraints. Then,*

- *$\phi_1 \Leftrightarrow \phi_2$ if and only if $M^t_{\phi_1} = M^t_{\phi_2}$,*
- *$\phi_1 \Rightarrow \phi_2$ if and only if $M^t_{\phi_1} \leq M^t_{\phi_2}$.*

It has been shown in [BGI09] that octagonal constraints are closed under existential quantification.

Proposition 2.22 *Let $\phi(\mathbf{x})$, $\mathbf{x} = \{x_1, \ldots, x_N\}$, be an octagonal-consistent octagonal constraint. Further, let $1 \leq k \leq 2N$ and M' be the restriction of M^t_ϕ to $\mathbf{y} \setminus \{y_{2k-1}, y_{2k}\}$. Then, M' is tightly closed, and $\Omega(M') \Leftrightarrow \exists x_k.\phi(\mathbf{x})$.*

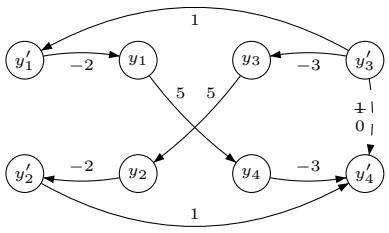
Proof: See [BGI09], Theorem 2. □

2.4.2 Octagonal Relations and Their Powers

Definition 2.23 *Let $\mathbf{x} = \{x_1, \ldots, x_N\}$ be a set of variables. A relation $R \in \mathbb{Z}^N \times \mathbb{Z}^N$ is an* octagonal relation *if it can be defined by an octagonal constraint $R(\mathbf{x}, \mathbf{x}')$.*

The class of relations defined by octagonal constraints is denoted by $\mathcal{R}_{oct}$ in the following.

Example 2.24 *Consider the octagonal relation $R(x_1, x_2, x'_1, x'_2) \Leftrightarrow x_1 + x_2 \leq 5 \wedge x'_1 - x_1 \leq -2 \wedge x'_2 - x_2 \leq -3 \wedge x'_2 - x'_1 \leq 1$. Its difference bounds representation is $\overline{R}(\mathbf{y}, \mathbf{y}') \Leftrightarrow y_1 - y_4 \leq 5 \wedge y_3 - y_2 \leq 5 \wedge y'_1 - y_1 \leq -2 \wedge y_2 - y'_2 \leq -2 \wedge y'_3 - y_3 \leq -3 \wedge y_4 - y'_4 \leq -3 \wedge y'_3 - y'_1 \leq 1 \wedge y'_2 - y'_4 \leq 1$, where $\mathbf{y} = \{y_1, \ldots, y_4\}$. Figure 2.5a shows the graph*



$\mathcal{G}_R$

$$M_R^t = \begin{array}{c} \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{array} \begin{pmatrix} y_1 & y_2 & y_3 & y_4 & y'_1 & y'_2 & y'_3 & y'_4 \\ 0 & \infty & \infty & 5 & \infty & \infty & \infty & 2 \\ \infty & 0 & \infty & \infty & \infty & -2 & \infty & -1 \\ \infty & 5 & 0 & \infty & \infty & 3 & \infty & 4 \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty & -3 \\ -2 & \infty & \infty & 3 & 0 & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty & 1 \\ -1 & 2 & -3 & 4 & 1 & 0 & 0 & 0 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

Figure 2.5: Graph and matrix representation of a relation.

representation $\mathcal{G}_R$. Note that the implicit constraint $y'_3 - y'_4 \leq 1$ (represented by a dashed edge in Figure 2.5a) is not tight. The tightening step replaces the bound 1 (crossed in Figure 2.5a) with 0. Figure 2.5b shows the tightly closed DBM representation of R, denoted M_R^t. □

A consequence of Proposition 2.22 is that $\mathcal{R}_{oct}$ is closed under composition.

Proposition 2.25 *$\mathcal{R}_{oct}$ is closed under intersection and composition.*

Proof: Let $R_1(\mathbf{x}, \mathbf{x}')$, $R_2(\mathbf{x}, \mathbf{x}')$ be octagonal constraints defining octagonal relations. By Definition 2.17, $R_1(\mathbf{x}, \mathbf{x}') \wedge R_2(\mathbf{x}, \mathbf{x}')$ is an octagonal constraints to. The composition of relations $R_1 \circ R_2$ can be defined as $\exists \mathbf{x}''$. $(R_1(\mathbf{x}, \mathbf{x}'') \wedge R_2(\mathbf{x}'', \mathbf{x}'))$ which is again an octagonal constraint, by Definition 2.17 and Proposition 2.22. □

We rely in the following chapters on the main result of [BGI09], which establishes the following relation between $M^t_{\overline{R}^m}$ (the tightly closed octagonal DBM corresponding to the m-th iteration of R) and $M^*_{\overline{R}^m}$ (the closed DBM corresponding to the m-th iteration of the difference bounds relation $\overline{R}$), for all $m \geq 0$:

Theorem 2.26 *[BGI09] Let $R(\mathbf{x}, \mathbf{x}')$, $\mathbf{x} = \{x_1, \ldots, x_N\}$, be a $*$-consistent octagonal relation. Then, $M^t_{\overline{R}^m} = M^t_{\overline{R}^m}$ for all $m \geq 0$. Consequently,*

$$(M^t_{\overline{R}^m})_{ij} = \min \left\{ (M^*_{\overline{R}^m})_{ij}, \left\lfloor \frac{(M^*_{\overline{R}^m})_{i\bar{i}}}{2} \right\rfloor + \left\lfloor \frac{(M^*_{\overline{R}^m})_{\bar{j}j}}{2} \right\rfloor \right\}$$

for all $1 \leq i, j \leq 4N$.

This relation is in fact a generalization of the tight closure definition from Theorem 2.20, from $m = 1$ to any $m \geq 0$.

2.5 Finite Monoid Affine Relations

Sections 2.3 and 2.4 presented two classes of non-deterministic relations. In this section, we present linear affine relations which are a general model of deterministic transition relations. Linear affine relations are relations of the form $\mathbf{x}' = A \times \mathbf{x} + \mathbf{b} \wedge \phi(\mathbf{x})$, where $\mathbf{x}' = A \times \mathbf{x} + \mathbf{b}$ is an affine transformation and $\phi(\mathbf{x})$ is a Presburger guard. We present two subclasses of linear affine relations, called *finite monoid affine relations* and *polynomially bounded affine relations*.

The class of finite monoid affine relations was the first class of integer relations for which the transitive closure has been shown to be Presburger definable by Boigelot [Boi99]. Informally, an affine relation is a finite monoid relation if the set of powers of its transformation matrix is finite. Originally, Boigelot characterized this class by two decidable conditions in [Boi99] (we report on these conditions in Lemma 2.28). Later, Finkel and Leroux noticed in [FL02] that Boigelot's conditions correspond to the finite monoid property, which is also known to be decidable [MS77].

The second subclass of polynomially bounded relations is defined by dropping one of the Boigelot's conditions and by requiring that the guard of a relation is linear. We study this subclass in Chapter 6 which presents a method for computation of termination preconditions for this class.

Definition 2.27 *Let $\mathbf{x} = \langle x_1, \ldots, x_N \rangle$ be a vector of variables ranging over $\mathbb{Z}$. A relation $R \in \mathbb{Z}^N \times \mathbb{Z}^N$ is an* affine relation *if it can be defined by a formula $R(\mathbf{x}, \mathbf{x}')$ of the form*

$$R(\mathbf{x}, \mathbf{x}') \;\Leftrightarrow\; \mathbf{x}' = A \times \mathbf{x} + \mathbf{b} \;\wedge\; \phi(\mathbf{x}) \tag{2.3}$$

where $A \in \mathbb{Z}^{N \times N}$, $\mathbf{b} \in \mathbb{Z}^N$, and ϕ is a Presburger formula over unprimed variables only, called the guard. *The formula $\mathbf{x}' = A \times \mathbf{x} + \mathbf{b}$, defining a linear transformation, is called the* update.

The affine transformation is said to have the *finite monoid property* [Boi99, FL02] if the monoid of powers of A, denoted as $\langle \mathcal{M}_A, \times \rangle$, where $\mathcal{M}_A = \{A^i \mid i \geq 0\}$, is finite. In this case, we also say that A has the finite monoid property. Here $A^0 = I_N$ and $A^i = A \times A^{i-1}$, for $i > 0$. Intuitively, the finite monoid property is equivalent to the fact that A has finitely many powers (considering the standard integer multiplication). A linear affine relation has the finite monoid property if and only if the matrix A defining the update has the finite monoid property.

It has been shown in [FL02] that finite monoid property can be equivalently characterized by a pair of conditions. Before presenting this characterization, we recall several notions of linear algebra.

If $A \in \mathbb{Z}^{n \times n}$ is a square matrix, and $\mathbf{v} \in \mathbb{Z}^n$ is a column vector of integer constants, then any complex number $\lambda \in \mathbb{C}$ such that $A\mathbf{v} = \lambda\mathbf{v}$, for some complex vector $\mathbf{v} \in \mathbb{C}^n$, is called an *eigenvalue* of A. The vector $\mathbf{v}$ in this case is called an *eigenvector* of A. It is known that the eigenvalues of A are the roots of the *characteristic polynomial* $P_A(x) = \det(A - xI_n) = 0$, which is an effectively computable univariate polynomial. The *minimal polynomial* of A is the polynomial μ_A of lowest degree such that $\mu_A(A) = 0$. By

the Cayley-Hamilton Theorem, the minimal polynomial always divides the characteristic polynomial, i.e. the roots of the former are root of the latter.

If $\lambda_1, \ldots, \lambda_m$ are the eigenvalues of A, then $\lambda_1^p, \ldots, \lambda_m^p$ are the eigenvalues of A^p, for all integers $p > 0$. A matrix is said to be *diagonalizable* if and only if there exists a non-singular matrix $U \in \mathbb{C}^{N \times N}$ and a diagonal matrix with the eigenvalues $\lambda_1, \ldots, \lambda_m$ occurring on the main diagonal, such that $A = U \times D \times U^{-1}$. This is the case if and only if μ_A has only roots of multiplicity one (see e.g. Thm 8.47 in [Boi99]).

A complex number r is said to be a *root of the unity* if $r^d = 1$ for some integer $d > 0$. The *cyclotomic polynomial* $F_d(x)$ is the product of all monomials $(x - \omega)$, where $\omega^d = 1$, and $\omega^e \neq 1$, for all $0 < e < d$. It is known that a polynomial has only roots which are roots of unity if and only if it is a product of cyclotomic polynomials.

With these notions, the finite monoid property is defined by the following equivalent conditions.

Theorem 2.28 *[Thm 8.42 and 8.44 in [Boi99] and Prop 2 in [FL02]] a relation $R \equiv A\mathbf{x} + \mathbf{b}$, where $A \in \mathbb{Z}^{N \times N}$ and $\mathbf{b} \in \mathbb{Z}^N$ has the finite monoid condition if and only if there exists $p > 0$ such that the following hold:*

1. *every eigenvalue of A^p belongs to the set $\{0, 1\}$,*
2. *the minimal polynomial $\mu_{A^p}(x)$ of A^p belongs to the set $\{0, x, x-1, x(x-1)\}$ (or, equivalently, A^p is diagonalizable).*

Both conditions in Theorem are decidable [Boi99, MS77].

In Chapter 6, we study another subclass of affine relations with linear guards and transformation matrix whose eigenvalues are either zero or roots of the unity.

Definition 2.29 *If $\mathbf{x} = \langle x_1, \ldots, x_N \rangle$ is a vector of variables ranging over $\mathbb{Z}$, a* polynomially bounded affine relation *is a relation of the form*

$$R(\mathbf{x}, \mathbf{x}') \;\Leftrightarrow\; \mathbf{x}' = A \times \mathbf{x} + \mathbf{b} \;\wedge\; C\mathbf{x} \geq \mathbf{d} \tag{2.4}$$

where $A \in \mathbb{Z}^{n \times n}$, $C \in \mathbb{Z}^{p \times n}$ are matrices, and $\mathbf{b} \in \mathbb{Z}^n$, $\mathbf{d} \in \mathbb{Z}^p$ are column vectors of integer constants and moreover, all eigenvalues of A are either zero or roots of the unity.

Note that if A is a finite monoid matrix, then all eigenvalues of A are either zero or roots of the unity. Thus, the condition on A is weaker for polynomially bounded affine relations. However, since the guard of finite monoid relations is more general (Presburger), the two classes are incomparable.

3 Computing Transitive Closures of Periodic Relations

In this chapter, we present a general framework for computing *closed forms* and *transitive closures* of certain relations, called *periodic*. The closed form of a relation $R(\mathbf{x}, \mathbf{x}')$ is a relation $\widehat{R}(k, \mathbf{x}, \mathbf{x}')$ where substituting the parameter k with an integer m gives a relation equivalent to R^m for each $m \geq 0$. Once the closed form is computed, the transitive closure of R can be defined as $\exists k \geq 1 \,.\, \widehat{R}(k, \mathbf{x}, \mathbf{x}')$.

We define a notion of periodicity on classes of relations that can be naturally represented as matrices. In general, an infinite sequence of integers is said to be *periodic* if the elements of the sequence situated at equal distance one from another differ by the same quantity (while admitting some non-periodic initial prefix of finite length in the sequence). This definition is generalized to matrices of integers, entry-wise. Assuming that each finite power R^k of a relation R is represented by a matrix M_k, R is said to be periodic if the infinite sequence $\{M_k\}_{k=0}^{\infty}$ of matrix representations of powers of R is periodic. Periodicity guarantees that this sequence has an infinite subsequence which can be captured by an existentially quantified formula. This formula consequently represents infinitely many powers of the relation. Then, the remaining powers can be computed by composing the existentially quantified formula with certain powers of the relation.

For instance, consider the relation R defined as $x' = y + 1 \wedge y' = x$. This relation is periodic, and its odd powers $R^1, R^3, R^5, \ldots$ can be represented by the formula $\exists \ell \geq 0 \,.\, (x' = y + \ell + 1 \wedge y' = x + \ell)$. Even powers $R^2, R^4, R^6, \ldots$ can then be represented by composing this formula with R.

We show that the closed form of a periodic relation can be defined, once the *period* and the *prefix* of the relation are known. We present a generic algorithm that finds a period and a prefix of periodic relations and computes their closed form and transitive closure.

Roadmap. We define the basic notions in Section 3.1. Next, Section 3.2 presents a theoretical framework in which the closed form of periodic relations can be computed. Finally, we present a generic algorithm that finds the period and the prefix of periodic relations and computes their transitive closure in Section 3.3.

3.1 Periodic Sequences

We first define the notion of periodic sequences.

Definition 3.1 *Given an infinite sequence $\{s_k\}_{k=0}^{\infty} \in \mathbb{Z}_\infty$, we say that it is* periodic *if and only if there exist integers $b \geq 0$, $c > 0$ and $\lambda_0, \ldots, \lambda_{c-1} \in \mathbb{Z}_\infty$ such that*

$s_{b+(k+1)c+i} = \lambda_i + s_{b+kc+i}$, *for all* $k \geq 0$ *and* $i = 0, 1, \ldots, c-1$ *The smallest values* $b \in \mathbb{N}$, $c \in \mathbb{N}_+$ *for which the above holds are called the* prefix *and the* period *of* $\{s_k\}_{k=0}^{\infty}$. *The values* $\lambda_0, \lambda_1, \ldots, \lambda_{c-1} \in \mathbb{Z}_\infty$ *are called the* rates *of* $\{s_k\}_{k=0}^{\infty}$.

Example 3.2 *The sequence* $\{\sigma_k\}_{k=0}^{\infty}$ *where* $\sigma_0 = \sigma_1 = 10$, $\sigma_k = 5\ell + 3$ *for each* $k = 2\ell$, $\ell \geq 1$, *and* $\sigma_k = 3\ell + 1$ *for each* $k = 2\ell + 1$, $\ell \geq 1$, *is periodic with prefix* $b = 2$, *period* $c = 2$ *and rates* $\lambda_0 = 5$, $\lambda_1 = 3$. *The sequence* $\{\tau_k\}_{k=0}^{\infty}$ *where* $\sigma_k = 7\ell + 1$ *for each* $k = 3\ell$, $\ell \geq 0$, $\sigma_k = \ell^2$ *for each* $k = 3\ell + 1$, $\ell \geq 0$, *and* $\sigma_k = \ell^3$ *for each* $k = 3\ell + 2$, $\ell \geq 0$ *is not periodic.* □

The notion of periodic sequences extends to sequences of matrices:

Definition 3.3 *A sequence of matrices* $\{A_k\}_{k=0}^{\infty} \in \mathbb{Z}_\infty^{m \times m}$ *is said to be periodic if, for all* $1 \leq i, j \leq m$, *the sequence* $\{(A_k)_{ij}\}_{k=0}^{\infty}$ *is periodic.*

The following lemma gives an alternative characterization of periodic sequences of matrices:

Lemma 3.4 *An infinite sequence of matrices* $\{A_k\}_{k=1}^{\infty} \in \mathbb{Z}_\infty^{m \times m}$ *is periodic if and only if there exist integers* $b \geq 0$, $c > 0$ *and* $\Lambda_0, \ldots, \Lambda_{c-1} \in \mathbb{Z}_\infty^{m \times m}$ *such that* $A_{b+(k+1)c+i} = \Lambda_i + A_{b+kc+i}$, *for all* $k \geq 0$ *and* $i = 0, 1, \ldots, c-1$

Proof: According to the definition, $\{A_k\}_{k=1}^{\infty}$ is periodic if and only if, for each $1 \leq i, j \leq m$ there exist $b_{ij} \geq 0$, $c_{ij} > 0$ and $\lambda_l^{ij} \in \mathbb{Z}_\infty$ such that $\left(A_{b_{ij}+(k+1)c_{ij}+l}\right)_{ij} = \lambda_l^{ij} + \left(A_{b_{ij}+kc_{ij}+l}\right)_{ij}$ for all $k \geq 0$, $l = 0, 1, \ldots, c_{ij} - 1$. Let c be the least common multiple of all c_{ij}, b be the maximum of all b_{ij} and let Λ_t, $t = 0, 1, \ldots, c-1$ be the matrix defined as:

$$(\Lambda_t)_{ij} = \left(\lambda^{ij}_{(b-b_{ij}+t) \mod c_{ij}}\right) \cdot \frac{c}{c_{ij}}$$

The condition $A_{b+(k+1)c+i} = \Lambda_i + A_{b+kc+i}$ is verified for all $k \geq 0$ and $i = 0, 1, \ldots, c-1$, with the above definitions. □

3.2 Periodic Relations

Let $\mathbf{x} = \{x_1, \ldots, x_N\}$ be a set of variables. In this section, we consider that $\mathcal{R}$ is a class of first-order arithmetic formulae with free variables in $\mathbf{x} \cup \mathbf{x}'$. These formulae denote integer relations $R \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$. We recall the definition of a $*$-*consistent* relation and next define the notion of a periodic relation.

Definition 3.5 *A relation* R *is* $*$*-consistent if and only if* R^n *is consistent for all* $n \geq 0$.

Definition 3.6 *A relation* $R \in \mathcal{R}$ *is said to be* periodic *if and only if, either (1) it is not* $*$*-consistent, or (2) it is* $*$*-consistent and there exist two functions:*

- $\sigma : \mathcal{R} \to (\mathbb{Z}_\infty^{m\times m})_\perp$ *mapping each* consistent *relation from* $\mathcal{R}$ *into a matrix, and each inconsistent relation into* $\perp$
- $\rho : \mathbb{Z}_\infty^{m\times m} \to \mathcal{R}$ *mapping each matrix into a relation from* $\mathcal{R}$*, such that* $\rho(\sigma(R)) \Leftrightarrow R$*, for each consistent relation* $R \in \mathcal{R}$

such that the infinite sequence of matrices $\{\sigma(R^i)\}_{i=0}^\infty \in \mathbb{Z}_\infty^{m\times m}$ *is periodic.*

If each relation $R \in \mathcal{R}$ is periodic, then the class of relations $\mathcal{R}$ is called periodic as well. The following lemma gives an alternative characterization of $*$-consistent periodic relations.

Lemma 3.7 *A* $*$*-consistent relation* R *is periodic if and only if there exist integers* $b \geq 0$*,* $c > 0$*,* $m > 0$ *and a matrices* $\Lambda_0, \ldots, \Lambda_{c-1} \in \mathbb{Z}_\infty^{m\times m}$ *such that*

$$R^{nc+b+i} \Leftrightarrow \rho(n \cdot \Lambda_i + \sigma(R^{b+i}))$$

for all $n \geq 0$ *and for all* $0 \leq i < c$*.*

Proof: By induction on $n \geq 0$, we prove that $\sigma(R^{nc+b+i}) = n \cdot \Lambda + \sigma(R^{b+i})$, for all $n \geq 0$ and for all $0 \leq i < c$. The base case trivially holds. For the induction step, observe that

$$\sigma(R^{b+i+(n+1)c}) = \Lambda_i + \sigma(R^{b+i+nc}) = \Lambda + n \cdot \Lambda + \sigma(R^{b+i}) = (n+1) \cdot \Lambda_i + \sigma(R^b).$$

The first equality is by Lemma 3.4, the second is by the induction hypothesis. □

Next, we define prefix, period and rates of periodic relations.

Definition 3.8 *If* R *is a* $*$*-consistent and periodic relation, we call* prefix, period *and* rates *of* R *the minimal integers* $b \geq 0$*,* $c > 0$ *and matrices* $\Lambda_0 \ldots, \Lambda_{c-1} \in \mathbb{Z}_\infty^{m\times m}$ *satisfying the condition of Lemma 3.7. If* R *is not* $*$*-consistent, we define its prefix and period as* $b = \inf\{n \geq 0 \mid R^n \text{ is inconsistent } \}$ *and* $c = 1$*.*

If k is a variable not in $\mathbf{x} \cup \mathbf{x}'$, let $\mathcal{R}[k]$ be the class of first-order arithmetic formulae with free variables from the set $\mathbf{x} \cup \mathbf{x}' \cup \{k\}$. The variable k is called a parameter, and these formulae are called *parametric relations*, in the following. The following definition emphasizes the role of parametric relations.

Definition 3.9 *Let* $R(\mathbf{x}, \mathbf{x}') \in \mathcal{R}$ *be a relation. The* closed form *of* R *is the formula* $\widehat{R}(k, \mathbf{x}, \mathbf{x}') \in \mathcal{R}[k]$ *such that* $\widehat{R}(k, \mathbf{x}, \mathbf{x}')[n/k] \Leftrightarrow R^n(\mathbf{x}, \mathbf{x}')$*, for all* $n \geq 0$*.*

It follows immediately from the above definition that the closed form of a relation is unique, up to equivalence. Defining the transitive closure of a relation is closely related to defining its closed form, since $R^+(\mathbf{x}, \mathbf{x}') \Leftrightarrow \exists k > 0 \ . \ \widehat{R}(k, \mathbf{x}, \mathbf{x}')$, for all $R \in \mathcal{R}$.

The algorithm presented in this section computes the transitive closure of a periodic relation R by computing the closed form of a subsequence $\{\sigma(R^{b+nc})\}_{n\geq 0}$ for some $b \geq 0, c > 0$ (not necessarily the prefix and the period of R).

Definition 3.10 *Given a relation $R(\mathbf{x},\mathbf{x}')$ and integers $b \geq 0$, $c > 0$, the* closed form *of the the infinite sequence $\{R^{b+nc}\}_{n\geq 0}$ is a formula $\widehat{R}_{b,c}(\ell,\mathbf{x},\mathbf{x}')$ such that $\widehat{R}_{b,c}(\ell,\mathbf{x},\mathbf{x}')[n/\ell] \Leftrightarrow R^{b+nc}$ for all $n \geq 0$.*

Once the closed form $\widehat{R}_{b,c}(\ell,\mathbf{x},\mathbf{x}')$ is computed for some $b \geq 0, c > 0$, both the transitive closure and the closed form of R can be defined as shown by the following lemma.

Lemma 3.11 *Let R be a relation, $b > 0$, $c > 0$ be arbitrary integers and $\widehat{R}_{b,c}(\ell,\mathbf{x},\mathbf{x}')$ be the closed form of the infinite sequence $\{R^{b+nc}\}_{n\geq 0}$. Then, the transitive closure and the closed form of R can be defined as:*

$$R^+ \Leftrightarrow \big(\bigvee_{i=1}^{b-1} R^i\big) \ \vee \ \exists \ell \geq 0 \ . \ \widehat{R}_{b,c}(\ell,\mathbf{x},\mathbf{x}') \circ \big(\bigvee_{j=0}^{c-1} R^j\big)$$

$$\widehat{R}(k,\mathbf{x},\mathbf{x}') \Leftrightarrow \big(\bigvee_{i=1}^{b-1} (k=i) \wedge R^i\big) \ \vee \ \exists \ell \geq 0 \ . \ \widehat{R}_{b,c}(\ell,\mathbf{x},\mathbf{x}') \circ \big(\bigvee_{j=0}^{c-1} (k=b+\ell c+j) \wedge R^j\big)$$

Proof: Let $\widehat{R}_{b,c}(\ell,\mathbf{x},\mathbf{x}')$ be the closed form of $\{R^{b+nc}\}_{n\geq 0}$ for some integers $b > 0, c > 0$. Observe that

$$\begin{array}{rcl} R^+ & \Leftrightarrow & \bigvee_{i=1}^{\infty} R^i \\ & \Leftrightarrow & \big(\bigvee_{i=1}^{b-1} R^i\big) \ \vee \ \big(\bigvee_{n=0}^{\infty} R^{nc+b}\big) \circ \big(\bigvee_{j=0}^{c-1} R^j\big) \\ & \Leftrightarrow & \big(\bigvee_{i=1}^{b-1} R^i\big) \ \vee \ \exists \ell \geq 0 \ . \ \widehat{R}_{b,c}(\ell,\mathbf{x},\mathbf{x}') \circ \big(\bigvee_{j=0}^{c-1} R^j\big) \end{array}$$

The last equivalence above follows from Definition 3.10. In a similar way, we infer that the closed form of a relation R can be defined as

$$\widehat{R}(k,\mathbf{x},\mathbf{x}') \Leftrightarrow \big(\bigvee_{i=1}^{b-1} (k=i) \wedge R^i\big) \ \vee \ \exists \ell \geq 0 \ . \ \widehat{R}_{b,c}(\ell,\mathbf{x},\mathbf{x}') \circ \big(\bigvee_{j=0}^{c-1} (k=b+\ell c+j) \wedge R^j\big),$$

which completes the proof. □

Next, consider a function $\pi : \mathbb{Z}[k]_{\infty}^{m\times m} \rightarrow \mathcal{R}[k]$ mapping matrices of linear terms of the form $\alpha \cdot k + \beta$, with integer coefficients, into parametric relations $R(k,\mathbf{x},\mathbf{x}')$, such that

$$\pi(M)[n/k] \Leftrightarrow \rho(M[n]), \text{for all } n \in \mathbb{Z}$$

for all $M \in \mathbb{Z}[k]_{\infty}^{m\times m}$. In other words, π is the *parametric counterpart* of the ρ function from Definition 3.6. Concrete examples of parametric matrix-relations mappings will be given in Sections 4.2, 4.3, and 4.4.

We next show that the transitive closure and the closed form of a periodic relation R with prefix b, period c, and rates $\Lambda_0, \ldots, \Lambda_{c-1}$ can be defined in first order arithmetic. If R is not $*$-consistent, then clearly $R^+ \Leftrightarrow \bigvee_{i=1}^{b-1} R^i$ and $\widehat{R}(k,\mathbf{x},\mathbf{x}') \Leftrightarrow \bigvee_{i=1}^{b-1}(k = i \wedge R^i)$. If R is $*$-consistent, then by Lemma 3.7, $R^{nc+b} \Leftrightarrow \rho(n \cdot \Lambda_0 + \sigma(R^b))$. Then, we apply the following proposition which shows that the closed form of the sequence $\{R^{b+nc}\}_{n\geq 0}$ can be defined as $\widehat{R}_{b,c}(\ell,\mathbf{x},\mathbf{x}') \Leftrightarrow \pi(\ell \cdot \Lambda_0 + \sigma(R^b))$.

Proposition 3.12 *Let R be a relation, $b > 0, c > 0$ be arbitrary integers, and $\Lambda \in \mathbb{Z}_\infty^{m\times m}$ be a matrix such that, for all $n \geq 0$:*

1. $\rho(n \cdot \Lambda + \sigma(R^b)) \not\Leftrightarrow \bot$, *and*
2. $R^{nc+b} \Leftrightarrow \rho(n \cdot \Lambda_0 + \sigma(R^b))$.

Then, R is $*$-consistent and $\widehat{R}_{b,c}(k, \mathbf{x}, \mathbf{x}') \Leftrightarrow \pi(k \cdot \Lambda + \sigma(R^b))$.

Proof: Clearly, if both conditions hold, it follows that R is $*$-consistent. The mapping $\pi : \mathbb{Z}[k]_\infty^{m\times m} \to \mathcal{R}[k]$ satisfies the following equivalence for all $M \in \mathbb{Z}[k]_\infty^{m\times m}$ and for all $n \in \mathbb{Z}$

$$\pi(M)[n/k] \Leftrightarrow \rho(M[n])$$

Thus, letting $M = k \cdot \Lambda + \sigma(R^b)$, it follows that if the two conditions hold, then

$$\pi(M)[n/k] \Leftrightarrow \rho(M[n]) \Leftrightarrow R^{b+nc}.$$

Hence, $\pi(M) = \pi(k \cdot \Lambda + \sigma(R^b))$ is the closed form of of the sequence $\{R^{b+nc}\}_{n\geq 0}$, by Definition 3.10. □

Having computed $\widehat{R}_{b,c}(\ell, \mathbf{x}, \mathbf{x}')$, we can finally apply Lemma 3.11 to define the transitive closure and the closed form of R.

3.3 Transitive Closure Algorithm

The result of this section is a generic algorithm that computes the transitive closure of a given periodic relation (Algorithm 1). The algorithm needs to be instantiated for a specific class $\mathcal{R}$ of periodic relations by providing the corresponding mappings σ, ρ (Definition 3.6) and π (the parametric counterpart of ρ) as discussed in the previous. This algorithm can be easily adapted to compute the closed form of a relation, instead of its transitive closure, as we show in the end of this section. Next, in Chapter 4 we show how this algorithm can be used with three classes of relations: difference bounds, octagons, and finite monoid affine transformations.

The main idea of the algorithm is to discover integers $b \geq 0$ and $c > 0$ such that the sequence $\{\sigma(R^{b+nc})\}_{n\geq 0}$ is periodic. Provided that relation R is periodic, the enumeration on lines 2,4,16 is guaranteed to eventually find a pair (b, c) for which the algorithm terminates at line 7 or 12, as we later prove in Theorem 3.16. For each prefix-period candidate (b, c), we consider the first three powers supposed to be equidistant, namely R^b, R^{b+c} and R^{b+2c}, and we check that all three are consistent (lines 5-6). If at least one is inconsistent, the relation is not $*$-consistent, and the transitive closure is the disjunction of all powers up to the first one which is inconsistent (line 7). Otherwise, if R^b, R^{b+c} and R^{b+2c} are consistent, the algorithm attempts to compute the first rate of the sequence (line 8), by comparing the matrices $\sigma(R^b)$, $\sigma(R^{b+c})$ and $\sigma(R^{b+2c})$. If the distance Λ between $\sigma(R^{b+c})$ and $\sigma(R^b)$ equals the one between $\sigma(R^{b+2c})$ and $\sigma(R^{b+c})$, then Λ is a potential candidate for the rate of the sequence $\{\sigma(R^b + nc)\}_{n\geq 0}$.

Algorithm 1 Transitive Closures of Periodic Relations

input a periodic relation R
output The transitive closure of R

1: **function** TransitiveClosure(R)
2: **let** $P \leftarrow R$ **and** $b \leftarrow 1$ **and** $b_{jump} = 1$
3: **while** *true* **do**
4: **for all** $c = 1, 2, \ldots, b$ **do**
5: **for all** $\ell = 0, 1, 2$ **do**
6: **if** $R^{b+\ell c} \Leftrightarrow \bot$ **then**
7: **return** $R^+ \Leftrightarrow P \vee \left(\bigvee_{i=b+1}^{b+\ell c-1} R^i\right)$
8: **if** $\exists \Lambda \,.\, \sigma(R^b) + \Lambda = \sigma(R^{b+c}) \wedge \sigma(R^{b+c}) + \Lambda = \sigma(R^{b+2c})$ **then**
9: $K \leftarrow$ maxConsistent(R, b, Λ)
10: $L \leftarrow$ maxPeriodic(R, b, Λ, c, K)
11: **if** $L = \infty$ **then**
12: **return** $P \vee \exists k \geq 0 \,.\, \pi(k \cdot \Lambda + \sigma(R^b)) \circ \left(\bigvee_{j=0}^{c-1} R^j\right)$
13: $b_{jump} \leftarrow \max\{b_{jump}, b + c \cdot (L+1)\}$
14: $b_{next} \leftarrow \max\{b+1, b_{jump}\}$
15: $P \leftarrow P \vee \bigvee_{i=b}^{b_{next}-1} R^i$
16: $b \leftarrow b_{next}$
17: **function** maxConsistent(R,b,Λ)
18: **return** $\sup\{n \in \mathbb{N} \mid \rho(n \cdot \Lambda + \sigma(R^b)) \not\Leftrightarrow \bot\}$
19: **function** maxPeriodic(R, b, Λ, c, K)
20: **return** $\sup\{n \leq K \mid \forall 0 \leq \ell < n \,.\, \rho(\ell \cdot \Lambda + \sigma(R^b)) \circ R^c \Leftrightarrow \rho((\ell+1) \cdot \Lambda + \sigma(R^b))\}$

Next, we validate the choices of b, c and Λ by checking on lines 9-11 whether (i) the sequence of relations $\{\rho(n \cdot \Lambda + \sigma(R^b))\}_{n=0}^{\infty}$ is $*$-consistent, and (ii) that it follows the pattern of a periodic sequence. Since, by the definition of MaxPeriodic, either $L = K = \infty$, or else $L < K$, the following Lemma 3.13 ensures that if the test on line 11 succeeds for the chosen b, c and Λ, then $\pi(k \cdot \Lambda + \sigma(R^b))$ is the closed form of the sequence $\{\sigma(R^{b+nc})\}_{n \geq 0}$ and consequently, the transitive closure can be defined using Lemma 3.11 (line 12).

Lemma 3.13 *Let R be a periodic relation, let $b > 0, c > 0$ be integers such that $R^b \not\Leftrightarrow \bot$, and $\Lambda \in \mathbb{Z}_{\infty}^{m \times m}$ be a matrix such that, for all $n \geq 0$:*

1. *$\rho(n \cdot \Lambda + \sigma(R^b)) \not\Leftrightarrow \bot$, and*
2. *$\rho((n+1) \cdot \Lambda + \sigma(R^b)) \Leftrightarrow \rho(n \cdot \Lambda + \sigma(R^b)) \circ R^c$.*

Then, R is $*$-consistent and $\widehat{R}_{b,c}(k, \mathbf{x}, \mathbf{x}') \Leftrightarrow \pi(k \cdot \Lambda + \sigma(R^b))$.

Proof: It is sufficient to prove that

$$R^{b+nc} \Leftrightarrow \rho(n \cdot \Lambda + \sigma(R^b)) \text{ for all } n \geq 0.$$

For, if the above is true, the statement of this lemma follows from Proposition 3.12. The base case $n = 0$ follows from Definition 3.6 and the fact that R^b is consistent. The induction step is as follows:

$$\begin{aligned} R^{(n+1)c+b} &\Leftrightarrow R^{nc+b} \circ R^c \\ &\Leftrightarrow \rho(n \cdot \Lambda + \sigma(R^b)) \circ R^c \quad \text{(by the induction hypothesis)} \\ &\Leftrightarrow \rho((n+1) \cdot \Lambda + \sigma(R^b)) \end{aligned}$$

□

Suppose now that the test on line 11 fails, i.e. the sequence of relations $\{\rho(n \cdot \Lambda + \sigma(R^b))\}_{n=0}^{\infty}$ is either not $*$-consistent or it is not periodic. In this case we could start looking for a new prefix, period and rate, by incrementing b, setting c to one, and continuing to look for another candidate rate Λ, which satisfies the test at line 8. This could be achieved by skipping line 13. However, for relations with very long prefixes, this would be quite ineffective, as shown by the following example.

Example 3.14 *Consider, for instance, the relation:*

$$R \equiv x' = x + 1 \wedge 0 \leq x \leq 10^9$$

The relation R^i is consistent for all $1 \leq i \leq 10^9$ and becomes inconsistent for $i = 10^9 + 1$. Without line 13, Algorithm 1 would need at least $\lceil \frac{10^9}{3} \rceil$ iterations of the main loop in order to discover the inconsistency (line 6).

Notice that MaxPeriodic returns the span of the interval in which the relation is periodic with the current rate. The algorithm optimizes the search by storing the upper bound of the periodic interval in b_{jump}. If the sequence is periodic for none of $c = 1, \ldots, b$, line 16 updates b with the upper bound of the periodic interval in case such interval was detected, or otherwise, it increments b by one as in the unoptimized case.

Example 3.15 *(contd.) For the relation in the previous example, the prefix 10^9 is discovered after the first iteration, since the call to* MaxPeriodic *with $b = c = 1$ returns $L = 10^9$. The inconsistency of the sequence $\{R^i\}_{i=1}^{\infty}$ is discovered at the second iteration of the main loop (line 6).*

For efficiency reasons, the algorithm maintains (and updates) a prefix relation P with the following invariant property:

$$P \Leftrightarrow \bigvee_{i=0}^{b} R^i \tag{3.1}$$

By updating P at line 15, we compute part of the prefix up to the next candidate for b.

Finally, we prove the correctness of Algorithm 1 and give bounds on the number of iterations of the main loop of Algorithm 1 and on the sizes of integers b, c considered during any iteration of the main loop, in terms of the prefix and the period of the input relation.

Theorem 3.16 *If R is a periodic relation with prefix B and period C, then Algorithm 1 eventually terminates after at most $(B+C)^2+C$ iterations of the main loop at lines 4-13 and returns the transitive closure of R. Moreover, $c \leq B+C$ and $b \leq B+C+3C^2+3BC$ for each prefix b and period c considered by the algorithm.*

Proof: Let us first prove that (3.1) holds whenever the control is at one of the lines 3-13. Initially $P \Leftrightarrow R$ and $b = 1$, so (3.1) holds trivially. If (3.1) holds before executing line 15, then it will also hold after executing lines 15-16, by the definition of MaxPeriodic.

Let b_i (c_i, respectively), $i \geq 1$ be the value of b (c, respectively) during the i-th iteration of the main loop at line 5. We next make several observations.

Observation 1:	By definition of B and C, if $b_i \geq B$ and $c_i = kC$ for some $k \geq 1$, then the algorithm returns at line 12.
Observation 2:	The prefix increases with each iteration of the outer loop (line 16).
Observation 3:	For each considered prefix b, the algorithm consecutively tests periods in the range $1, \ldots, b$.

By observation 1, proving termination of Algorithm 1 amounts to showing that $b_i \geq B$ and $c_i = kC$ for some $i, k \geq 1$. We next prove that the algorithm terminates in at most $(B+C)^2+C$ iterations of the main loop and that $c_i \leq B+C$ and $b_i \leq B+C+3C^2+3BC$ for each $i \geq 1$.

First suppose that R is not $*$-consistent and hence, $R^B \Leftrightarrow \bot$ and $R^{B-1} \not\Leftrightarrow \bot$, by definition of B. The algorithm eventually reaches line 7, since b increases with each iteration of the outer loop, by observation 2. If the test at line 8 succeeds during i-th iteration, then definition of MaxConsistent procedure guarantees that $b_i + c_i K_i < B$. Since $L_i \leq K_i$ by definition of MaxPeriodic procedure, if follows that $b_i + c_i L_i < B$. Consequently $b_{i+1} \leq B + C$. If $b_{i+1} \geq B$, the algorithm terminates in the $(i+1)$-th iteration at line 7. The algorithm thus terminates after at most $(B-1)^2 + 1$ iterations of the main loop. Clearly, $b_i \leq B + C$ for each $i \geq 1$ and since $c_i \leq b_i$, it follows that $c_i \leq B + C$ too.

Next, suppose that R is $*$-consistent. Let us first consider the unoptimized algorithm without line 13. Then, b is incremented by one in each iteration of the outer loop at line 16. Consequently, the algorithm returns at line 12 when $b_i = B + C$ and $c_i = C$ at the latest, by observation 3. Clearly, the algorithm terminates after at most $(B + C)^2$ iterations of the main loop and in each iteration, $c_i \leq b_i \leq B + C$.

Next, let us consider a $*$-consistent relation and the optimized algorithm with line 13. If the algorithm returns for some $b_i \leq B+C$, the bounds follow easily. Suppose that the algorithm has not returned for some $b_i \leq B + C$ and let $i \geq 1$ be the unique index such that $b_i \leq B+C$ and $b_{i+1} > B+C$. Clearly, $c_i = b_i$ and $c_{i+1} = 1$, by observation 3. Notice that if $b_{i+1} = b_i + 1$, then $b_i = B + C$ and therefore, the algorithm terminates at line 12 for prefix-period candidate $(B + C, C)$ at the latest. Thus, if the algorithm does not terminate for a prefix candidate in the range $1..(B + C)$, it cannot be that $b_{i+1} = b_i + 1$. Consequently, $b_{i+1} > b_i + 1$ and hence, $b_{i+1} = b_{jump}$ for some $b_{jump} > b_i + 1$. Since $b_{jump} > b_i + 1$, a periodic interval $\langle b_i, \ldots, b_i + Lc_i \rangle$ of the sequence $\{\sigma(R^m)\}_{m \geq 0}$ must

have been detected for some $2 \leq L < \infty$ by the MaxPeriodic procedure and thus, $b_{i+1} = b_{jump} = b_i + (L+1)c_i$. We now demonstrate that $b_{i+1} \leq B + C + 3C^2 + 3BC$. By contradiction, suppose that $b_{i+1} > B + C + 3C^2 + 3BC$. We define (see Figure 3.1 for illustration):

$$Q = \mathrm{lcm}(C, c_i) \qquad d = \lceil \tfrac{B+C-b_i}{Q} \rceil \cdot \tfrac{Q}{c_i} \qquad e = b_i + c_i d \qquad j = (e - B) \bmod C$$

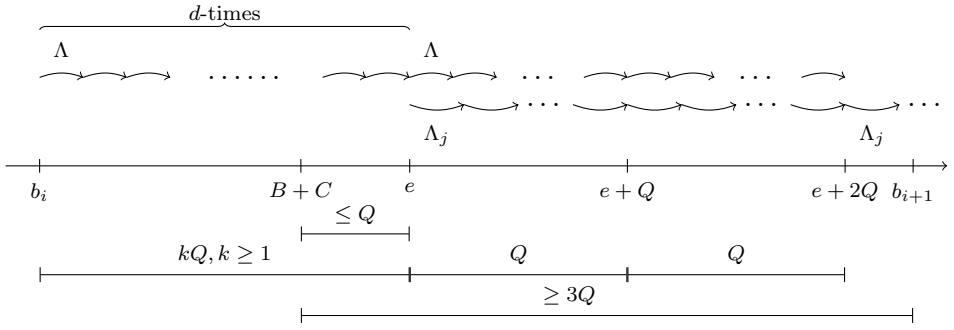


Figure 3.1: Bounding the prefix b_{i+1}

It follows that $e - b_i = kQ$ for some $k \geq 1$ and $e - (B+C) \leq Q$. Since $c_i \leq b_i \leq B + C$, it follows that $Q \leq C(B+C) = C^2 + BC$. Since we assume that $b_{i+1} > B + C + 3C^2 + 3BC$, it follows that $b_{i+1} - (B + C) \geq 3Q$ and therefore $b_{i+1} - e \geq 2Q$. Let $\Lambda_0, \ldots, \Lambda_{C-1}$ be the rates of the sequence $\{\sigma(R^m)\}_{m \geq 0}$ corresponding to B and C and Λ be the rate considered by the algorithm in the i-th iteration. Let denote $M_m = \sigma(R^m)$ for all $m \geq 0$. By periodicity of the sequence $\{\sigma(R^m)\}_{m \geq 0}$, it follows that $M_e + \frac{Q}{C}\Lambda_j = M_{e+Q}$. Recall that $b_{i+1} \geq e + 2Q$ and that $e - b_i$ is a multiple of c_i. It follows that $M_e + \frac{Q}{c_i}\Lambda = M_{e+Q}$. More generally, $M_{e+\ell c_i+Q} = M_{e+\ell c_i} + \frac{Q}{c_i}\Lambda$ for all $0 \leq \ell < \frac{Q}{c_i}$. Combining this observation with the fact that $\{M_m\}_{m \geq 0}$ is periodic with respect to B, C, it follows that

$$\begin{aligned} M_{e+\ell c_i+kQ} &= M_{e+\ell c_i} + k\tfrac{Q}{c_i}\lambda, \text{ and} \\ M_{e+(\ell+1)c_i+kQ} &= M_{e+(\ell+1)c_i} + k\tfrac{Q}{c_i}\lambda \end{aligned}$$

for all $k \geq 0$, $\ell \geq 0$. Consequently,

$$M_{e+(\ell+1)c_i+kQ} - M_{e+\ell c_i+kQ} = M_{e+(\ell+1)c_i} - M_{e+\ell c_i}$$

for all $k \geq 0$, $\ell \geq 0$. In other words, $M_{e+kc_i} = M_e + k\lambda$ for all $k \geq 0$. Combining it with $M_{b_i+kc_i} = M_{b_i} + k\lambda$ for all $0 \leq k \leq d$, where $b_i + dc_i = e$, we finally infer that $M_{b_i+kc_i} = M_{b_i} + k\lambda$ for all $k \geq 0$. This however implies that $L_i = \infty$, contradiction. Thus, $c_i \leq b_i \leq B + C + 3C^2 + 3BC$ in each iteration of the main loop. Notice that if $b_i \leq B + C$ and $b_{i+1} > B + C$, it takes at most C more iterations to return at line 12. Hence, the main loop is iterated at most $(B + C)^2 + C$ times in total.

Combining the bounds inferred in the above three cases, we obtain the bounds stated in this theorem. We finally prove if the algorithm terminates, the returned relation is indeed the transitive closure.

Algorithm 2 Closed Form of Periodic Relations

input a periodic relation R
output The closed form of R

1: **function** ClosedForm(R)
2: **let** $P \leftarrow R$ **and** $b \leftarrow 1$ **and** $b_{jump} = 1$
3: **while** *true* **do**
4: **for all** $c = 1, 2, \ldots, b$ **do**
5: **for all** $\ell = 0, 1, 2$ **do**
6: **if** $R^{b+\ell c} \Leftrightarrow \bot$ **then**
7: **return** $\widehat{R}(k, \mathbf{x}, \mathbf{x}') \Leftrightarrow \bigvee_{j=1}^{b+ic-1}(k = j) \wedge R^j$
8: **if** $\exists \Lambda \,.\, \sigma(R^b) + \Lambda = \sigma(R^{b+c}) \wedge \sigma(R^{b+c}) + \Lambda = \sigma(R^{b+2c})$ **then**
9: $K \leftarrow$ maxConsistent(R, b, Λ)
10: $L \leftarrow$ maxPeriodic(R, b, Λ, c, K)
11: **if** $L = \infty$ **then**
12: **return** $\widehat{R}(k, \mathbf{x}, \mathbf{x}') \Leftrightarrow \begin{array}{l} \left(\bigvee_{i=1}^{b+c-1}(k = i) \wedge R^i\right) \;\vee\; \exists \ell \geq 1 \,. \\ \pi(\ell \cdot \Lambda + \sigma(R^b)) \circ \left(\bigvee_{j=0}^{c-1}(k = b + \ell c + j) \wedge R^j\right) \end{array}$
13: $b_{jump} \leftarrow \max\{b_{jump}, b + c \cdot (L + 1)\}$
14: $b_{next} \leftarrow \max\{b + 1, b_{jump}\}$
15: $P \leftarrow P \vee \bigvee_{i=b}^{b_{next}-1}(k = i \wedge R^i)$
16: $b \leftarrow b_{next}$
17: **function** maxConsistent(R,b,Λ)
18: **return** $\sup\{n \in \mathbb{N} \mid \rho(n \cdot \Lambda + \sigma(R^b)) \not\Rightarrow \bot\}$
19: **function** maxPeriodic(R, b, Λ, c, K)
20: **return** $\sup\{n \leq K \mid \forall 0 \leq \ell < n \,.\, \rho(\ell \cdot \Lambda + \sigma(R^b)) \circ R^c \Leftrightarrow \rho((\ell + 1) \cdot \Lambda + \sigma(R^b))\}$

- R is not $*$-consistent, then here exists an integer $K > 0$ such that $R^K \Leftrightarrow \bot$. Then the algorithm will return at line 7, with the correct result:

$$P \vee \bigvee_{i=b+1}^{b+\ell c-1} R^i \Leftrightarrow \bigvee_{i=1}^{b} R^i \vee \bigvee_{i=b+1}^{b+\ell c-1} R^i \Leftrightarrow \bigvee_{i=1}^{b+\ell c-1} R^i$$

- R is $*$-consistent, then by previous arguments, the algorithm reaches the line 12 after at most $(B + C)^2 + C$ iterations of the main loop and returns the result:

$$\begin{array}{c} P \vee \exists k \geq 0 \,.\, \pi(k \cdot \Lambda + \sigma(R^b)) \circ \left(\bigvee_{j=0}^{c-1} R^j\right) \qquad \Leftrightarrow \\ \left(\bigvee_{i=1}^{b} R^i\right) \vee \left(\exists k \geq 0 \,.\, \pi(k \cdot \Lambda + \sigma(R^b)) \circ \left(\bigvee_{j=0}^{c-1} R^j\right)\right) \end{array}$$

  which is indeed the transitive closure of R, by Lemma 3.13 and Lemma 3.11. □

Algorithm 2 is a straightforward adaptation of Algorithm 1 that computes the closed form of a relation instead of its transitive closure, by modifying lines 7,12, and 15. Later, we use Algorithm 2 to compute transitive closures of finite monoid affine relations in Section 4.4 and to prove decidability of the termination problem in Chapter 6.

4 Periodicity of Integer Relations

This chapter is dedicated to instantiations of Algorithm 1 from Chapter 3 to three classes of arithmetic relations for which the transitive closure is known to be definable in Presburger arithmetic: difference bounds relations, octagonal relations, and finite monoid affine transformations. To compute the transitive closure of these relations using Algorithm 1, one first needs to prove that the three classes are periodic, otherwise termination of Algorithm 1 is not guaranteed. Our proofs rely mostly on a fact that any matrix is periodic when its powers are computed in the tropical semiring $(\mathbb{Z}_\infty, \min, +, \infty, 0)$. The intuition behind periodicity of difference bounds relations is that the k-th power of a relation from this class can be encoded by minimal runs of length k in *zigzag automata* which in turn can be computed as the k-th tropical power of the incidence matrix of the automaton. Thus, periodicity of the sequence of tropical powers of the incidence matrix entails periodicity of a difference bounds relation. Periodicity of the three classes thus provides common grounds to the acceleration problem and also gives shorter proofs for the fact that the transitive closures of these three classes are definable in Presburger arithmetic.

The efficiency of Algorithm 1 depends on two factors. Given a relation with prefix b and period c, Theorem 3.16 proves that Algorithm 1 makes $O((b+c)^2)$ iterations of the main loop. Thus, the prefix b and the period c are important complexity parameters, and we give asymptotic bounds for them in Chapter 5. For difference bounds and octagonal relations, these bounds are closely related to bounds on the prefix and the period of the incidence matrix of zigzag automata. This chapter therefore gives an alternative proof to the fact that each matrix is periodic in the tropical semiring which moreover gives asymptotic bounds.

Another important efficiency factor is the complexity of the procedures MaxConsistent and MaxPeriodic, which are called by Algorithm 1 to detect the maximal interval that is periodic with respect to the current prefix and period candidates. In general, for all three classes of relations we consider, these procedures can be implemented using Presburger arithmetic queries. However, in practice, one would like to avoid as much as possible using Presburger solvers, due to reasons of high complexity of decision procedures for Presburger arithmetic. In this chapter, we give direct decision methods which avoid calls to external Presburger or SMT solvers completely and which are of polynomial time complexity in the size of the prefix, period, $\|R\|$, and N, where $\|R\|$ denotes the sum of absolute values of the coefficients of a relation R and N denotes the number of variables used to define a given relation R.

Related Work. Acceleration of affine relations has been considered primarily in the works of Annichini et al. [AAB00], Boigelot [Boi99], and Finkel and Leroux [FL02]. Finite monoid affine relations have been first studied by Boigelot [Boi99] who shows that the finite monoid property is decidable and that the transitive closure is Presburger definable in this case. On what concerns non-deterministic transition relations, the transitive closure of a difference bounds constraint was first shown to be Presburger definable by Comon and Jurski [CJ98]. Their proof was subsequently simplified and extended to parametric difference bounds constraints in [BIL09]. It was shown in [BGI09] that also octagonal relations can be accelerated precisely, and that the transitive closure is also Presburger definable. The proofs of periodicity from this chapter are based on some of the previous results from [BIL09, BGI09]. For difference bounds constraints, we simplify the proof from [BIL09] by applying a result from tropical semiring theory [Sch00] on periodicity of matrices in the tropical semiring. We also give an alternative proof of matrix periodicity that moreover establishes bounds on the size of the prefix and the period of a matrix.

Roadmap. In Section 4.1, we prove that every matrix is periodic in the tropical semiring and establish asymptotic bounds on the size of its prefix and period. Next, Sections 4.2, 4.3, and 4.4 study the classes of difference bounds, octagonal, and finite monoid affine relations, respectively. In each of these sections, we prove that the respective class is periodic, present implementations of the MaxConsistent and MaxPeriodic procedures, and study their complexity. We defer all experiments with Algorithm 1 to Chapter 8.

4.1 Periodicity of Matrices

In this section, we prove that each matrix is periodic when its powers are computed in the *tropical semiring* which is defined as follows.

An *idempotent semiring* is a set $(\mathcal{S}, +, \cdot, \mathbf{0}, \mathbf{1})$ equipped with two operations, the addition $+$ and the multiplication $\cdot$, such that $(\mathcal{S}, +, \mathbf{0})$ is an idempotent (i.e., $p + p = p$ for all $p \in \mathcal{S}$) commutative monoid with neutral element $\mathbf{0}$ and $(\mathcal{S}, \cdot, \mathbf{1})$ is a monoid with neutral element $\mathbf{1}$. Moreover, multiplication distributes both left and right over addition and $\mathbf{0} \cdot r = r \cdot \mathbf{0} = \mathbf{0}$, for all $r \in \mathcal{S}$. The *tropical semiring*[1] is an idempotent semiring $(\mathbb{Z}_\infty, \min, +, \infty, 0)$ [Sch00] with the extended arithmetic operations $x + \infty = \infty$, and $\min(x, \infty) = x$, for all $x \in \mathbb{Z}$, where $\min(x, y)$ denotes the minimum between the values x and y.

If S is a set, let $S^{m \times m}$ denote the set of square matrices of size m, with entries in S. For two matrices $A, B \in \mathbb{Z}_\infty^{m \times m}$, we define the sum $(A + B)_{ij} = A_{ij} + B_{ij}$. The classical product is defined for $A, B \in \mathbb{Z}^{m \times m}$ as $(A \times B)_{ij} = \sum_{k=1}^{m} (a_{ik} \cdot b_{kj})$. The *tropical product* is defined for $A, B \in \mathbb{Z}_\infty^{m \times m}$ as $(A \boxtimes B)_{ij} = \min_{k=1}^{m} (a_{ik} + b_{kj})$. Let $\mathbf{I}_m \in \mathbb{Z}^{m \times m}$ be the identity matrix, i.e. $\mathbf{I}_{ii} = 1$ and $\mathbf{I}_{ij} = 0$, for all $1 \leq i, j \leq m$, $i \neq j$, and $\mathbb{I}_m \in \mathbb{Z}_\infty^{m \times m}$ be

[1] Actually, the dual structure $(\mathbb{Z}_{-\infty}, \max, +, -\infty, 0)$ is also known as the tropical semiring in the literature.

the tropical identity matrix, i.e. $\mathbb{I}_{ii} = 0$ and $\mathbb{I}_{ij} = \infty$, for all $1 \leq i, j \leq m$, $i \neq j$. Then we define $A^0 = \mathbf{I}$, $A^{\boxtimes 0} = \mathbb{I}$ and $A^k = A^{k-1} \times A$, $A^{\boxtimes k} = A^{\boxtimes k-1} \boxtimes A$, for all $k > 0$.

With these notions, a periodic matrix can be defined as follows.

Definition 4.1 *A matrix $A \in \mathbb{Z}_\infty^{m \times m}$ is called periodic if the sequence of tropical powers $\{A^{\boxtimes k}\}_{k=1}^\infty$ is periodic.*

Intuitively, if A is the incidence matrix of a weighted digraph, then the sequence $\{A^{\boxtimes k}\}_{k=1}^\infty$ of tropical powers of A gives the minimal weight paths of lengths $k = 1, 2, \ldots$ between any two vertices of the graph. It has been proved in [Sch00] that every matrix $A \in \mathbb{Z}_\infty^{m \times m}$ is periodic. We define the prefix (period) of a matrix A as the prefix (period) of the sequence $\{A^{\boxtimes k}\}_{k=1}^\infty$. We will often refer to periodicity (or prefix, period) of graphs, by which we mean periodicity (or prefix, period, respectively) of the corresponding incidence matrix.

We have argued that the complexity of the transitive closure algorithm depends on the size of the prefix and the period of the input relations, which will be later (in Section 4.2) shown to be bounded by the size of the prefix and the period of a certain kind of graphs, for difference bounds and octagonal relations. The result of [Sch00] is however not suitable to establish bounds on the prefix and period of a graph. In this section, we therefore give an alternative proof of periodicity of matrices that moreover establishes bounds on sizes of their prefix and period.

Recall from Section 2.1 that given a path π, we denote by $w(\pi)$ its weight and that $\overline{w}(\pi)$ denotes its average weight. If $\lambda_1, \ldots, \lambda_k$ are pairwise distinct elementary cycles, the expression $\theta = \sigma_1 \cdot \lambda_1^* \cdot \sigma_2 \ldots \sigma_k \cdot \lambda_k^* \cdot \sigma_{k+1}$ is called a *path scheme of size k*. A path scheme $\theta = \sigma_1.\lambda_1^*.\sigma_2$ such that $|\sigma_1.\sigma_2| \leq |V|^4$ is called *basic*. A path scheme encodes the infinite set of paths $[\![\theta]\!] = \{\sigma_1 \cdot \lambda_1^{n_1} \cdot \sigma_2 \ldots \sigma_k \cdot \lambda_k^{n_k} \cdot \sigma_{k+1} \mid n_1, \ldots, n_k \in \mathbb{N}\}$. Given a weighted digraph $G = \langle V, E, w \rangle$, we denote by M_G its incidence matrix.

The first observation is that, for every minimal weight path in a weighted graph, there is an equivalent path following a path scheme whose size is bounded by the square of the size of the graph.

Proposition 4.2 *Let $G = \langle V, E, w \rangle$ be a weighted digraph and ρ be a minimal weight path in G. Then there exists a path scheme $\theta = \sigma_1 \cdot \lambda_1^* \cdot \ldots \cdot \sigma_k \cdot \lambda_k^* \cdot \sigma_{k+1}$ in G, such that $\sigma_1, \ldots, \sigma_{k+1}$ are acyclic and $k \leq \|V\|^2$, and a path $\rho' \in [\![\theta]\!]$ starting and ending in the same vertices as ρ, such that $|\rho| = |\rho'|$ and $w(\rho) = w(\rho')$.*

Proof: For each vertex $v \in V$, we partition the set of elementary cycles that start and end in v, according to their length. The representative of each equivalence class is chosen to be a cycle of minimal weight in the class. Since the length of each elementary cycle is at most $\|V\|$, there are at most $\|V\|^2$ such equivalence classes.

Let ρ be any path of minimal weight in G. First, notice that ρ can be factorized as:

$$\rho = \sigma_1 \cdot \lambda_1 \cdot \ldots \cdot \sigma_k \cdot \lambda_k \cdot \sigma_{k+1}$$

where $\sigma_1, \ldots, \sigma_{k+1}$ are elementary acyclic paths, and $\lambda_1, \ldots, \lambda_k$ are elementary cycles. This factorization can be achieved by a traversal of ρ while collecting the vertices along

the way in a bag. The first vertex which is already in the bag marks the first elementary cycle. Then we empty the bag and continue until the entire path is traversed.

Next, we repeat the following two steps until nothing changes:

1. For all $i = 1, \ldots, k-1$ move all cycles λ_j, $j > i$, starting and ending with the same vertex as λ_i, next to λ_i, in the ascending order of their lengths. The result is a path ρ' of the same length and weight as ρ.

2. Factorize any remaining non-elementary acyclic path $\sigma_i \cdot \sigma_{i+1} \cdot \ldots \cdot \sigma_{i+j}$ as in the previous.

The loop above is shown to terminate, since the sum of the lengths of the remaining acyclic paths decreases with every iteration. The result is a path of the same length and weight as ρ, which starts and ends in the same vertices as ρ, in which all elementary cycles of the same length are grouped together. Since $w(\rho)$ is minimal for $|\rho|$, same holds for ρ', and moreover, all elementary cycles can be replaced by their equivalence class representatives, without changing neither the length, nor the weight of the path. The result is a path which belongs to a scheme with at most $\|V\|^2$ cycles. □

Second, for every minimal weight path in the graph, there exists an equivalent path which follows a basic path scheme.

Lemma 4.3 *Let $G = \langle V, E, w\rangle$ be a weighted digraph and ρ be a minimal weight path. Then there exists a path ρ', starting and ending in the same vertices as ρ, such that $w(\rho) = w(\rho')$ and $|\rho| = |\rho'|$, and a basic path scheme $\theta = \sigma \cdot \lambda^* \cdot \sigma'$ such that $\rho' \in [\![\theta]\!]$.*

Proof: By Proposition 4.2, for any path ρ in G there exists a path scheme $\theta = \sigma_1 \cdot \lambda_1^* \cdot \sigma_2 \ldots \sigma_k \cdot \lambda_k^* \cdot \sigma_{k+1}$, such that $\sigma_1, \ldots, \sigma_{k+1}$ are acyclic and $k \leq \|V\|^2$, and a path ρ', starting and ending in the same vertices as ρ, of the same weight and length as ρ, such that $\rho' = \sigma_1 \cdot \lambda_1^{n_1} \cdot \sigma_2 \ldots \sigma_k \cdot \lambda_k^{n_k} \cdot \sigma_{k+1}$ for some $n_1, \ldots, n_k \geq 0$. Suppose that λ_i is a cycle with minimal average weight among all cycles in the scheme, i.e. $\frac{w(\lambda_i)}{|\lambda_i|} \leq \frac{w(\lambda_j)}{|\lambda_j|}$, for all $1 \leq j \leq k$. For each n_j there exist $p_j \geq 0$ and $0 \leq q_j < |\lambda_i|$, such that $n_j = p_j \cdot |\lambda_i| + q_j$. Let ρ' be the path:

$$\sigma_1 \cdot \lambda_1^{q_1} \cdot \sigma_2 \ldots \sigma_{i-1} \cdot \lambda_i^{n_i + \sum_{j=1}^{i-1} p_j \cdot |\lambda_j| + \sum_{j=i+1}^{k} p_j \cdot |\lambda_j|} \cdot \sigma_{i+1} \cdot \ldots \sigma_k \cdot \lambda_k^{q_k} \cdot \sigma_{k+1}$$

It is easy to check that $|\rho'| = |\rho|$ and $w(\rho') \leq w(\rho)$.

Clearly ρ' follows the path scheme $\rho_1 \cdot \lambda_i^* \cdot \rho_2$, where $\rho_1 = \sigma_1 \cdot \lambda_1^{q_1} \cdot \sigma_2 \ldots \sigma_{i-1}$ and $\rho_2 = \sigma_{i+1} \cdot \ldots \sigma_k \cdot \lambda_k^{q_k} \cdot \sigma_{k+1}$. Since $\lambda_1, \ldots, \lambda_k$ are elementary paths, all their lengths are strictly smaller than $\|V\|$. Since $q_j < |\lambda_i| \leq \|V\|$, and $k \leq \|V\|^2$, by Proposition 4.2, we have that $|\rho_1 \cdot \rho_2| < \|V\|^4$. Thus, $\rho_1 \cdot \lambda_i^* \cdot \rho_2$ is basic. □

The following lemma shows that, for a sufficiently long minimal weight path, there exists an equivalent path which follows a basic path scheme and moreover, this path scheme is followed by infinitely many minimal paths. Recall that $\mu(G) = \max\{|n| \mid u \xrightarrow{n} v \text{ in } G\}$ denotes the maximum absolute value of all weights in G.

Lemma 4.4 *Let $G = \langle V, E, w\rangle$ be a weighted digraph, and $u, v \in V$ be two vertices. Then for every minimal weight path ρ form u to v, such that $|\rho| \geq \mu(G) \cdot \|V\|^6$, there exists a path ρ' from u to v, such that $w(\rho) = w(\rho')$ and $|\rho| = |\rho'|$, and a basic path scheme $\theta = \sigma \cdot \lambda^* \cdot \sigma'$, such that $\rho' = \sigma \cdot \lambda^b \cdot \sigma'$, for some $b \geq 0$. Moreover, there exists $c \mid \frac{lcm(1,\ldots,\|V\|-1)}{|\lambda|}$ such that $\sigma \cdot \lambda^{b+kc} \cdot \sigma'$ is a minimal weight path from u to v, for all $k \geq 0$.*

Proof: By Lemma 4.3, every minimal weight path from u to v follows a basic path scheme. Let $L > 0$ be an integer, and let $\sigma_i \cdot \lambda_i^* \cdot \sigma_i'$ and $\sigma_j \cdot \lambda_j^* \cdot \sigma_j'$ be two possible path schemes such that $\rho_i = \sigma_i \cdot \lambda_i^{b_i} \cdot \sigma_i'$, $\rho_j = \sigma_j \cdot \lambda_j^{b_j} \cdot \sigma_j'$ are two paths of length L, for some $b_i, b_j \geq 0$. We assume without loss of generality that λ_i has smaller average weight, i.e. $\overline{w}(\lambda_i) < \overline{w}(\lambda_j)$. We first prove that, if $L \geq \mu(G) \cdot \|V\|^6$, then $w(\rho_i) \leq w(\rho_j)$. We have:

$$\begin{aligned} b_i &= \tfrac{L-|\sigma_i \cdot \sigma_i'|}{|\lambda_i|}, \\ b_j &= \tfrac{L-|\sigma_j \cdot \sigma_j'|}{|\lambda_j|}, \end{aligned}$$

and

$$\begin{aligned} w(\rho_i) &= w(\sigma_i \cdot \sigma_i') + \tfrac{L-|\sigma_i \cdot \sigma_i'|}{|\lambda_i|} w(\lambda_i), \\ w(\rho_j) &= w(\sigma_j \cdot \sigma_j') + \tfrac{L-|\sigma_j \cdot \sigma_j'|}{|\lambda_j|} w(\lambda_j). \end{aligned}$$

Then $w(\rho_i) \leq w(\rho_j)$ if and only if

$$L \geq \frac{|\lambda_i||\lambda_j|(w(\sigma_i \cdot \sigma_i') - w(\sigma_j \cdot \sigma_j')) + |\lambda_i||\sigma_j \cdot \sigma_j'|w(\lambda_j) - |\lambda_j||\sigma_i \cdot \sigma_i'|w(\lambda_i)}{w(\lambda_j)|\lambda_i| - w(\lambda_i)|\lambda_j|}.$$

Since λ_i has a strictly smaller average weight than λ_j, we have that $w(\lambda_j)|\lambda_i| - w(\lambda_i)|\lambda_j| > 0$, and, since $w(\lambda_i), w(\lambda_j), |\lambda_i|, |\lambda_j| \in \mathbb{Z}$, we have $w(\lambda_j)|\lambda_i| - w(\lambda_i)|\lambda_j| \geq 1$. By Lemma 4.3, we have $|\sigma_i \cdot \sigma_i'|, |\sigma_j \cdot \sigma_j'| \leq \|V\|^4$, and $w(\sigma_i \cdot \sigma_i') - w(\sigma_j \cdot \sigma_j') \leq \mu(G) \cdot \|V\|^4$. We compute:

$$\begin{aligned} L &\geq \mu(G) \cdot \|V\|^6 \\ &\geq |\lambda_i||\lambda_j|(w(\sigma_i \cdot \sigma_i') - w(\sigma_j \cdot \sigma_j')) + |\lambda_i||\sigma_j \cdot \sigma_j'|w(\lambda_j) - |\lambda_j||\sigma_i \cdot \sigma_i'|w(\lambda_i) \\ &\geq \tfrac{|\lambda_i||\lambda_j|(w(\sigma_i \cdot \sigma_i') - w(\sigma_j \cdot \sigma_j')) + |\lambda_i||\sigma_j \cdot \sigma_j'|w(\lambda_j) - |\lambda_j||\sigma_i \cdot \sigma_i'|w(\lambda_i)}{w(\lambda_j)|\lambda_i| - w(\lambda_i)|\lambda_j|} \end{aligned}$$

Since the choice of ρ_i and ρ_j was arbitrary, for each $L \geq \mu(G) \cdot \|V\|^6$, the path scheme with minimal average weight cycle is chosen by the minimal weight path of length L.

Second, we show that this happens periodically. For two paths $\rho_i = \sigma_i \cdot \lambda_i^{b_i} \cdot \sigma_i'$ and $\rho_j = \sigma_j \cdot \lambda_j^{b_j} \cdot \sigma_j'$ of equal lengths, as before, let $c_{ij} = lcm(|\lambda_i|, |\lambda_j|)$, $c_i = \frac{c_{ij}}{|\lambda_i|}$ and $c_j = \frac{c_{ij}}{|\lambda_j|}$. We have that $|\lambda_i^{kc_i}| = |\lambda_j^{kc_j}| = kc_{ij}$, for all $k \geq 0$. Moreover, since $\overline{w}(\lambda_i) < \overline{w}(\lambda_j)$, we have that $w(\lambda_i^{kc_i}) < w(\lambda_j^{kc_j})$. It follows that

$$w(\sigma_i \cdot \lambda_i^{b_i + kc_i} \cdot \sigma_i') \leq w(\sigma_j \cdot \lambda_j^{b_j + kc_j} \cdot \sigma_j')$$

for all $k \geq 0$. Finally, since $|\lambda_i|, |\lambda_j| < \|V\|$, we have that $c_{ij} \mid lcm(1, \ldots, \|V\| - 1)$ and thus $c_i \mid \frac{lcm(1,\ldots,\|V\|-1)}{|\lambda_i|}$. Since the choice of i does not change this fact, it is enough to take $c = \frac{lcm(1,\ldots,\|V\|-1)}{|\lambda|}$ and $b = b_i$ to obtain that $\sigma_i \cdot \lambda_i^{b+kc} \cdot \sigma_i'$ has minimal weight among all paths from u to v of the same length, for all $k \geq 0$. □

The following lemma is essential to prove an upper bound on the period of weighed digraphs.

Lemma 4.5 *For each $n \geq 1$, $lcm(1, \ldots, n)$ is bounded by $2^{\mathcal{O}(n)}$.*

Proof: We know that $lcm(1, \ldots, n) = \prod_{p \leq n} p^{\lfloor log_p(n) \rfloor}$ where the product is taken only over primes p. Obviously, for every prime p we have that $p^{\lfloor log_p(n) \rfloor} \leq p^{log_p(n)} = n$. Hence, $lcm(1, \ldots, n) \leq \prod_{p \leq n} n = n^{\pi(n)}$, where $\pi(n)$ denotes the prime-counting function (which gives the number of primes less than or equal to n, for every natural number n). Using the prime number theorem which states that $\lim_{n\to\infty} \frac{\pi(n)}{n/ln(n)} = 1$ we can effectively bound $\pi(n)$. That is, for any $\epsilon > 0$, there exists n_ϵ such that $\frac{\pi(n)}{n/ln(n)} \leq (1 + \epsilon)$ for all $n \geq n_\epsilon$. Consequently, $n^{\pi(n)} \leq n^{(1+\epsilon)n/ln(n)} = e^{(1+\epsilon)n} = 2^{log_2(e)(1+\epsilon)n} = 2^{\mathcal{O}(n)}$ for all $n \geq n_\epsilon$, and completes the proof. □

The following theorem gives asymptotic bounds on the size of the prefix and the period of a weighted digraph $G = \langle V, E, w \rangle$, in terms of $\mu(G)$ and $\|V\|$.

Theorem 4.6 *Let $G = \langle V, E, w \rangle$ be a digraph, and $M_G \in \mathbb{Z}_\infty^{\|V\| \times \|V\|}$ be its incidence matrix. Then, the sequence $\{M_G^{\boxtimes^i}\}_{i \geq 0}$ is periodic. Moreover, its prefix b is bounded by $\mu(G) \cdot \mathcal{O}(\|V\|^6)$, and its period divides* $\text{lcm}(1, \ldots, \|V\| - 1)$ *and is bounded by $2^{\mathcal{O}(\|V\|)}$.*

Proof: A direct consequence of Lemma 4.4 is that each minimal weight path ρ in G of length at least $\mu(G) \cdot \|V\|^6$ must be of the form $\rho = \sigma \cdot \lambda^b \cdot \sigma'$, and moreover, for some $c \mid \frac{lcm(1,\ldots,\|V\|-1)}{|\lambda|}$, we have that $\sigma \cdot \lambda^{b+kc} \cdot \sigma'$ is a minimal weight path, for all $k \geq 0$. Hence, for each $1 \leq i, j \leq \|V\|$, the sequence $\{(M_G^{\boxtimes^k})_{ij}\}_{k=0}^\infty$ is periodic with prefix at most $\mu(G) \cdot \|V\|^6$ and period which divides $lcm(1, \ldots, \|V\| - 1)$. The prefix b of M_G is the maximum of all prefixes, and the period c is the least common multiple of the periods of the sequences $\{(M_G^{\boxtimes^i})_{ij}\}_{k=0}^\infty$, respectively. Hence b is bounded by $\mu(G) \cdot \mathcal{O}(\|V\|^6)$ and c divides $\text{lcm}(1, \ldots, \|V\| - 1)$. By Lemma 4.5, $lcm(1, \ldots, \|V\| - 1)$ is bounded by $2^{\mathcal{O}(\|V\|)}$. Since c divides $\text{lcm}(1, \ldots, \|V\| - 1)$, the same bound for c follows immediately. □

4.2 Difference Bounds Relations

Recall from Section 2.3 that a difference bounds relation $R \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$ can be equivalently represented as a difference bounds matrix (DBM) M_R. Similarly, for each DBM M there is a corresponding difference bounds relation Φ_M. Furthermore, difference bounds relations can be represented canonically by DBMs M_R^*.

The first step to proving that the class $\mathcal{R}_{db}$ is periodic is defining the mappings between relations and matrices (Definition 3.6). Given a consistent difference bounds relation $R \in \mathcal{R}_{db}$, we define $\sigma(R) = M_R^* \in \mathbb{Z}_\infty^{2N \times 2N}$ to be the closed characteristic DBM of R. Dually, for any DBM $M \in \mathbb{Z}_\infty^{2N \times 2N}$, let $\rho(M) = \Phi_M \in \mathcal{R}_{db}$ be the difference bounds relation corresponding to M. We clearly have $\rho(\sigma(R)) \Leftrightarrow R$, for all consistent relations R, as required by Definition 3.6.

In order to define a function $\pi : \mathbb{Z}[k]_\infty^{m \times m} \to \mathcal{R}[k]$ mapping matrices of linear terms of the form $\alpha \cdot k + \beta$, with integer coefficients, into parametric relations $R(k, \mathbf{x}, \mathbf{x}')$, we define the class of *parametric* difference bounds relations.

Definition 4.7 *A formula $\phi(\mathbf{x}, k)$ is a* parametric difference bounds constraint *if it is equivalent to a finite conjunction of atomic propositions of the form $x_i - x_j \leq t_{ij}$, for some $1 \leq i, j \leq N$, $i \neq j$, where t_{ij} are univariate linear terms in k.*

The class of parametric difference bounds relations with parameter k is denoted as $\mathcal{R}_{db}[k]$. Similar to the non-parametric case (Definition 2.3), a parametric difference bounds constraint $\phi(k)$ can be represented by a matrix $M_\phi[k] \in \mathbb{Z}[k]_\infty^{N \times N}$ of univariate linear terms, where $(M_\phi[k])_{ij} = t_{ij}$ if $x_i - x_j \leq t_{ij}$ occurs in ϕ, and ∞ otherwise. Dually, a matrix $M[k]$ of linear terms corresponds to the formula $\Phi_M(k) \Leftrightarrow \bigwedge_{M[k]_{ij} \neq \infty} x_i - x_j \leq M[k]_{ij}$. With these considerations, we define $\pi(M[k]) = \Phi_M(k)$ to be the parametric counterpart of the ρ function from Definition 3.6. Clearly, for each matrix $M \in \mathbb{Z}[k]_\infty^{m \times m}$, the mapping π satisfies the required property that $\pi(M)[n/k] \Leftrightarrow \rho(M[n])$ for all $n \in \mathbb{Z}$.

4.2.1 Proving Periodicity

In this section, we prove that all relations defined using difference bounds constraints are periodic in the sense of Definition 3.6. A direct consequence is that these relations are also periodic, which ensures the termination of Algorithm 1 on the $\mathcal{R}_{db}$ class.

Let $R \in \mathcal{R}_{db}$ be an arbitrary difference bounds relation for the rest of this section. If R is not $*$-consistent, then by Definition 3.6, R is periodic. We consider from now on that R is $*$-consistent and prove that the sequence $\{\sigma(R^i)\}_{i=0}^\infty$ is periodic in this case. We have $\sigma(R^i) = M_{R^i}^*$ for any $i \geq 0$.

The proof idea is that the entries of the sequence $\{M_{R^i}^*\}_{i=0}^\infty$ represent minimal weight paths in the graph corresponding to the i-times "unfolding" of R, for any $i \geq 0$. These paths form a regular language recognized by a finite weighted automaton. Consequently, the minimal weights for $i = 0, 1, 2, \ldots$ are entries in the sequence of tropical powers of the incidence matrix of this automaton. But then they form periodic sequences, according to Theorem 4.6.

For all $1 \leq i, j \leq N$, we obtain the following equalities:

$$\begin{array}{lcl} [\sigma(R^m)]_{i,j} & = & \min\{x_i^0 \to x_j^0\} \\ [\sigma(R^m)]_{i+N,j+N} & = & \min\{x_i^m \to x_j^m\} \\ [\sigma(R^m)]_{i,j+N} & = & \min\{x_i^0 \to x_j^m\} \\ [\sigma(R^m)]_{i+N,j} & = & \min\{x_i^m \to x_j^0\} \end{array} \tag{1}$$

Recall the definition of the *zigzag automata* from Section 2.3.3 that recognize paths within constraint graphs. In the following, we view these automata as reasoning tools, needed to prove the periodicity of the difference bounds constraints. Recall that $T_R = \langle Q, \Delta, w \rangle$ is the common transition table of all zigzag automata for $R \in \mathcal{R}_{db}$. Let $\mathcal{M}_R \in \mathbb{Z}_\infty^{\|Q\| \times \|Q\|}$ be the incidence matrix of T_R, where $\|Q\|$ is the number of control states in T_R. Without loss of generality, we assume that states in Q are both reachable and co-reachable[2]. For each pair of variables x_i, x_j, there are eight indices, denoted as $I^{ef}_{i,j}$, F^{ef}, I^{eb}, $F^{eb}_{i,j}$, I^{of}_i, F^{of}_j, I^{ob}_i, $F^{ob}_j \in \{1, \ldots, \|Q\|\}$ corresponding to the initial and final states of the four zigzag automata, respectively. According to Lemma 2.15, the minimal weight path of length $m+2$ from $I^{ef}_{i,j}$ to F^{ef} matches the minimal weight path between the extremal points x^0_i and x^0_j of $\mathcal{G}^m_R$. Similarly for paths from I^{eb} to $F^{eb}_{i,j}$, from I^{of}_i to F^{of}_j, and from I^{ob}_i to F^{ob}_j. However the weights of the paths in the zigzag automata are captured by the tropical powers of $\mathcal{M}_R$, as follows:

$$\begin{array}{rcl} \min\{x^0_i \to x^0_j\} & = & \left[{\mathcal{M}_R}^{\boxtimes^{m+2}}\right]_{I^{ef}_{i,j}, F^{ef}} \\ \min\{x^m_i \to x^m_j\} & = & \left[{\mathcal{M}_R}^{\boxtimes^{m+2}}\right]_{I^{eb}, F^{eb}_{i,j}} \\ \min\{x^0_i \to x^m_j\} & = & \left[{\mathcal{M}_R}^{\boxtimes^{m+2}}\right]_{I^{of}_i, F^{of}_j} \\ \min\{x^m_i \to x^0_j\} & = & \left[{\mathcal{M}_R}^{\boxtimes^{m+2}}\right]_{I^{ob}_i, F^{ob}_j} \end{array} \tag{2}$$

By Theorem 4.6, the tropical powers of $\mathcal{M}_R$ form a periodic sequence, therefore the sequence $\{{\mathcal{M}_R}^{\boxtimes^{m+2}}\}_{m \geq 0}$ is periodic. By equating the equivalences (1) and (2) from the previous, we obtain that the sequence $\{\sigma(R^m)\}_{m \geq 0}$ is periodic as well. The following theorem summarizes the above arguments.

Theorem 4.8 *The class of difference bounds relations is periodic.*

Moreover, since M_{R^m} is a projection of $\mathcal{M}_R^{\boxtimes^{m+2}}$ for all $m \geq 0$ if R is $*$-consistent, the prefix of a $*$-consistent relation R is bounded by the prefix of T_R. Similar claim can be made for the period of R.

Proposition 4.9 *Let $T_R = (Q, \Delta, w)$ be the common transition table of zigzag automata defined for a $*$-consistent difference bounds relation R. Then, the size of the prefix of R is bounded by the size of the prefix of T_R and the period of R divides the period of T_R.*

Proof: The proof of Lemma 3.4 shows that if b_{ij} (c_{ij}) is the prefix (period) of the sequence $\{(\mathcal{M}_R^{\boxtimes^m})_{ij}\}_{m \geq 0}$ for all $1 \leq i, j \leq \|Q\|$, then the sequence $\{\mathcal{M}_R^{\boxtimes^m}\}_{m \geq 0}$ has the prefix defined as $b = \max_{ij}\{b_{ij}\}$ and the period defined as $c = \mathrm{lcm}_{ij}\{c_{ij}\}$. Since R is $*$-consistent, M_{R^m} is a projection of $\mathcal{M}_R^{\boxtimes^{m+2}}$ for all $m \geq 0$. Then the bounds stated in this proposition follow from the definition of b and c. □

[2] A state is said to be *reachable* if there exists a path from an initial state to it, and *co-reachable* if there exists a path from it to a final state.

In conclusion, Algorithm 1 will terminate on difference bounds relations. Moreover, the result is formula definable in Presburger arithmetic. In particular, this also simplifies the proof that transitive closures of difference bounds relations are Presburger definable, from [BIL09]. The following result is needed in the following section to design a cost-effective implementation of Algorithm 1.

Corollary 4.10 *If $R \in \mathcal{R}_{db}$ be a difference bounds relation, the rate Λ of the periodic sequence $\{\sigma(R^i)\}_{i=0}^{\infty}$ is a closed DBM.*

Proof: Since $\sigma(R^m) = M^*_{R^m}$ for all $m \geq 0$, $\sigma(R^m)$ is closed and thus we have for all $1 \leq i, j, k \leq 2N$ and $m \geq 0$:

$$\left[\sigma(R^m)\right]_{i,j} \leq \left[\sigma(R^m)\right]_{i,k} + \left[\sigma(R^m)\right]_{k,j}$$

Since $\{\sigma(R^m)\}_{m=0}^{\infty}$ is periodic, there exists $b \geq 0$, $c > 0$ and $\Lambda \in \mathbb{Z}_{\infty}^{2N \times 2N}$ such that:

$$\sigma(R^{b+nc}) = \sigma(R^b) + n \cdot \Lambda$$

for all $n \geq 0$. Consequently, we have, for all $1 \leq i, j, k \leq 2N$ and all $n \geq 0$:

$$\left[\sigma(R^b)\right]_{i,j} + n \cdot \Lambda_{ij} \leq \left[\sigma(R^b)\right]_{i,k} + \left[\sigma(R^b)\right]_{k,j} + n \cdot (\Lambda_{ik} + \Lambda_{kj})$$

We obtain:

$$n \cdot (\Lambda_{ij} - \Lambda_{ik} - \Lambda_{kj}) \leq \left[\sigma(R^b)\right]_{i,k} + \left[\sigma(R^b)\right]_{k,j} - \left[\sigma(R^b)\right]_{i,j},\ \forall n \geq 0$$

Suppose that Λ is not closed i.e., there exist $1 \leq i, j, k \leq 2N$ such that $\Lambda_{ij} > \Lambda_{ik} + \Lambda_{kj}$, we get a contradiction with the above. □

4.2.2 Checking $*$-consistency and Periodicity

In this section, we describe cost-effective ways to implement the MaxConsistent and MaxPeriodic procedures from Algorithm 1 for difference bounds relations.

First, we need to introduce several technical notions. A *univariate linear term* is a term of the form $\alpha \cdot k + \beta$, where $\alpha, \beta \in \mathbb{Z}$ are integer constants. Let $\mathbb{Z}[k]$ denote the set of all univariate linear terms with variable k. For two sets $S, T \subseteq \mathbb{Z}[k]$, we define $S \oplus T = \{(\alpha_1 + \alpha_2) \cdot k + \beta_1 + \beta_2 \mid \alpha_1 \cdot k + \beta_1 \in S,\ \alpha_2 \cdot k + \beta_2 \in T\}$. For two linear terms $t_1 = \alpha_1 \cdot k + \beta_1$ and $t_2 = \alpha_2 \cdot k + \beta_2$, we define the partial order on terms $t_1 \preceq t_2 \Leftrightarrow \alpha_1 \leq \alpha_2 \ \wedge \ \beta_1 \leq \beta_2$. We denote the strict inequality on terms by $t_1 \prec t_2 \Leftrightarrow t_1 \preceq t_2 \ \wedge \ t_2 \not\preceq t_1$. For a finite set of linear terms S, we denote by $\textsc{MinTerms}(S) = \{t \in S \mid \forall s \in S \,.\, s \not\prec t\}$ the set of minimal terms in S, with respect to this order. For a set of integer constants $\{a_1, \ldots, a_n\}$, we denote by $\textsc{MaxMin}\{a_i\}_{i=1}^n$ the positive value $\max\{a_i\}_{i=1}^n - \min\{a_i\}_{i=1}^n$.

Proposition 4.11 *Let $S = \{\alpha_i \cdot k + \beta_i\}_{i=1}^m$ be a set of univariate linear terms. Then*

$$\|\textsc{MinTerms}(S)\| \leq \min(\textsc{MaxMin}\{\alpha_j\}_{j=1}^m, \textsc{MaxMin}\{\beta_j\}_{j=1}^m)$$

Proof: A term $\alpha \cdot k + \beta$ corresponds to the point (α, β) in the 2-dimensional space. All terms from S are points in the rectangle defined by the bottom left corner $(\min\{\alpha_j\}_{j=1}^m, \min\{\beta_j\}_{j=1}^m)$ and the upper right corner $(\max\{\alpha_j\}_{j=1}^m, \max\{\beta_j\}_{j=1}^m)$. Since all terms in $\textsc{MinTerms}(S)$ are incomparable w.r.t. $\preceq$, there can be at most $\min(\max\{\alpha_j\}_{j=1}^m - \min\{\alpha_j\}_{j=1}^m, max\{\beta_j\}_{j=1}^m - \min\{\beta_j\}_{j=1}^m)$ such terms. Hence the result. □

Given a linear term $t = \alpha \cdot k + \beta$, we denote by $t(n)$ the value $\alpha \cdot n + \beta$, for any $n \in \mathbb{N}$. The set of valuations of a term t, with respect to the threshold ℓ is $[\![t]\!]_{\geq \ell} = \{t(n) \mid n \geq \ell\}$. These notations are naturally lifted to sets of terms, i.e. $T(n) = \{t(n) \mid t \in T\}$. and $[\![T]\!]_{\geq \ell} = \bigcup_{t \in T} [\![t]\!]_{\geq \ell}$.

Unlike DBMs with constant entries, parametric DBMs do not have a closed form, since in general, the minimum of two univariate linear terms cannot be defined again as a linear term. A way around this problem is using matrices of sets of univariate linear terms, with the convention that a set $T = \{t_1(k), \ldots, t_n(k)\}$ of univariate linear terms denotes the function $k \mapsto min\{t_1, \ldots, t_n\}$, and $\min(\emptyset) = \infty$. The Floyd-Warshall algorithm for computing closed forms of DBMs with constant entries can be easily adapted to parametric DBMs.

Algorithm 3 Closure Algorithm for Parametric DBMs

```
1:  procedure ParametricFW(M)
2:      for all i = 0, ..., m − 1 do
3:          for all j = 0, ..., m − 1 do
4:              if M[i][j] = ∞ then
5:                  M[i][i] ← ∅
6:              else
7:                  M[i][j] ← {M[i][j]}
8:      for all k = 0, ..., m − 1 do
9:          for all i = 0, ..., m − 1 do
10:             for all j = 0, ..., m − 1 do
11:                 T0 ← M[i][j]
12:                 T1 ← M[i][k]
13:                 T2 ← M[k][j]
14:                 M[i][j] ← MinTerms (T0 ∪ (T1 ⊕ T2))
15:     return M[i][j]
16: procedure MinTerms(S)
17:     return {t ∈ S | ∀s ∈ S . s ⋠ t}
```

Algorithm 3 takes as input a matrix of univariate linear terms, and produces a matrix of sets of such terms (each set of terms T is interpreted as $\min(T)$). Lines 2-7 initialize the

output matrix with sets of terms. Lines 8-14 correspond to the classical Floyd-Warshall iteration.

Proposition 4.12 *Let $M \in \mathbb{Z}_\infty^{m\times m}[k]$ be a parametric DBM, such that $M_{ij} = \alpha_{ij}\cdot k+\beta_{ij}$, for all $1 \leq i, j \leq m$. Then, Algorithm 3 runs in at most $\mathcal{O}(\mu^3 \cdot m^6)$ time where*

$$\mu = \min(\max_{1\leq i,j\leq m}\{|\alpha_{ij}|\}, \max_{1\leq i,j\leq m}\{|\beta_{ij}|\})$$

Moreover, we have $\|\mathcal{M}_{ij}\| \leq 2m \cdot \mu$.

Proof: Each term $\alpha \cdot k + \beta \in \mathcal{M}[i][j]$ is a sum of at most m terms $\alpha_{ij} \cdot k + \beta_{ij}$. We have:

$$\begin{array}{rcl} -m \cdot \max_{1\leq i,j\leq m}\{|\alpha_{ij}|\} & \leq & \alpha \leq m \cdot \max_{1\leq i,j\leq m}\{|\alpha_{ij}|\}, \text{ and} \\ -m \cdot \max_{1\leq i,j\leq m}\{|\beta_{ij}|\} & \leq & \beta \leq m \cdot \max_{1\leq i,j\leq m}\{|\beta_{ij}|\}. \end{array}$$

By an argument similar to the one used in Proposition 4.11, we have that $\|\mathcal{M}[i][j]\| \leq 2m \cdot \mu$, where μ is defined as $\mu = \min(\max_{1\leq i,j\leq m}\{|\alpha_{ij}|\}, \max_{1\leq i,j\leq m}\{|\beta_{ij}|\})$. Therefore, each call to MinTerms takes at most $\mathcal{O}(m^3 \cdot \mu^3)$ time. Since the classical Floyd-Warshall algorithm (i.e. Algorithm 3 in which we consider that MinTerms needs constant time) runs in time $\mathcal{O}(m^3)$, we obtain the result. □

MaxConsistent. Given a difference bounds relation R, integers $b \geq 0, c > 0$ such that R^{b+2c} is consistent, and a matrix $\Lambda \in \mathbb{Z}_\infty^{2N\times 2N}$, let us denote $M_{R,b,\Lambda} = k \cdot \Lambda + \sigma(R^b) \in \mathbb{Z}[k]_\infty^{2N\times 2N}$. With this notation, we have:

$$\textsc{MaxConsistent}(R, b, \Lambda) = \sup\{n \in \mathbb{N} \mid M_{R,b,\Lambda}[n] \text{ is consistent}\}.$$

Since R^{b+2c} is consistent, it follows that $\textsc{MaxConsistent}(R, b, \Lambda) > 2$ and hence, we can define $\textsc{MaxConsistent}(R, b, \Lambda)$ equivalently as

$$\textsc{MaxConsistent}(R, b, \Lambda) = \inf\{n \in \mathbb{N} \mid M_{R,b,\Lambda}[n] \text{ is inconsistent}\} - 1.$$

In analogy to the non-parametric case, the inconsistency of a parametric difference bounds constraint amounts to the existence of a strictly negative elementary cycle in the constraint graph corresponding to $M_{R,b,\Lambda}[n]$ for some valuation $n \in \mathbb{N}$ of k. The MaxConsistent procedure can be implemented as follows. Let

$$\mathcal{M} = \textsc{ParametricFW}(M_{R,b,\Lambda})$$

as returned by Algorithm 3. Clearly $M_{R,b,\Lambda}[n]$ is inconsistent if and only if $\min(\mathcal{M}_{ii}[n]) < 0$ for some $i = 1, \ldots, 2N$. The minimal value of n for which this is the case is $K' = \min\{\Gamma(\mathcal{M}_{ii})\}_{i=1}^{2N}$, where Γ is a constant defined in the following lemma. Then, MaxPeriodic returns integer K defined as $K = K' - 1$.

Lemma 4.13 *Let $T = \{\alpha_i \cdot k + \beta_i\}_{i=1}^m$ be a set of univariate linear terms and $\ell \in \mathbb{N}$ be a constant. Then $\min_{n \geq \ell} T(n) < 0$ if and only if there exists $1 \leq i \leq m$ such that either $\alpha_i < 0$, or $\alpha_i \cdot \ell + \beta_i < 0$. Moreover the smallest value n such that $\min_{n \geq \ell}\{\alpha_i \cdot n + \beta_i\}_{i=1}^m < 0$ is $\Gamma(T) = \min_{j=1}^m \gamma_j$ where:*

$$\gamma_j = \begin{cases} \max(\ell, \lfloor -\frac{\beta_j}{\alpha_j} \rfloor + 1) & \text{if } \alpha_j < 0 \\ \ell & \text{if } \alpha_j \geq 0 \ \wedge \ \alpha_j \cdot \ell + \beta_j < 0 \\ \infty & \text{otherwise} \end{cases}$$

Proof: $\min_{n \geq \ell} T(n) < 0$ iff there exists $1 \leq i \leq m$ such that $\alpha_i \cdot n + \beta_i < 0$. Let us fix i for the rest of the proof. There are three cases:

- if $\alpha_i < 0$ we have $n \geq \lfloor -\frac{b_i}{\alpha_i} \rfloor + 1$, hence $n \geq \gamma_i = \max(\ell, \lfloor -\frac{b_i}{\alpha_i} \rfloor + 1)$.
- if $\alpha_i \geq 0$ and $\alpha_i \cdot \ell + \beta_i \geq 0$, we have $\alpha_i \cdot n + \beta_i \geq 0$, for all $n \geq \ell$, contradiction.
- else, we have $\alpha_i \geq 0$ and $\alpha_i \cdot \ell + \beta_i < 0$, in which case we have $n \geq \gamma_i = \ell$.

□

Proposition 4.14 *For a difference bounds relation R, integers $b \geq 0, c > 0$ such that R^b is consistent and a matrix $\Lambda \in \mathbb{Z}_\infty^{2N \times 2N}$,* MaxConsistent*$(R, b, \Lambda)$ runs in time at most $\mathcal{O}((b+c)^3 \cdot \|R\|^3 \cdot N^9)$.*

Proof: Computing $\mathcal{M}$ requires one application of Algorithm 3. By Proposition 4.12, the call to Algorithm 3 requires time at most $\mathcal{O}(\mu^3 \cdot N^6)$, where:

$$\mu = \min(\max_{1 \leq i,j \leq 2N}\{|\Lambda_{ij}|\}, \max_{1 \leq i,j \leq 2N}\{|(\sigma(R^b))_{ij}|\})$$

Since the constraint graph $\mathcal{G}_R^b$ has $(b+1) \cdot N$ nodes, any minimal path between extremes may not exceed weight $(b+1) \cdot N \cdot \|R\|$. This is because R^b is consistent, i.e. there are no negative cycles in $\mathcal{G}_R^b$, and a path going through a positive cycle is not minimal. Since the rate Λ is computed as $\Lambda = \sigma(R^{b+c}) - \sigma(R^b)$, we similarly infer that $\Lambda_{ij} \leq (b+c+1) \cdot N \cdot \|R\|$ for all $1 \leq i, j \leq 2N$. Hence $\mu \leq (b+c+1) \cdot N \cdot \|R\|$, which gives the result. □

MaxPeriodic. Given a difference bounds relation R, integers $K \in \mathbb{N}_\infty$, $b \geq 0$ and $c > 0$, such that R^b is consistent, and a matrix $\Lambda \in \mathbb{Z}_\infty^{2N \times 2N}$, the procedure

$$\textsc{MaxPeriodic}(R, b, \Lambda, c, K)$$

returns the maximal integer[3] $0 \leq n \leq K$ such that:

$$\forall 0 \leq \ell < n \ . \ \rho(\ell \cdot \Lambda + \sigma(R^b)) \circ R^c \Leftrightarrow \rho((\ell+1) \cdot \Lambda + \sigma(R^b))$$

[3]The successful test at line 8 of Algorithm 1 implies that $n \geq 2$.

or ∞, if $K = \infty$ and the above equivalence holds for all ℓ. The left-hand side of the equivalence can be encoded by a matrix of terms of the form $\min\{t_i\}_{i=1}^m$, where t_i are univariate linear terms, and can be computed by Algorithm 3. The DBM corresponding to the right-hand side is shown to be closed for all valuations of k, which means that the relation on the right-hand side of the equivalence can be defined simply by a parametric DBM, instead of a matrix of min-terms, which is the case for the left-hand side.

Lemma 4.15 *Let $R \in \mathcal{R}_{db}$ be a difference bounds relation, and Λ be the rate of the periodic sequence $\{\sigma(R^i)\}_{i=0}^{\infty}$. Then, for all $b \geq 0$ and $n > 0$, the DBM $n \cdot \Lambda + \sigma(R^b)$ is closed.*

Proof: A direct consequence of the fact that $\sigma(R^b)$ is closed by definition, and that Λ is also closed, by Corollary 4.10. □

We need thus to check equivalence (for all $k \geq 0$) between a matrix of minima of sets of linear terms in k and a parametric DBM. By Proposition 2.8, equivalence of two difference bounds constraints amounts to the equality of their closed DBMs. In order to find the maximal interval $0, \ldots, n$ in which $\min\{\alpha_i \cdot k + \beta_i\}_{i=1}^m = \alpha_0 \cdot k + \beta_0$ holds, for all $k = 0, \ldots, n$, we apply the following lemma to each entry in the left and right-hand side of the above equivalence, and return the minimal value among all entries, for which the equivalence holds, incremented by one[4].

Lemma 4.16 *Let $T = \{\alpha_i \cdot k + \beta_i\}_{i=1}^m$ be a set of univariate linear terms, $t_0 = \alpha_0 \cdot k + \beta_0 \in \mathbb{Z}[k]$ be a term, and $\ell \in \mathbb{N}$ be a constant. Then there exists an integer $\kappa > \ell + 1$ such that $\min T(n) = t_0(n)$, for all $\ell \leq n \leq \kappa$, if and only if the following hold:*

1. $\bigvee_{i=1}^m (\alpha_i = \alpha_0 \wedge \beta_i = \beta_0) \ \wedge \ \bigwedge_{i=1}^m \bigwedge_{j=0}^2 [\alpha_0 \cdot (\ell + j) + \beta_0 \leq \alpha_i \cdot (\ell + j) + \beta_i]$
2. $\kappa \leq \min\{\lfloor \frac{\beta_i - \beta_0}{\alpha_0 - \alpha_i} \rfloor \mid 1 \leq i \leq m, \ \alpha_0 \neq \alpha_i, \ \lfloor \frac{\beta_i - \beta_0}{\alpha_0 - \alpha_i} \rfloor > \ell + 1\}$

Proof: We assume without loss of generality that $m \geq 2$ and that all terms $\alpha_i \cdot k + \beta_i$, $i = 1, \ldots, m$ are distinct.
"⇒" If $\min T(n) = t_0(n)$, for all $\ell \leq n \leq \kappa$ and $\kappa > \ell + 1$, then clearly $\bigwedge_{i=1}^m \bigwedge_{j=0}^2 [\alpha_0 \cdot (\ell + j) + \beta_0 \leq \alpha_i \cdot (\ell + j) + \beta_i]$, i.e. the second conjunct of the first point is valid. To show the validity of the first conjunct of the first point, suppose without loss of generality that:

$$\begin{array}{rcccl} t_0(\ell) & = & t_1(\ell) & \leq & t_2(\ell) \\ t_0(\ell+1) & = & t_2(\ell+1) & \leq & t_1(\ell+1) \\ t_0(\ell+2) & < & t_1(\ell+2) & & \\ t_0(\ell+2) & < & t_2(\ell+2) & & \end{array}$$

The choice of t_1 and t_2 is not important. We obtain a contradiction in the following way:

$$\begin{array}{rcccl} \ell & = & \frac{\beta_1 - \beta_0}{\alpha_0 - \alpha_1} & \geq & \frac{\beta_2 - \beta_0}{\alpha_0 - \alpha_2} \\ \ell + 1 & = & \frac{\beta_2 - \beta_0}{\alpha_0 - \alpha_2} & \geq & \frac{\beta_1 - \beta_0}{\alpha_0 - \alpha_1} \end{array}$$

[4]Since $\forall 0 \leq \ell \leq \kappa \ . \ \phi$ if and only if $\forall 0 \leq \ell < \kappa + 1 \ . \ \phi$.

To show the second point, assume by contradiction that $\kappa > \lfloor \frac{\beta_i - \beta_0}{\alpha_0 - \alpha_i} \rfloor > \ell + 1$ for some $i = 1, \ldots, m$, such that the term $\alpha_i \cdot k + \beta_i$ is distinct from $\alpha_0 \cdot k + \beta_0$. Since, by the first point, we have $\beta_0 \leq \beta_i$, and $\frac{\beta_i - \beta_0}{\alpha_0 - \alpha_i} > 0$, then $\alpha_0 > \alpha_i$. It follows that $\alpha_0 \cdot \kappa > \alpha_i \cdot \kappa + \beta_i$, which contradicts the fact that t_0 is minimal in the interval $\ell, \ldots, \kappa$.
"$\Leftarrow$" By the first point, $t_0(n) = t_i(n)$, for all n, and $\min T(n) = t_0(n)$, for $n = \ell, \ell+1, \ell+2$. To prove that $\min T(n) = t_0(n)$ for all $\ell \leq n \leq \kappa$, assume by contradiction, that $\min T(p) = t_i(p) < t_0(p)$ for some $p = \ell, \ldots, \kappa$ and some $i = 1, \ldots, m$, such that t_i is distinct from t_0. But then we have $(\alpha_i - \alpha_0) \cdot p < \beta_0 - \beta_i \leq 0$. The last inequality is due to the first point. Since $p > 0$, we have that $\alpha_i < \alpha_0$, hence $\kappa \geq p > \lfloor \frac{\beta_i - \beta_0}{\alpha_0 - \alpha_i} \rfloor$, contradiction. □

Proposition 4.17 *For a difference bounds relation R, and integers $b \geq 0, c > 0$ such that R^{b+c} is consistent and a matrix $\Lambda \in \mathbb{Z}_\infty^{2N \times 2N}$,* MaxPeriodic *runs in time at most $\mathcal{O}((b+c)^3 \cdot \|R\|^3 \cdot N^9)$.*

Proof: We apply Algorithm 3 to compose the parametric DBM $k \cdot \Lambda + \sigma(R^b)$ with $\sigma(R^c)$, which requires time $\mathcal{O}(\mu^3 \cdot N^6)$, cf. Proposition 4.12. By an argument similar to the one used in the proof of Proposition 4.14, we obtain $\mu \leq (b + c + 1) \cdot N \cdot \|R\|$. The result of MaxPeriodic is the κ bound from Lemma 4.16, which can be established during the computation of the min-sets using Algorithm 3. Hence the result follows. □

Finally, we prove the asymptotic complexity on the running of Algorithm 1 for a difference bounds relation R in terms of its prefix, period, the number of variables used to define R, and the sum of absolute values of coefficients of R.

Theorem 4.18 *Let R be a difference bounds relation with prefix B and period C. Then, Algorithm 1 computes the transitive closure of R in at most $\mathcal{O}((B+C)^8 \cdot \|R\|^3 \cdot N^9)$ time.*

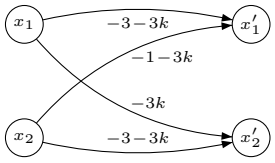
Proof: By Theorem 3.16, Algorithm 1 takes at most $\mathcal{O}((B + C)^2)$ iterations of the main loop and in each iteration and moreover, the algorithm considers a prefix and period candidates b and c such that both b and c are bounded by $\mathcal{O}((B + C)^2)$. By Proposition 4.14, Procedure MaxConsistent runs in time at most $\mathcal{O}((b+c)^3 \cdot \|R\|^3 \cdot N^9)$. Combining this bound with the bound on b and c, if follows that MaxConsistent runs in time at most $\mathcal{O}((B + C)^6 \cdot \|R\|^3 \cdot N^9)$. We obtain the same bound on running time of MaxPeriodic, by Proposition 4.17. The test on line 8 can be performed in $\mathcal{O}(N^2)$ time, by Proposition 2.8. The greatest power of a relation that is computed by the algorithm is R^{b+2c}. Since the composition of difference bounds relations can be computed in $\mathcal{O}(N^3)$ time, if follows that these computations are performed in $\mathcal{O}((B+C) \cdot N^3)$ time. Since the algorithm takes at most $\mathcal{O}((B + C)^2)$ iterations, we finally infer that the total running time of Algorithm 1 is bounded by $\mathcal{O}((B + C)^8 \cdot \|R\|^3 \cdot N^9)$. □

$$M_{R^b} = M_{R^c} = \begin{pmatrix} 0 & \infty & -3 & 0 \\ \infty & 0 & -1 & -3 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix} M_{R^{b+c}} = \begin{pmatrix} 0 & \infty & -6 & 3 \\ \infty & 0 & -4 & -6 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix} \Lambda = \begin{pmatrix} 0 & \infty & -3 & -3 \\ \infty & 0 & -3 & -3 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

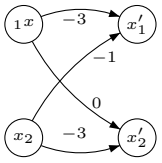
Figure 4.1: Candidate rate Λ for $(b, c) = (2, 2)$.

$$\begin{pmatrix} 0 & \infty & (-3-3k) & -3k \\ \infty & 0 & (-1-3k) & (-3-3k) \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

(a) $k \cdot \Lambda + M_{R^b}$



(b) $\mathcal{G}_{\sigma(k\cdot\Lambda+M_{R^b})}$



(c) $\mathcal{G}_{R^c}$

Figure 4.2: Left-hand side of the MaxPeriodic equivalence test

Running Example. We demonstrate the main steps of Algorithm 1 applied to the difference bounds relation $R \Leftrightarrow x_1 - x'_1 \leq 1 \wedge x_1 - x'_2 \leq -1 \wedge x_2 - x'_1 \leq -2 \wedge x_2 - x'_2 \leq 2$. The first valid guess for $(b, c) = (2, 2)$, for which the test on line 9 succeeds, leads to the candidate rate Λ (Figure 4.1). The MaxConsistent procedure first computes the parametric DBM corresponding to $\rho(k \cdot \Lambda + \sigma(R^b))$, shown in Figure 4.2a. The DBM is already closed and thus, application of Algorithm 1 doesn't change its entries. Next, Lemma 4.13 is applied to compute the value $K = \infty$ that MaxConsistent returns.

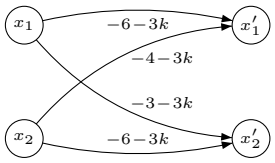
The MaxPeriodic procedure checks that

$$\rho(k \cdot \Lambda + \sigma(R^b)) \circ R^c \Leftrightarrow \rho((k+1) \cdot \Lambda + \sigma(R^b))$$

for all $k \geq 0$. The parametric DBM $\rho((k+1) \cdot \Lambda + \sigma(R^b))$ representing the right-hand side of the equivalence is shown in Figure 4.3. The left-hand side is equivalent to the composition of $\sigma(k \cdot \Lambda + M_{R^b})$ (Figures 4.2a and 4.2b) with R^c (Figure 4.2c). This amounts to the computation of shortest paths between extremal vertices of the graph in Figure 4.4 which results in a graph identical to the one in Figure 4.3. Since this graph represents the right-hand side, the above equivalence holds for all $k \geq 0$ and thus, MaxPeriodic returns $L = \infty$.

$$\begin{pmatrix} 0 & \infty & (-6-3k) & (-3-3k) \\ \infty & 0 & (-4-3k) & (-6-3k) \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

(a) $(k+1) \cdot \Lambda + M_{R^b}$



(b) $\mathcal{G}_{\sigma((k+1)\cdot\Lambda+M_{R^b})}$

Figure 4.3: Right-hand side of the MaxPeriodic equivalence test

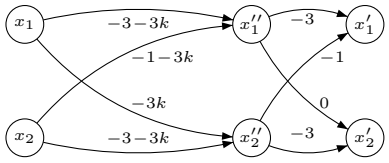


Figure 4.4: Computing parametric composition $\sigma(k \cdot \Lambda + M_{R^b}) \circ R^c$.

Then, a test on line 12 succeeds and Algorithm 1 returns the transitive closure:

$$\begin{aligned} R^+ \;\Leftrightarrow\; & \bigvee_{i=1}^{b-1} R^i \;\vee\; \exists k \geq 0 \,.\, \bigvee_{i=0}^{c-1} \pi(k \cdot \Lambda + \sigma(R^2)) \circ R^i \\ \Leftrightarrow\; & (x_1 - x_1' \leq 1 \wedge x_1 - x_2' \leq -1 \wedge x_2 - x_1' \leq -2 \wedge x_2 - x_2' \leq -2) \vee \exists k \geq 0 \,. \\ & (x_1 - x_1' \leq -3k-3 \wedge x_1 - x_2' \leq -3k \wedge x_2 - x_1' \leq -3k-1 \wedge x_2 - x_2' \leq -3k-3) \vee \\ & (x_1 - x_1' \leq -3k-2 \wedge x_1 - x_2' \leq -3k-4 \wedge x_2 - x_1' \leq -3k-5 \wedge x_2 - x_2' \leq -3k-2) \end{aligned}$$

After the elimination of the existential quantifier, we obtain:

$$\begin{aligned} R^+ \;\Leftrightarrow\; & (x_1 - x_1' \leq 1 \wedge x_1 - x_2' \leq -1 \wedge x_2 - x_1' \leq -2 \wedge x_2 - x_2' \leq -2) \vee \\ & (x_1 - x_1' \leq -3 \wedge x_1 - x_2' \leq 0 \wedge x_2 - x_1' \leq -1 \wedge x_2 - x_2' \leq -3) \vee \\ & (x_1 - x_1' \leq -2 \wedge x_1 - x_2' \leq -4 \wedge x_2 - x_1' \leq -5 \wedge x_2 - x_2' \leq -2) \end{aligned}$$

4.3 Octagonal Relations

Recall from Section 2.4 that an octagonal relation $R \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$ defined by a formula $R(\mathbf{x}, \mathbf{x}')$ can be represented as a difference bounds relation $\overline{R}(\mathbf{y}, \mathbf{y}')$ defined over the dual set of variables with the convention that y_{2i-1} stands for x_i and y_{2i} for $-x_i$. Then, an octagonal relation R can be represented by a difference bounds matrix $M_{\overline{R}}$. Similarly, for each DBM M, there is a corresponding octagonal relation Ω_M. Furthermore, octagonal relations can be represented canonically by DBMs $M^t_{\overline{R}}$.

We start by defining the mappings between octagonal relations and their matrix encodings required by Definition 3.6. Given a consistent octagonal relation $R(\mathbf{x}, \mathbf{x}')$ let $\sigma(R) = M^t_{\overline{R}}$. Dually, for any coherent DBM $M \in \mathbb{Z}_\infty^{4N \times 4N}$, let $\rho(M) = \Omega_M$. Clearly, $\rho(\sigma(R)) \Leftrightarrow R$, as required by Definition 3.6. In order to define the mapping π, we first define the class of *parametric* octagonal relations.

Definition 4.19 *A formula $\phi(\mathbf{x}, k)$ is a* parametric octagonal constraint *if it is equivalent to a finite conjunction of terms of the form $x_i - x_j \leq u_{ij}$, $x_i + x_j \leq v_{ij}$ or $x_i + x_j \geq t_{ij}$, where u_{ij}, v_{ij} and t_{ij} are univariate linear terms in k, for all $1 \leq i, j \leq N$.*

A parametric octagon $\phi(\mathbf{x}, k)$ is represented by a matrix $M_{\overline{\phi}}[k] \in \mathbb{Z}[k]_\infty^{2N \times 2N}$ of univariate linear terms. Vice versa, a matrix $M[k] \in \mathbb{Z}[k]_\infty^{2N \times 2N}$ encodes a parametric octagonal relation, denoted as $\Omega_M(k)$. With these considerations, we define $\pi(M[k]) = \Omega_M(k)$ to be the parametric counterpart of the ρ function from Definition 3.6.

4.3.1 Proving Periodicity

In order to prove that the class $\mathcal{R}_{oct}$ of octagonal relations is periodic, we need to prove that the sequence $\{\sigma(R^m)\}_{m=0}^{\infty}$ is periodic, for an arbitrary relation $R \in \mathcal{R}_{oct}$. It is sufficient to consider only the case where R is $*$-consistent i.e., $\sigma(R^m) = M^t_{\overline{R^m}}$, for all $m \geq 0$. We rely in the following on Theorem 2.26 which gives a method to compute $M^t_{\overline{R^m}}$, the tightly closed DBM representation of R^m, from $M^*_{\overline{R}^m}$, the closed DBM representation of $\overline{R}^m$.

We have previously shown, in Section 4.2, that difference bounds relations are periodic. In particular, this means that the sequence $\{M^*_{\overline{R}^m}\}_{m=0}^{\infty}$, corresponding to the iteration of the difference bounds relation $\overline{R}$, is periodic. To prove that the sequence $\{M^t_{\overline{R^m}}\}_{m=0}^{\infty}$ is also periodic, it is sufficient to show that (i) the minimum and (ii) the sum of two periodic sequences are periodic, and also that (iii) the integer half of a periodic sequence is also periodic.

Lemma 4.20 *Let $\{s_m\}_{m=0}^{\infty}$ and $\{t_m\}_{m=0}^{\infty}$ be two periodic sequences. Then the sequences $\{\min(s_m, t_m)\}_{m=0}^{\infty}$, $\{s_m + t_m\}_{m=0}^{\infty}$ and $\left\{\left\lfloor \frac{s_m}{2} \right\rfloor\right\}_{m=0}^{\infty}$ are periodic as well. Let b_s (c_s) be the prefix (period) of $\{s_m\}_{m=0}^{\infty}$, let b_t (c_t) be the prefix (period) of $\{t_m\}_{m=0}^{\infty}$, and let define $b = \max(b_s, b_t)$, $c = \mathrm{lcm}(c_s, c_t)$, and $b_m = b + \max_{i=0}^{c-1} K_i c$, where*

$$K_i = \begin{cases} \left\lceil \frac{s_{b+i} - t_{b+i}}{\lambda_i^{(t)} - \lambda_i^{(s)}} \right\rceil & \text{if } \lambda_i^{(s)} < \lambda_i^{(t)} \text{ and } t_{b+i} < s_{b+i} \\ \left\lceil \frac{t_{b+i} - s_{b+i}}{\lambda_i^{(s)} - \lambda_i^{(t)}} \right\rceil & \text{if } \lambda_i^{(t)} < \lambda_i^{(s)} \text{ and } s_{b+i} < t_{b+i} \\ 0 & \text{otherwise} \end{cases}$$

for each $i = 0, \ldots, c-1$ and where $\lambda_0^{(s)}, ..., \lambda_{c-1}^{(s)}$ ($\lambda_0^{(t)}, ..., \lambda_{c-1}^{(t)}$) are rates of $\{s_m\}_{m=0}^{\infty}$ ($\{t_m\}_{m=0}^{\infty}$) with respect to the common prefix b and period c. Then, the prefix and the period of the above sequences are:

	prefix	*period*
$\{s_m + t_m\}_{m=0}^{\infty}$	b	c
$\left\{\left\lfloor \frac{s_m}{2} \right\rfloor\right\}_{m=0}^{\infty}$	b	$2c$
$\{\min(s_m, t_m)\}_{m=0}^{\infty}$	b_m	c

Proof: We can show that the sum sequence $\{s_m + t_m\}_{m=0}^{\infty}$ is periodic as well, with prefix b, period c and rates $\lambda_0^{(s)} + \lambda_0^{(t)}, ..., \lambda_{c-1}^{(s)} + \lambda_{c-1}^{(t)}$. In fact, for every $k \geq 0$ and $i = 0, ..., c-1$ we have successively:

$$\begin{aligned} (s+t)_{b+(k+1)c+i} &= s_{b+(k+1)c+i} + t_{b+(k+1)c+i} && (4.3) \\ &= \lambda_i^{(s)} + s_{b+kc+i} + \lambda_i^{(t)} + t_{b+kc+i} && (4.4) \\ &= \lambda_i^{(s)} + \lambda_i^{(t)} + s_{b+kc+i} + t_{b+kc+i} && (4.5) \\ &= (\lambda_i^{(s)} + \lambda_i^{(t)}) + (s+t)_{b+kc+i} && (4.6) \end{aligned}$$

For the min sequence $\{\min(s_m, t_m)\}_{m=0}^{\infty}$, it can be shown that, for each $i = 0, ..., c-1$ precisely one of the following assertions hold:

1. $(\lambda_i^{(s)} < \lambda_i^{(t)}$ or $\lambda_i^{(s)} = \lambda_i^{(t)}$ and $s_{b+i} < t_{b+i})$ and $\forall k \geq 0$. $s_{b+K_ic+kc+i} \leq t_{b+K_ic+kc+i}$
2. $(\lambda_i^{(t)} < \lambda_i^{(s)}$ or $\lambda_i^{(s)} = \lambda_i^{(t)}$ and $t_{b+i} < s_{b+i})$ and $\forall k \geq 0$. $t_{b+K_ic+kc+i} \leq s_{b+K_ic+kc+i}$

Intuitively, starting from the position $b+K_ic$, on every period c, the minimum amongst the two sequences is always defined by the same sequence i.e., the one having the minimal rate on index i, or if the rates are equal, the one having the smaller starting value.

We can show now that the min sequence $\{\min(s_m, t_m)\}_{m=0}^{\infty}$ is periodic starting at $b_m = b + \max_{i=0}^{c-1} K_ic$, with period c and rates $\min(\lambda_0^{(s)}, \lambda_0^{(t)}), ..., \min(\lambda_{c-1}^{(s)}, \lambda_{c-1}^{(t)})$. That is, we have successively, for every $k \geq 0$ and $i = 0, ..., c-1$, and whenever i satisfies the condition (1) above (the case when i satisfies the condition (2) being similar):

$$\begin{aligned}
\min(s_{b_m+(k+1)c+i}, t_{b_m+(k+1)c+i}) &= s_{b_m+(k+1)c+i} \\
&= \lambda_i^{(s)} + s_{b_m+kc+i} \\
&= \min(\lambda_i^{(s)}, \lambda_i^{(t)}) + \min(s_{b_m+kc+i}, t_{b_m+kc+i})
\end{aligned}$$

For the sequence $\left\{\left\lfloor \frac{s_m}{2} \right\rfloor\right\}_{m=0}^{\infty}$, assume that the sequence $\{s_m\}_{m=0}^{\infty}$ is periodic with prefix b, period c and rates $\lambda_0, ..., \lambda_{c-1}$. It can be easily shown that the sequence $\left\lfloor \frac{s_m}{2} \right\rfloor$ is periodic as well with prefix b, period $2c$, and rates $\lambda_0, ..., \lambda_{c-1}, \lambda_0, ..., \lambda_{c-1}$.

We have successively for any $k \geq 0$, and for any $i = 0, ..., c-1$:

$$\left\lfloor \frac{s_{b+(k+1)2c+i}}{2} \right\rfloor = \left\lfloor \frac{2\lambda_i + s_{b+k\cdot 2c+i}}{2} \right\rfloor = \lambda_i + \left\lfloor \frac{s_{b+k\cdot 2c+i}}{2} \right\rfloor$$

Similarly, for any $k \geq 0$ and for any $i = 0, ..., c-1$, we have:

$$\left\lfloor \frac{s_{(b+k+1)2c+c+i}}{2} \right\rfloor = \left\lfloor \frac{2\lambda_i + s_{b+k\cdot 2c+c+i}}{2} \right\rfloor = \lambda_i + \left\lfloor \frac{s_{b+k\cdot 2c+c+i}}{2} \right\rfloor$$

□

The theorem below is an immediate consequence of Theorem 2.26 and Lemma 4.20.

Theorem 4.21 *The class of octagonal relations is periodic.*

Corollary 4.22 *If $R \in \mathcal{R}_{oct}$ is an octagonal relation, the rate Λ of the periodic sequence $\{\sigma(R^i)\}_{i=0}^{\infty}$ is tightly closed.*

Proof: On one hand, $\sigma(R^i) = M_{R^i}^t$ is tightly closed, by definition. By Theorem 4.21, there exist $b \geq 0$, $c > 0$ and $\Lambda \in \mathbb{Z}_{\infty}^{4N \times 4N}$, for all $n \geq 0$:

$$\sigma(R^{nc+b}) = n \cdot \Lambda + \sigma(R^b)$$

The first two points of Definition 2.19 are immediate. The closure (point 3 of Definition 2.19) is by Corollary 4.10. We are left with proving the last point, namely that for all $1 \leq i, j \leq 4N$:

$$\Lambda_{ij} \leq \lfloor \frac{\Lambda_{i\bar{i}}}{2} \rfloor + \lfloor \frac{\Lambda_{\bar{j}j}}{2} \rfloor \tag{4.7}$$

Since $\sigma(R^{nc+b})$ is tightly closed, for all $n \geq 0$, we have, for all $1 \leq i, j \leq 4N$:

$$\begin{aligned} n \cdot \Lambda_{ij} + \sigma(R^b)_{ij} &\leq \lfloor \tfrac{n \cdot \Lambda_{i\bar{i}} + \sigma(R^b)_{i\bar{i}}}{2} \rfloor + \lfloor \tfrac{n \cdot \Lambda_{\bar{j}j} + \sigma(R^b)_{\bar{j}j}}{2} \rfloor \\ &\leq n \cdot (\lfloor \tfrac{\Lambda_{i\bar{i}}}{2} \rfloor + \lfloor \tfrac{\Lambda_{\bar{j}j}}{2} \rfloor) + \lfloor \tfrac{\sigma(R^b)_{i\bar{i}}}{2} \rfloor + \lfloor \tfrac{\sigma(R^b)_{\bar{j}j}}{2} \rfloor + 2 \end{aligned}$$

The last inequality holds because, for all $x, y \in \mathbb{Z}$ and $n \geq 0$:

- $\lfloor \frac{x+y}{2} \rfloor \leq \lfloor \frac{x}{2} \rfloor + \lfloor \frac{y}{2} \rfloor + 1$,
- $\lfloor \frac{n \cdot x}{2} \rfloor = n \cdot \lfloor \frac{x}{2} \rfloor$ if x is even,

and $\Lambda_{i\bar{i}}$ and $\Lambda_{\bar{j}j}$ are both even. We calculate:

$$n \cdot (\Lambda_{ij} - \lfloor \frac{\Lambda_{i\bar{i}}}{2} \rfloor - \lfloor \frac{\Lambda_{\bar{j}j}}{2} \rfloor) \leq \lfloor \frac{\sigma(R^b)_{i\bar{i}}}{2} \rfloor + \lfloor \frac{\sigma(R^b)_{\bar{j}j}}{2} \rfloor - \sigma(R^b)_{ij} + 2, \ \forall n \geq 0$$

Then the condition (4.7) follows. □

4.3.2 Checking $*$-consistency and Periodicity

Similar to the case of difference bounds relations, in this section we give efficient ways to implement the MaxConsistent and MaxPeriodic procedures from Algorithm 1. In the rest of this section, *a univariate linear half-term* is a term of the form $\lfloor \frac{\alpha \cdot k + \beta}{2} \rfloor$, where the mapping $x \mapsto \lfloor \frac{x}{2} \rfloor$ denotes the integer division by two.

Unlike the case of octagonal constraints with constants coefficients, the matrices representing parametric octagons do not have a tightly closed canonical form. To overcome this problem, one can use Algorithm 3 and Theorem 2.20 to define the tight closure of a parametric octagonal matrix as a matrix whose entries are either ∞ or terms of the form $\min\{t_i(k)\}_{i=1}^m$, where $t_i(k)$ are either univariate linear terms or sums of half-terms.

MaxConsistent. Given an octagonal relation R, integers $b \geq 0, c > 0$ such that R^{b+2c} is consistent, and a matrix $\Lambda \in \mathbb{Z}_\infty^{4N \times 4N}$, let us denote $M_{R,b,\Lambda} = k \cdot \Lambda + \sigma(R^b) \in \mathbb{Z}[k]_\infty^{4N \times 4N}$. Similarly as in the difference bounds case, we have:

$$\begin{aligned} \textsc{MaxConsistent}(R, b, \Lambda) &= \sup\{n \in \mathbb{N} \mid M_{R,b,\Lambda}[n] \text{ is octagonal-consistent}\} \\ &= \inf\{n \in \mathbb{N} \mid M_{R,b,\Lambda}[n] \text{ is octagonal-inconsistent}\} - 1 \end{aligned}$$

According to Theorem 2.20, $M_{R,b,\Lambda}[n]$ is octagonal-inconsistent for some valuation $n \in \mathbb{N}$ of k, if either (i) $M[n]_{R,b,\Lambda}$ is inconsistent, or (ii) $\lfloor \frac{(M[n]^*_{R,b,\Lambda})_{i\bar{i}}}{2} \rfloor + \lfloor \frac{(M[n]^*_{R,b,\Lambda})_{\bar{i}i}}{2} \rfloor < 0$, for some $1 \leq i \leq 4N$.

Let $\mathcal{M} = \text{ParametricFW}(M_{R,b,\Lambda})$, as returned by Algorithm 3. Checking for the case (i) can be done in a similar way as for difference bounds constraints (Section 4.2). The condition of case (ii) is equivalent to the following:

$$\min [\![\{\lfloor \frac{t}{2} \rfloor + \lfloor \frac{u}{2} \rfloor \mid t \in \mathcal{M}_{i\bar{i}}, u \in \mathcal{M}_{\bar{i}i}\}]\!]_{\geq 0} < 0, \text{ for some } 1 \leq i \leq 4N \tag{4.8}$$

The following lemma shows that a set of sums of univariate half-terms is semantically equivalent to the union of two sets of univariate linear terms.

Lemma 4.23 *Let $T = \{t_i + \sum_{j=1}^{m_i} \lfloor \frac{u_{ij}}{2} \rfloor\}_{i=1}^{n}$, where $t_i = \alpha_i \cdot k + \beta_i, u_{ij} = \gamma_{ij} \cdot k + \delta_{ij}$ are univariate linear terms. Then there exist two sets $\mathcal{L}, \mathcal{U}$ of univariate linear terms such that for all $k \geq 0$:*

$$\begin{array}{rcl} \min\{\mathcal{L}(k)\} & = & \min\{T(2k)\} \\ \min\{\mathcal{U}(k)\} & = & \min\{T(2k+1)\} \end{array}$$

Moreover $\|\mathcal{L}\| \leq \|T\|$ and $\|\mathcal{U}\| \leq \|T\|$.

Proof: For a univariate linear term $\gamma \cdot k + \delta \in \mathbb{Z}[k]$, we have:

$$[\![\lfloor \frac{\gamma \cdot k + \delta}{2} \rfloor]\!]_{\geq 0} = [\![\gamma \cdot k + \lfloor \frac{\delta}{2} \rfloor]\!]_{\geq 0} \cup [\![\gamma \cdot k + \lfloor \frac{\gamma + \delta}{2} \rfloor]\!]_{\geq 0}.$$

Let

$$\begin{array}{rcl} \mathcal{L} & = & \{(2\alpha_i + \sum_{j=1}^{m_i} \gamma_{ij}) \cdot k + \beta_i + \sum_{j=1}^{m_i} \lfloor \frac{\delta_{ij}}{2} \rfloor\}_{i=1}^{n}, \text{ and} \\ \mathcal{U} & = & \{(2\alpha_i + \sum_{j=1}^{m_i} \gamma_{ij}) \cdot k + \alpha_i + \beta_i + \sum_{j=1}^{m_i} \lfloor \frac{\gamma_{ij} + \delta_{ij}}{2} \rfloor\}_{i=1}^{n}. \end{array}$$

Clearly $[\![T]\!]_{\geq 0} = [\![\mathcal{L}]\!]_{\geq 0} \cup [\![\mathcal{U}]\!]_{\geq 0}$, $\min\{\mathcal{L}(k)\} = \min\{T(2k)\}$, and $\min\{\mathcal{U}(k)\} = \min\{T(2k+1)\}$ for all $k \geq 0$. It is easy to see that $\|\mathcal{L}\| \leq \|T\|$ and $\|\mathcal{U}\| \leq \|T\|$. □

Let us denote $\mathcal{T}_i = \{\lfloor \frac{t}{2} \rfloor + \lfloor \frac{u}{2} \rfloor \mid t \in \mathcal{M}_{i\bar{i}}, u \in \mathcal{M}_{\bar{i}i}\}$. Then, the condition (4.8) can be rewritten as

$$\min [\![\mathcal{T}_i]\!]_{\geq 0} < 0, \text{ for some } 1 \leq i \leq 4N.$$

By Lemma 4.23, for each $i = 1, \ldots, 4N$ there exist sets of univariate linear terms $\mathcal{L}_i$ and $\mathcal{U}_i$ such that $\min\{\mathcal{L}_i(k)\} = \min\{\mathcal{T}_i(2k)\}$ and $\min\{\mathcal{U}_i(k)\} = \min\{\mathcal{T}_i(2k+1)\}$ for all $k \geq 0$. Therefore, for all $1 \leq i \leq 4N$, we have

$$(\min [\![\mathcal{T}_i]\!]_{\geq 0} < 0) \Leftrightarrow (\min [\![\mathcal{L}_i]\!]_{\geq 0} < 0) \vee (\min [\![\mathcal{U}_i]\!]_{\geq 0} < 0).$$

With these considerations, the MaxConsistent procedure can be implemented as follows. We define

$$\begin{array}{rcl} K_{db} & = & \min\{\Gamma(\mathcal{M}_{ii})\}_{i=1}^{4N}, \\ K_{\mathcal{L}} & = & \min\{\Gamma(\mathcal{L}_i)\}_{i=1}^{4N}, \\ K_{\mathcal{U}} & = & \min\{\Gamma(\mathcal{U}_i)\}_{i=1}^{4N} \end{array}$$

where given a set of univariate linear terms $\mathcal{T}$, $\Gamma(\mathcal{T})$ is a constant defined in the Lemma 4.13. Note that K_{db} is the minimal integer $n \in \mathbb{N}$ such that $M[n]_{R,b,\Lambda}$ is inconsistent. By Lemma 4.23, $\min\{2 \cdot K_{\mathcal{L}}, 2 \cdot K_{\mathcal{U}} - 1\}$ is the minimal integer for which (4.8) holds, or

equivalently, the minimal integer $n \in \mathbb{N}$ such that $\lfloor \frac{(M[n]^*_{R,b,\Lambda})_{i\bar{i}}}{2} \rfloor + \lfloor \frac{(M[n]^*_{R,b,\Lambda})_{\bar{i}i}}{2} \rfloor < 0$, for some $1 \leq i \leq 4N$. Consequently, the MaxConsistent procedure returns:

$$\textsc{MaxConsistent}(R, b, \Lambda) \quad = \quad \min\{K_{db}, 2 \cdot K_{\mathcal{L}}, 2 \cdot K_{\mathcal{U}} - 1\} - 1.$$

Proposition 4.24 *For an octagonal relation R, an integer $b \geq 0$ such that R^b is consistent and a matrix $\Lambda \in \mathbb{Z}_\infty^{4N \times 4N}$,* MaxConsistent *runs in time at most $\mathcal{O}((b+c)^3 \cdot \|R\|^3 \cdot N^9)$.*

Proof: Computing $\mathcal{M}$ requires one application of Algorithm 3. By Proposition 4.12, the call to Algorithm 3 requires time at most $\mathcal{O}(\mu^3 \cdot N^6)$, where:

$$\mu = \min(\max_{1 \leq i,j \leq 2N}\{|\Lambda_{ij}|\}, \max_{1 \leq i,j \leq 2N}\{|(\sigma(R^b))_{ij}|\})$$

Moreover, the size of $\mathcal{M}_{ij}$ is bounded by $8N \cdot \mu$, by Proposition 4.12. By an argument similar to the one in the proof of Proposition 4.12, one infers that $\mu \leq (b+c+1) \cdot 2N \cdot \|R\|$. Consequently, $\mathcal{M}$ takes at most $\mathcal{O}((b+c)^3 \cdot \|R\|^3 \cdot N^9)$ time and $\|\mathcal{M}_{ij}\|$ is bounded by $\mathcal{O}((b+c) \cdot \|R\| \cdot N^2)$.

Hence, computing $\mathcal{L}_i$ and $\mathcal{U}_i$ can be done in time at most $\mathcal{O}(N^2 \cdot (b+c) \cdot \|R\|)$. By Lemma 4.23, $\|\mathcal{L}_i\| \leq \|\mathcal{M}_{ij}\|$, and $\|\mathcal{U}_i\| \leq \|\mathcal{M}_{ij}\|$ and consequently, $K_{db}, K_{\mathcal{L}}$, an $K_{\mathcal{L}}$ can be computed in $\mathcal{O}((b+c) \cdot \|R\| \cdot N^3)$ time for each $1 \leq i \leq 4N$. Hence, MaxConsistent procedure runs in time at most $\mathcal{O}((b+c)^3 \cdot \|R\|^3 \cdot N^9)$. □

MaxPeriodic. Given an octagonal relation R, two integers $b \geq 0$ and $c > 0$, such that R^b is consistent, a constant $K \in \mathbb{N}_\infty$, and a matrix $\Lambda \in \mathbb{Z}_\infty^{4N \times 4N}$, the procedure $\textsc{MaxPeriodic}(R, b, \Lambda, c, K)$ returns the maximal positive integer $n \leq K$ such that the equivalence $\rho(\ell \cdot \Lambda + \sigma(R^b)) \circ R^c \Leftrightarrow \rho((\ell+1) \cdot \Lambda + \sigma(R^b))$ holds for all $0 \leq \ell < n$, or ∞ if the above holds for all positive ℓ.

The left-hand side of the equivalence can be encoded by a matrix M_1 of terms of the form $\min\{t_i\}_{i=1}^m$, where t_i are either univariate linear terms or sums of univariate half-terms, and can be computed by Algorithm 3. We show that the DBM M_2 that encodes the right-hand side relation is tightly closed, for all valuation of k, meaning that the right-hand side can be simply represented by a parametric DBM, under the octagonal interpretation.

Lemma 4.25 *Let $R \in \mathcal{R}_{oct}$ be an octagonal relation, and Λ be the rate of the periodic sequence $\{\sigma(R^i)\}_{i=0}^\infty$. Then, for all $b, n \geq 0$, the DBM $(n+1) \cdot \Lambda + \sigma(R^b)$ is tightly closed.*

Proof: As a direct consequence of the fact that $\sigma(R^b)$ is tightly closed, by definition, and that Λ is tightly closed, by Corollary 4.22. □

Two octagonal relations are equivalent whenever their tightly closed DBM encodings are equal (Proposition 2.21). Hence we need to check for equality (inside some interval) between min-sets of univariate linear terms and sums of half-terms (the left-hand side

matrix M_1) and univariate linear terms (the right-hand side matrix M_2). Here again Lemma 4.23 comes to rescue. We compute M_1 using Algorithm 3 and Theorem 2.26. Using Lemma 4.23, we split M_1 to $M_{1,L}$ and $M_{1,U}$, where $M_{1,L}, M_{1,U}$ are matrices with sets of univariate terms as entries such that $\min\{(M_{1,L})_{ij}(k)\} = \min\{M_{ij}(2k)\}$ and $\min\{(M_{1,U})_{ij}(k)\} = \min\{M_{ij}(2k+1)\}$ for all $k \geq 0$. Similarly, we split M_2 to $M_{2,L}$ and $M_{2,U}$. Then, we apply Lemma 4.16 to compute the upper bound P_L (P_U) of the interval in which $M_{1,L}$ and $M_{2,L}$ ($M_{1,U}$ and $M_{2,U}$) are equal. Finally, the upper bound of the interval in which M_1 and M_2 are equal is computed as $\min\{2 \cdot P_L, 2 \cdot P_U - 1\} + 1$.

Proposition 4.26 *For a difference bounds relation R, and integers $b \geq 0, c > 0$ such that R^{b+c} is consistent and a matrix $\Lambda \in \mathbb{Z}_{\infty}^{2N \times 2N}$,* MaxPeriodic *runs in time at most $\mathcal{O}((b+c)^3 \cdot \|R\|^3 \cdot N^9)$.*

Proof: By an argument similar to the one used in the proof of Proposition 4.24, Algorithm 3 computes the matrix $\mathcal{M}$ of sets representing the parametric composition of $k \cdot \Lambda + \sigma(R^b)$ with $\sigma(R^c)$ in time $\mathcal{O}((b+c)^3 \cdot \|R\|^3 \cdot N^9)$. Moreover the size of each entry $\mathcal{M}_{ij}$ is bounded by $\mathcal{O}((b+c) \cdot \|R\| \cdot N^2)$. Computing the minimal bound from Lemma 4.16 requires then $\mathcal{O}((b+c) \cdot \|R\| \cdot N^4)$ time. Hence the result. □

Finally, we prove the asymptotic complexity on the running of Algorithm 1 for an octagonal relation R in terms of its prefix, period, the number of variables used to define R, and the sum of absolute values of coefficients of R.

Theorem 4.27 *Let R be an octagonal relation with prefix B and period C. Then, Algorithm 1 computes the transitive closure of R in at most $\mathcal{O}((B+C)^8 \cdot \|R\|^3 \cdot N^9)$ time.*

Proof: The bounds on the running time of procedures MaxConsistent (Propositions 4.24) and MaxPeriodic (Proposition 4.26) for octagonal relations are same as for difference bounds relations (Propositions 4.14 and 4.17, respectively). Similarly, relational composition of octagons has the same asymptotic bound $\mathcal{O}(N^3)$ as difference bounds constraints. Hence, we obtain the same bound as in Theorem 4.18. □

Running Example Consider the octagonal relation $R(x_1, x_2, x_1', x_2') \Leftrightarrow x_1 + x_2 \leq 5 \wedge x_1' - x_1 \leq -2 \wedge x_2' - x_2 \leq -3 \wedge x_2' - x_1' \leq 1$. The check at line 9 of Algorithm 1 succeeds for $(b, c) = (1, 1)$. In order to compute MaxPeriodic, one needs to compose parametric difference bound matrices, similarly as in the case of difference bound relations. Moreover, the tightening step must be performed. Figure 4.5 shows the parametric matrix M representing the left-hand side of the equivalence checked by MaxPeriodic. The matrix is closed, but not tightly closed. We illustrate the tightening for the constraint $y_3'' - y_4'$, which depends on constraints $y_3'' - y_4''$ and $y_4'' - y_4''$ and is thus computed as

$$\min\{-6k - 3, \lfloor \frac{-6k}{2} \rfloor + \lfloor \frac{-6k-6}{2} \rfloor\}$$

$$
\begin{array}{c}
 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_1'' \\ y_2'' \\ y_3'' \\ y_4'' \\ y_1' \\ y_2' \\ y_3' \\ y_4'
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc}
y_1 & y_2 & y_3 & y_4 & y_1'' & y_2'' & y_3'' & y_4'' & y_1' & y_2' & y_3' & y_4'
\end{array} \\
\left(\begin{array}{cccccccccccc}
0 & \infty & \infty & 5 & \infty & \infty & \infty & -3k+2 & \infty & \infty & \infty & -3k-1 \\
\infty & 0 & \infty & \infty & \infty & -2k-2 & \infty & -3k-1 & \infty & -2k-4 & \infty & -3k-4 \\
\infty & 5 & 0 & \infty & \infty & -2k+3 & \infty & -3k+4 & \infty & -2k+1 & \infty & -3k+1 \\
\infty & \infty & \infty & 0 & \infty & \infty & \infty & -3k-3 & \infty & \infty & \infty & -3k-6 \\
-2k-2 & \infty & \infty & -2k+3 & 0 & \infty & \infty & -5k & \infty & \infty & \infty & -5k-3 \\
\infty & \infty & \infty & \infty & \infty & 0 & \infty & 1 & \infty & -2 & \infty & -2 \\
-3k-1 & -3k+2 & -3k-3 & -3k+4 & 1 & -5k & 0 & -6k & \infty & -5k-2 & \infty & -6k-3 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & -3 \\
-2k-4 & \infty & \infty & -2k+1 & -2 & \infty & \infty & -5k-2 & 0 & \infty & \infty & -5k-5 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & 1 \\
-3k-4 & -3k-1 & -3k-6 & -3k+1 & -2 & -5k-3 & -3 & -6k-3 & 1 & -5k-5 & 0 & -6k-6 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0
\end{array}\right)
\end{array}
$$

Figure 4.5: Left-hand side before tightening

By Lemma 4.23, we obtain

$$
\begin{array}{lcllcl}
[\![-6k-3]\!]_{\geq 0} & = & L_1 \cup U_1 & = & [\![-12k-3]\!]_{\geq 0} & \cup \; [\![-12k-9]\!]_{\geq 0} \\
[\![\lfloor \frac{-6k}{2} \rfloor]\!]_{\geq 0} & = & L_2 \cup U_2 & = & [\![-6k]\!]_{\geq 0} & \cup \; [\![-6k-3]\!]_{\geq 0} \\
[\![\lfloor \frac{-6k-6}{2} \rfloor]\!]_{\geq 0} & = & L_3 \cup U_3 & = & [\![-6k-3]\!]_{\geq 0} & \cup \; [\![-6k-6]\!]_{\geq 0}
\end{array}
$$

The tightening step splits M into M_L and M_U. In M_L, the tightened constraint $y_3'' - y_4'$ is computed as

$$\min\{-12k-3, (-6k) + (-6k-3)\} = -12k-3$$

and similarly in M_U, the tightened constraint $y_3'' - y_4'$ is computed as

$$\min\{-12k-9, (-6k-3) + (-6k-6)\} = -12k-9$$

Further checks are similar as in the difference bounds case. Then, Algorithm 1 returns the following result:

$$
\begin{array}{rcl}
R^+ & \Leftrightarrow & \bigvee_{i=1}^{b-1} R^i \;\vee\; \exists k \geq 0 \,.\, \bigvee_{i=0}^{c-1} \pi(k \cdot \Lambda + \sigma(R)) \circ R^i \\
& \Leftrightarrow & \exists k \geq 0 \,.\, x_2' \leq -3k \wedge x_1' - x_1 \leq -2k-2 \wedge x_2' - x_1' \leq 1 \wedge x_2' - x_1 \leq -3k-1 \wedge \\
& & x_2' - x_2 \leq -3k-3 \wedge x_1 + x_2 \leq 5 \wedge x_1 + x_2' \leq -3k+2 \wedge \\
& & x_1' + x_2 \leq -2k+3 \wedge x_1' + x_2' \leq -5k \wedge x_2' + x_2 \leq -3k+4
\end{array}
$$

After quantifier elimination, we obtain:

$$
\begin{array}{rcl}
R^+ & \Leftrightarrow & x_2' \leq 0 \wedge x_1' - x_1 \leq -2 \wedge x_2' - x_1 \leq -1 \wedge x_2' - x_1' \leq 1 \wedge \\
& & x_2' - x_2 \leq -3 \wedge x_1 + x_2 \leq 5 \wedge x_1 + x_2' \leq 2 \wedge \\
& & x_1' + x_2 \leq 3 \wedge x_1' + x_2' \leq 0 \wedge x_2 + x_2' \leq 4
\end{array}
$$

□

4.4 Finite Monoid Affine Relations

Recall from Section 2.5 that an affine relation $R \in \mathbb{Z}^N \times \mathbb{Z}^N$ is defined by a linear arithmetic constraint of the form $\mathbf{x}' = A\mathbf{x} + \mathbf{b}$, where $A \in \mathbb{Z}^{N\times N}$ is a square matrix, and $\mathbf{b} \in \mathbb{Z}^N$ is a column vector. The relation is said to have the finite monoid property if the set $\{A^0, A^1, \ldots\}$ of matrix powers of A is finite.

It is easy to see that A is finite monoid if and only if there exists $p \geq 0$ and $l > 0$ such that $A^p = A^{p+l}$, i.e. $\mathcal{M}_A = \{A^0, \ldots, A^p, \ldots, A^{p+l-1}\}$. If A has the finite monoid property, it can be shown that the transitive closure of T can be defined in Presburger arithmetic [Boi99, FL02]. We achieve the same result below, by showing that the update of a finite monoid affine relations is a periodic relation. As a consequence, the closed form of the update can be computed by Algorithm 1. Since the update relation is $*$-consistent and deterministic, the transitive closure can be computed by applying the following lemma.

Lemma 4.28 *Let $R(\mathbf{x}, \mathbf{x}') \in \mathcal{R}$ be a $*$-consistent deterministic relation and $\varphi(\mathbf{x})$ be a guard. Then the transitive closure of the relation $R \wedge \varphi$ can be defined as:*

$$(R \wedge \varphi)^+(\mathbf{x}, \mathbf{x}') \Leftrightarrow \exists k > 0 \ . \ \widehat{R}(k, \mathbf{x}, \mathbf{x}') \wedge \forall 0 \leq \ell < k \ \exists \mathbf{y} \ . \ \widehat{R}(\ell, \mathbf{x}, \mathbf{y}) \wedge \varphi(\mathbf{y})$$

where $\widehat{R}$ defines the closed form of R.

Proof: "$\Rightarrow$" Let $\mathbf{v}, \mathbf{v}'$ be a pair of valuations of $\mathbf{x}$ and $\mathbf{x}'$, respectively, such that $\mathbf{v}, \mathbf{v}' \models (R \wedge \varphi)^+$. Then there exists $n > 0$ such that $\mathbf{v}, \mathbf{v}' \models (R \wedge \varphi)^n$. Consequently, there exists a sequence of valuations $\mathbf{v} = \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n = \mathbf{v}'$ such that $\mathbf{v}_i, \mathbf{v}_{i+1} \models R \wedge \varphi$. By Definition 3.9, we have that $\models \widehat{R}(n, \mathbf{v}_0, \mathbf{v}_n)$ and $\models \widehat{R}(i, \mathbf{v}_0, \mathbf{v}_i) \wedge \varphi(\mathbf{v}_i)$, for all $i = 0, \ldots, n-1$.
"$\Leftarrow$" Let $\mathbf{v}$ and $\mathbf{v}'$ be two valuations such that $\models \widehat{R}(n, \mathbf{v}, \mathbf{v}')$ for some $n > 0$ and for all $i = 0, \ldots, n-1$ we have $\models \widehat{R}(i, \mathbf{v}, \mathbf{v}_i)$ and $\models \varphi(\mathbf{v}_i)$, for some valuation $\mathbf{v}_i$ of $\mathbf{x}$. Since $\widehat{R}(n) \Leftrightarrow R^n$, by Definition 3.9, there exists a sequence of valuations $\mathbf{v} = \mathbf{v}'_0, \mathbf{v}'_1, \ldots, \mathbf{v}'_n = \mathbf{v}'$ such that $\mathbf{v}'_i, \mathbf{v}'_{i+1} \models R$. By the fact that R was assumed to be deterministic, we have $\mathbf{v}_i = \mathbf{v}'_i$ for all $i = 0, \ldots, n-1$, hence $\mathbf{v}'_i \models \varphi$, for all $i = 0, \ldots, n-1$. Clearly then $\mathbf{v}, \mathbf{v}' \models (R \wedge \varphi)^+$. □

To compute the transitive closure of an affine relation, it is enough to compute the closed form of its update. This can be computed by Algorithm 1 whenever the update relation is shown to be periodic. In the following, we show that this is indeed the case, when A has the finite monoid property. For simplicity reasons, we will work with the equivalent homogenous form of (2.3)

$$T_h \Leftrightarrow \mathbf{x}'_h = A_h \times \mathbf{x}_h \ \wedge \ \phi_h(\mathbf{x}_h) \quad \text{where} \quad A_h \Leftrightarrow \left(\begin{array}{c|c} A & \mathbf{b} \\ \hline 0 \ldots 0 & 1 \end{array}\right)$$

where $x_h = \langle x_1, \ldots x_N, x_{N+1}\rangle$ with $x_{N+1} \notin \mathbf{x}$ being a fresh variable and $\phi_h(\mathbf{x}_h) \Leftrightarrow \phi(\mathbf{x}) \wedge x_{N+1} = 1$. The encoding of an affine update $T_u \Leftrightarrow \mathbf{x}'_h = A_h \times \mathbf{x}_h$ is defined as $\sigma(T_u) = A_h \in \mathbb{Z}_\infty^{(N+1)\times(N+1)}$. Dually, for some $M \in \mathbb{Z}[k]_\infty^{(N+1)\times(N+1)}$, we define $\pi(M) \Leftrightarrow \mathbf{x}'_h = M \times \mathbf{x}_h$. With these definitions, we have $\sigma(T_u^k) = {A_h}^k$, for all $k > 0$. The next lemma proves that the class of finite monoid affine updates is periodic.

Lemma 4.29 *Let $A \in \mathbb{Z}^{N\times N}$ be a finite monoid matrix, $\mathbf{b} \in \mathbb{Z}^N$ be a vector, and let write the finite monoid generated by A as $\mathcal{M}_A = \{A^0, \ldots, A^p, \ldots, A^{p+l-1}\}$, where $p \geq 0$, $l > 0$, and $A^p = A^{p+l}$. Then, the sequence $\{{A_h}^k\}_{k=0}^{\infty}$ is periodic with prefix p and period l.*

Proof: Let $A \in \mathbb{Z}^{N\times N}$ be a matrix, $\mathbf{b} \in \mathbb{Z}^N$ be a vector, and

$$A_h \Leftrightarrow \left(\begin{array}{c|c} A & \mathbf{b} \\ \hline 0\ldots 0 & 1 \end{array}\right)$$

Then we have, for all $k \geq 0$:

$$(A_h)^k = \left(\begin{array}{c|c} A^k & \sum_{i=0}^{k-1} A^i \times \mathbf{b} \\ \hline 0\ldots 0 & 1 \end{array}\right)$$

For $i = N+1,\ 1 \leq j \leq N+1$, $\{({A_h}^k)_{ij}\}_{k=0}^{\infty}$ is trivially periodic. For $1 \leq i, j \leq N$, $\{({A_h}^k)_{ij}\}_{k=0}^{\infty}$ is periodic due to the fact that A is finite monoid. It remains to be proven that, for all $1 \leq j \leq N$, the sequence $\{(\sum_{i=0}^{k-1} A^i \times \mathbf{b})_j\}_{k=0}^{\infty}$ is periodic. Without loss of generality, assume that the monoid of A is $\mathcal{M}_A = \{A^0, A^1, \ldots, A^p, \ldots A^{p+l-1}\}$, where $A^p = A^{p+l}$. Then, for $k \geq p$, we have:

$$\textstyle\sum_{i=0}^{k-1} A^i = \sum_{i=0}^{p-1} A^i + \lfloor \frac{k-p+1}{l} \rfloor \cdot \sum_{i=p}^{p+l-1} A^i + \sum_{i=p}^{p+((k-p+1) \mod l)} A^i.$$

Hence the sequence $\{\sum_{i=0}^{k-1} A^i\}_{k=0}^{\infty}$ is periodic with prefix p, period l, and rates

$$\Lambda_j = \textstyle\sum_{i=p}^{p+l-1} A^i$$

for all $j = 0, 1, \ldots, l-1$. Consequently, A_h is periodic with the same prefix and period. □

As a direct consequence, we have the following theorem.

Theorem 4.30 *The class of finite monoid affine updates is periodic. Moreover, the transitive closures of finite monoid affine relations with Presburger definable guards are effectively Presburger definable.*

The implementation of the procedures MaxConsistent and MaxPeriodic for finite monoid affine relations is rather simple. Since we run Algorithm 1 for $*$-consistent updates of the form $\mathbf{x'}_h = A_h \times \mathbf{x}_h$ only, MaxConsistent needs to return always ∞. The MaxPeriodic test can be implemented as an equivalence check between two homogeneous linear systems with univariate linear coefficients. More precisely, given a homogeneous transformation $x' = A \times x$, with $A \in \mathbb{Z}^{(N+1)\times(N+1)}$ and a matrix $\Lambda \in \mathbb{Z}^{(N+1)\times(N+1)}$, we are looking for valuations of k that satisfy the following equality

$$(A^b + k \cdot \Lambda) \times A^c = A^b + (k+1) \cdot \Lambda \tag{4.9}$$

Both $(A^b + k \cdot \Lambda) \times A^c$ and $A^b + (k+1) \cdot \Lambda$ are matrices where each entry is a univariate linear term. The test on line 9 of the Algorithm 1 guarantees that the above equality holds for at least k=0 and k=1. Clearly, if $t_1(0) = t_2(0)$ and $t_1(1) = t_2(1)$ for two univariate linear terms t_1, t_2, then $t_1(k) = t_2(k)$ for all $k \geq 0$. Hence, the MaxPeriodic returns always $L = \infty$. We summarize these observations in the following proposition

Proposition 4.31 *Given a finite monoid matrix $A \in \mathbb{Z}^{N \times N}$, integers $b \geq 0$, $c > 0$, and a matrix $\Lambda \in \mathbb{Z}^{(N+1)\times(N+1)}$ such that $A_h^{b+c} = A_h^b + \Lambda$ and $A_h^{b+1c} = A_h^{b+c} + \Lambda$, the procedures* MaxConsistent *and* MaxPeriodic *run in constant time.*

Finally, we prove the asymptotic complexity on the running of Algorithm 1 for a finite monoid affine relation R in terms of its prefix, period, and the number of variables used to define R.

Theorem 4.32 *Let R be a finite monoid affine relation with prefix B and period C. Then, Algorithm 1 computes the transitive closure of R in at most $\mathcal{O}((B + C)^2 \cdot N^3)$ time.*

Proof: Let R be a finite monoid affine relation and let R_u be the update of R. If follows from Lemma 4.29 that asymptotic bound on the time needed to compute the transitive closure of R and R_u are same. Thus, we consider only an update relation in a homogenous form encoded as a matrix $A_h \in \mathbb{Z}_\infty^{(N+1)\times(N+1)}$.

By Theorem 3.16, Algorithm 1 takes at most $\mathcal{O}((B + C)^2)$ iterations of the main loop and in each iteration and moreover, the algorithm considers a prefix and period candidates b and c such that both b and c are bounded by $\mathcal{O}((B+C)^2)$. By Proposition 4.31, procedures MaxConsistent and MaxConsistent run in constant time. The test on line 8 amounts to equality of two matrices and can be thus performed in $\mathcal{O}(N^2)$ time. The greatest power of a relation that is computed by the algorithm is R^{b+2c}. Since the composition of an update in a homogenous form with itself amounts to matrix multiplication, if follows that these computations are performed in $\mathcal{O}((B+C) \cdot N^3)$ time. Hence, the total bound on the running time of Algorithm 1 is $\mathcal{O}((B + C)^2 \cdot N^3)$. □

5 Complexity of the Transitive Closure Algorithm

This chapter is concerned with the worst-case complexity of the transitive closure algorithm from Chapter 3 (Algorithm 1) when applied to difference bounds, octagonal, and finite monoid affine relations. For a periodic relation $R \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$ with prefix $b \geq 0$ and period $c > 0$, the asymptotic bound on the running time of Algorithm 1 is $\mathcal{O}((b+c)^8 \cdot \|R\|^3 \cdot N^9)$ if R is a difference bounds or an octagonal relation (by Theorem 4.18 and 4.27), where $\|R\|$ denotes the sum of absolute values of the coefficients of R. The asymptotic bound is $\mathcal{O}((b+c)^2 \cdot N^3)$ if R is a finite monoid affine relation (by Theorem 4.32).

The main issue, dealt with in this chapter, is thus the evaluation of the upper bounds of the prefix b and period c for each of these classes of relations. We prove that for difference bounds relations, b is asymptotically bounded by $\|R\| \cdot 2^{\mathcal{O}(N)}$ and c is bounded by $2^{\mathcal{O}(N)}$. For octagonal relations, the bound on the period is same as for difference bounds relations and the prefix is bounded by $\|R\|^2 \cdot 2^{\mathcal{O}(N)}$. For finite monoid affine relations, $b + c$ is the size of the monoid, which in turn is proved to be bounded by $2^{\mathcal{O}(N^{\log_{10} 11})}$. Columns 2 and 3 in Table 5.1 summarize these results. Combining the bounds on the size of the prefix and the period with the bounds given by Theorem 4.18, 4.27, and 4.32, we obtain asymptotic bounds on the running time of Algorithm 1 in terms of N and $\|R\|$ (the last column in Table 5.1).

Table 5.1: Transitive Closure Complexities for Periodic Relations

Class	Prefix	Period	Transitive Closure
difference bounds	$\|R\| \cdot 2^{\mathcal{O}(N)}$	$2^{\mathcal{O}(N)}$	$\|R\|^8 \cdot 2^{\mathcal{O}(N)}$
octagonal	$\|R\|^2 \cdot 2^{\mathcal{O}(N)}$	$2^{\mathcal{O}(N)}$	$\|R\|^{16} \cdot 2^{\mathcal{O}(N)}$
finite monoid affine	$2^{\mathcal{O}(N^{\log_{10} 11})}$	$2^{\mathcal{O}(N^{\log_{10} 11})}$	$2^{\mathcal{O}(N^{\log_{10} 11})}$

In all cases, Algorithm 1 runs in EXPTIME in the number of variables, and PTIME in the sum of absolute values of the coefficients of R or, equivalently, in EXPTIME in the size of the binary representation of R.

5.1 Difference Bounds Relations

Any difference bounds relation $R \in \mathcal{R}_{db}$ is periodic, by Theorem 4.8. This result extends to the octagonal class $\mathcal{R}_{oct}$, by Theorem 4.21. The periodicity of difference bounds relations is a consequence of the fact that the sequence of tropical matrix powers $\{\mathcal{M}_R^{\boxtimes^i}\}_{i\geq 0}$ where $\mathcal{M}_R$ is the incidence matrix of the common transition table $T_R = (Q, \Delta, w)$ of the zigzag automata defined for R (see Section 2.3.3), is periodic by Theorem 4.6. Since each power of R is encoded by a matrix which is a projection of a tropical power of $\mathcal{M}_R$, the prefix of R is not greater than the prefix of $\{\mathcal{M}_R^{\boxtimes^i}\}_{i\geq 0}$, while the period of R is a divisor of the period of $\{\mathcal{M}_R^{\boxtimes^i}\}_{i\geq 0}$. In this section, we prove a $\|R\| \cdot 2^{\mathcal{O}(N)}$ upper bound for the prefix and a $2^{\mathcal{O}(N)}$ upper bound for the period of $\mathcal{M}_R$. By the previous arguments, these bounds are also bounds for the prefix and the period of R, respectively.

In the rest of this section, let $\mathbf{x} = \{x_1, \ldots, x_N\}$ be a set of variables, $R(\mathbf{x}, \mathbf{x}')$ be a difference bounds relation, and $T_R = (Q, \Delta, w)$ be the common transition table of zigzag automata defined for R.

5.1.1 Bounding the Prefix

We start by instantiating Lemma 4.4 for the common transition table $T_R = (Q, \Delta, w)$ of zigzag automata defined for R.

Corollary 5.1 *Let $T_R = (Q, \Delta, w)$ be the common transition table of zigzag automata defined for a difference bounds relation R, and $u, v \in Q$ be two control states. Then for every minimal weight path ρ form u to v, such that $|\rho| \geq \|R\| \cdot \|Q\|^6$, there exists a path ρ' from u to v, such that $w(\rho) = w(\rho')$ and $|\rho| = |\rho'|$, and a basic path scheme $\theta = \sigma \cdot \lambda^* \cdot \sigma'$, such that $\rho' = \sigma \cdot \lambda^b \cdot \sigma'$, for some $b \geq 0$. Moreover, there exists $c \mid \frac{lcm(1,\ldots,\|Q\|-1)}{|\lambda|}$ such that $\sigma \cdot \lambda^{b+kc} \cdot \sigma'$ is a minimal weight path from u to v, for all $k \geq 0$.*

Proof: We obtain the statement of the corollary by instantiating Lemma 4.4 with $T_R = (Q, \Delta, w)$, in which case $\mu(T_R) \leq \|R\|$. □

Similarly, we instantiate Theorem 4.6.

Corollary 5.2 *Let $T_R = (Q, \Delta, w)$ be the common transition table of zigzag automata defined for a difference bounds relation R, and let $\mathcal{M}_R$ be its incidence matrix. Then, the sequence $\{\mathcal{M}_R^{\boxtimes^i}\}_{i\geq 0}$ is periodic. Moreover, its prefix b is bounded by $\|R\| \cdot 2^{\mathcal{O}(N)}$, and its period divides $\mathrm{lcm}(1, \ldots, 5^N)$ and is bounded by $2^{2^{\mathcal{O}(N)}}$.*

Proof: We obtain the statement of the corollary by instantiating Theorem 4.6 with $T_R = (Q, \Delta, w)$, in which case $\mu(T_R) \leq \|R\|$ and $\|Q\| = 5^N$. □

A direct consequence of Corollary 5.2 is that the prefix of a difference bounds relation R is bounded by $\|R\| \cdot 2^{\mathcal{O}(N)}$.

Corollary 5.3 *Let $\mathbf{x} = \{x_1, \ldots, x_N\}$ be a set of variables. Given a difference bounds relation $R(\mathbf{x}, \mathbf{x}')$, its prefix is bounded by $\|R\| \cdot 2^{\mathcal{O}(N)}$.*

Proof: We distinguish two cases. First, if R is $*$-consistent, then the bound $\|R\| \cdot 2^{\mathcal{O}(N)}$ on the prefix of $T_R = (Q, \Delta, w)$ that follows from Corollary 5.2 is also the bound on the prefix of R, by Proposition 4.9.
If R is not $*$-consistent, there exists a power $\ell > 0$ such that $R^\ell \Leftrightarrow \bot$. Consequently, there exists a minimal weight path ρ of length ℓ in the even zigzag automaton for R, recognizing a negative cycle. By Lemma 4.3, there exists an equivalent path ρ' of the form $\sigma \cdot \lambda^k \cdot \sigma'$, where $|\sigma \cdot \sigma'| < 5^{4N}$ and $|\lambda| < 5^N$, for some $k \geq 0$. We have $\ell = |\sigma \cdot \sigma'| + k|\lambda|$. The prefix of R is the minimal length ℓ such that $w(\rho) = w(\rho') < 0$. If $w(\sigma \cdot \sigma') < 0$, then this length is $|\sigma\sigma'| < 5^{4N}$. Otherwise, if $w(\sigma \cdot \sigma') \geq 0$, we have $w(\lambda) < 0$, or else ρ' could not encode a negative cycle, independently of how large it is. Then $w(\rho') < 0$ if and only if $k > \frac{w(\sigma \cdot \sigma')}{-w(\lambda)}$ Since $-w(\lambda) > 0$ and $w(\lambda) \in \mathbb{Z}$, we have $-w(\lambda) \geq 1$. A sufficient condition is that $k > \|R\| \cdot 5^{4N} > w(\sigma \cdot \sigma')$, hence $\ell = |\rho'| > 5^{4N} + \|R\| \cdot 5^{5N}$, i.e. the prefix of a $*$-inconsistent relation R is also asymptotically bounded by $\|R\| \cdot 2^{\mathcal{O}(N)}$. □

Similarly, a direct consequence of Corollary 5.2 is the bound on the period which is double exponential in N. In the next section, we prove that this bound can be improved to $2^{\mathcal{O}(N)}$.

Corollary 5.4 *Let $\mathbf{x} = \{x_1, \ldots, x_N\}$ be a set of variables. Given a difference bounds relation $R(\mathbf{x}, \mathbf{x}')$, its period is bounded by $2^{2^{\mathcal{O}(N)}}$.*

Proof: The period of $*$-inconsistent relation R is 1, by Definition 3.8, which is clearly bounded by $2^{2^{\mathcal{O}(N)}}$.

Let $T_R = (Q, \Delta, w)$ be the common transition table of zigzag automata of a $*$-consistent R. By Corollary 5.2, the period c of T_R is bounded by $2^{2^{\mathcal{O}(N)}}$. Since each element M_{R^i}, $i \geq 0$, of the sequence $\{M_{R^n}\}_{n \geq 0}$ of powers of R is obtained as a projection of $\mathcal{M}_R^{\boxtimes i}$, it follows that the period of R divides the period of T_R. Thus, the period of R is bounded by $2^{2^{\mathcal{O}(N)}}$ too. □

5.1.2 Bounding the Period

In this section, we refine Corollary 5.4 and show that the period of difference bounds relations is bounded by a single exponential. We start by defining several key notions and giving a high level idea of the proof.

Key Notions and a Proof Idea

Given a difference bounds relation $R(\mathbf{x}, \mathbf{x}')$, $\mathbf{x} = \{x_1, \ldots, x_N\}$ and the unfolded graph $\mathcal{G}_R^\omega$, we define the composition, power, relative length, and relative average weight operators on paths in $\mathcal{G}_R^\omega$.

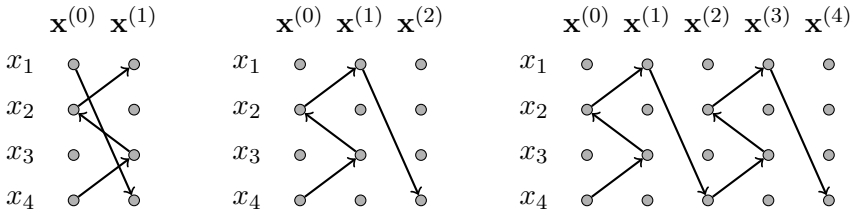
Definition 5.5 *Let $R(\mathbf{x}, \mathbf{x}')$, $\mathbf{x} = \{x_1, \ldots, x_N\}$, be a difference bounds relation and let*

$$\rho : x_{i_0}^{(l_0)} \to x_{i_1}^{(l_1)} \to \ldots \to x_{i_m}^{(l_m)} \qquad \rho' : x_{j_0}^{(k_0)} \to x_{j_1}^{(k_1)} \to \ldots \to x_{j_n}^{(k_n)}$$

$m, n \geq 1$, *be two paths in* $\mathcal{G}_R^\omega$. *The* relative path length operator *is defined as* $\|\rho\| = |l_m - l_0|$. *If* $\|\rho\| > 0$, *we define the* relative average weight *operator* $\overline{\overline{w}}(\rho) = \frac{w(\rho)}{\|\rho\|}$. *If* $i_m = j_0$, *we define* $\rho.\rho'$, *the* composition *of* ρ *with* ρ', *as*

$$\rho.\rho' = x_{i_0}^{(l_0)} \to x_{i_1}^{(l_1)} \to \ldots \to x_{i_m}^{(l_m)} \to x_{j_1}^{(k_1-d)} \to \ldots \to x_{j_n}^{(k_n-d)}$$

where $d = k_0 - l_m$. *Further, if* $i_0 = i_m$, *we define* ρ^k, $k \geq 1$, *the* k*-th* power *of* ρ *as* k*-times composition of* ρ *with itself.*



(a) Edges (b) Repeating path ρ (c) ρ^2

Figure 5.1: Illustration of repeating path.

Example 5.6 *Consider the edges in Figure 5.1(a) and build paths* $\rho_1 : x_4^{(0)} \to x_3^{(1)} \to x_2^{(0)} \to x_1^{(1)}$ *and* $\rho_2 : x_1^{(0)} \to x_4^{(1)}$. *Their concatenation* $\rho_1.\rho_2$ *results in a path* $\rho : x_4^{(0)} \to x_3^{(1)} \to x_2^{(0)} \to x_1^{(1)} \to x_4^{(2)}$ *depicted in Figure 5.1(b). Note that* $\|\rho\| = 2$. *The second power of* ρ, *denoted* ρ^2, *is depicted in Figure 5.1(c).*

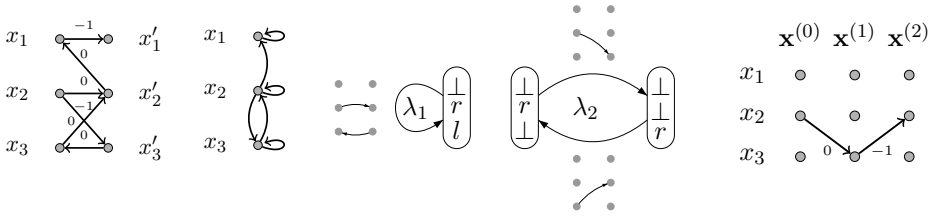
Next, we define several notions characterizing the structure of paths in $\mathcal{G}_R^\omega$.

Definition 5.7 *Let* $R(\mathbf{x}, \mathbf{x}')$, $\mathbf{x} = \{x_1, \ldots, x_N\}$, *be a difference bounds relation and let* $\rho = x_{i_0}^{(l_0)} \rightsquigarrow x_{i_m}^{(l_m)}$ *be a path in* $\mathcal{G}_R^\omega$. *We say that* ρ *is* forward (fw) *if and only if* $l_m > l_0$. *We say that* ρ *is* backward (bw) *if and only if* $l_m < l_0$. *We say that* ρ *is* repeating *if and only if* $i_0 = i_m$. *We say that* ρ *is* essential *if and only if for all* $1 \leq j < k \leq m$, $i_j = i_k$ *only if* $j = 0$ *and* $k = m$. *Path* ρ *is said to be a cycle if and only* $i_0 = i_m$ *and* $l_0 = l_m$. *We say that* ρ *is cyclic if and only if* ρ *has a subpath that is a cycle. A path is acyclic if and only if it is not cyclic.*

Intuitively, repeating path can composed with itself arbitrary many times. Note that the length of an essential path ρ is at most N, $|\rho| \leq N$. Consequently, its relative length is at most N too, $\|\rho\| \leq N$. Next, we define $\mathcal{G}_R^f$, the *folded graph* of R. Intuitively, $\mathcal{G}_R^f$ projects all edges onto unprimed variables $\mathbf{x}$.

Definition 5.8 *Let* $R(\mathbf{x}, \mathbf{x}')$, $\mathbf{x} = \{x_1, \ldots, x_N\}$, *be a difference bounds relation and let* $\mathcal{G}_R$ *be its graph representation. The folded graph of* $\mathcal{G}_R$ *is defined as* $\mathcal{G}_R^f = (\mathbf{x}, E)$, *where* $x_i \to x_j$ *is an edge in* E *if and only if* $x_i \xrightarrow{c} x_j$, $x_i' \xrightarrow{c} x_j'$, $x_i \xrightarrow{c} x_j'$, *or* $x_i' \xrightarrow{c} x_j$ *is an edge of* $\mathcal{G}_R$. *We write* $x_i \sim x_j$ *if and only if* x_i *and* x_j *belong to the same strongly connected component of* $\mathcal{G}_R^f$. *Clearly,* $\sim$ *is an equivalence relation.*

Note that folded graphs are not weighted. Figure 5.2(b) depicts the folded graph $\mathcal{G}_R^f$ of $\mathcal{G}_R$ from Figure 5.2(a). The folded graph in Figure 5.2(b) has two strongly connected components $\{x_1\}$ and $\{x_2, x_3\}$.



(a) $\mathcal{G}_R$ (b) $\mathcal{G}_R^f$ (c) T_R with a simple cycle λ_2 (d) The label of λ_2

Figure 5.2: Folded graph. Zigzag automaton with a simple cycle.

Then, we observe that each cycle λ in $T_R = (Q, \Delta, w)$ (a *zigzag cycle*, for short) encodes a set of forward $*$-acyclic and backward $*$-acyclic paths.

Definition 5.9 *A repeating path ρ is* $*$-acyclic *if and only if ρ^k is acyclic for all $k \geq 1$.*

Intuitively, a $*$-acyclic path can be composed with itself arbitrary many times without producing cyclic subpaths. For instance, the path ρ depicted in Figure 5.1(b) is $*$-acyclic. Note that each essential repeating path is $*$-acyclic.

Further, we study the structure of path schemes $\sigma.\lambda^*.\sigma'$ from Lemma 5.1 and prove that one can without loss of generality assume that λ is *simple*.

Definition 5.10 *A cycle λ in the transition table of a zigzag automaton $T_R = (Q, \Delta, w)$ is* simple *if and only if it encodes at most one $*$-acyclic path per equivalence class of $\sim$ relation. A basic path scheme $\theta = \sigma.\lambda^*.\sigma'$ is* simple *if and only if λ is simple.*

Figure 5.2(c) illustrates a part of the transition table of the zigzag automaton corresponding to the relation in Figure 5.2(a). The cycle λ_1 depicted in Figure 5.2(c) is not simple, since it encodes two $*$-acyclic paths $x_2^{(0)} \xrightarrow{0} x_2^{(1)}$ and $x_3^{(0)} \xrightarrow{0} x_3^{(1)}$ from the same strongly connected component $\{x_2, x_3\}$. On the other hand, λ_2 is simple, since it encodes only one $*$-acyclic path $x_2^{(0)} \xrightarrow{0} x_3^{(1)} \xrightarrow{-1} x_2^{(2)}$ depicted in Figure 5.2(d).

Next, we prove that we can make the statement of Lemma 5.1 even more accurate and consider, without loss of generality, only path schemes with cycles whose length divides $\mathrm{lcm}(1, \ldots, N)$. Let μ_j be a $*$-acyclic path of the form $\mu_j : x_{i_j} \rightsquigarrow x_{i_j}$ encoded in a simple zigzag cycle λ, where $x_{i_j} \in \mathbf{z}_j$ for some equivalence class $\mathbf{z}_j \in \mathbf{x}_{/\sim}$. We first observe that there exists an essential $*$-acyclic path μ_j of the form $\nu_j : x_{k_j} \rightsquigarrow x_{k_j}$, where $x_{k_j} \in \mathbf{z}_j$ as well. Supposing that λ encodes m $*$-acyclic paths, the intuition is to build a cycle λ' as follows: letting $L = \mathrm{lcm}\{\|\nu_1\|, \ldots, \|\nu_m\|\}$, the cycle λ' will encode paths $\nu_j^{d_j}$, where $d_j = \frac{L}{\|\nu_j\|}$. Then, since $|\lambda'| = L$ and $\|\nu_j\| \leq N$, it follows that the length of λ' divides $\mathrm{lcm}\{1, \ldots, N\}$.

Since μ_j and ν_j belong to the same equivalence class $\mathbf{z}_j$, by the above observations, it follows that there exist *essential* paths $\xi_j : x_{i_j} \rightsquigarrow x_{k_j}$ and $\zeta : x_{k_j} \rightsquigarrow x_{i_j}$. These paths can

be used to connect μ_j with ν_j and vice versa. The notion of a *connecting path* captures this idea:

Definition 5.11 *A forward repeating path $\tau : x_i \rightsquigarrow x_i$ is called a* connecting path *if it is of the form*

$$\tau \quad = \quad \mu^r \quad . \quad \xi \quad . \quad \nu^s \quad . \quad \zeta \quad . \quad \mu^t$$

where

- *$\mu : x_i \rightsquigarrow x_i$, $\nu : x_k \rightsquigarrow x_k$ are forward $*$-acyclic paths, and*
- *$\xi : x_i \rightsquigarrow x_k$, $\zeta : x_k \rightsquigarrow x_i$ are essential paths.*

Note that in a connecting path τ, the repeating paths μ and ν are allowed to be raised to a positive powers r, s, t. Figure 5.3 illustrates a connecting path.

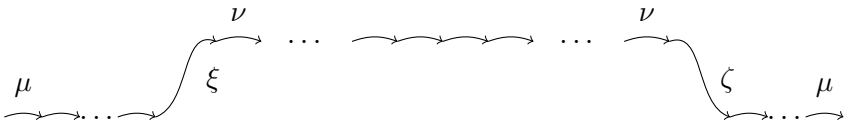


Figure 5.3: Connecting path τ.

In order to ensure correctness of the construction, we also need to build two zigzag paths π_1 and π_2 that will connect λ with λ' and vice versa, respectively. In other words, we need to ensure that $\lambda.\pi_1.\lambda'.\pi_2.\lambda$ will form a valid zigzag path. To this end, we build connecting paths $\tau_1, \ldots, \tau_m$. Here we encounter a problem of synchronizing the positions at which ν_j appears for the first time in τ_j. This problem is due to the fact that the relative length of ξ_j may be arbitrary – we only know that its relative length is bounded by N, since ξ_j is essential. Lemma 5.26 proves that this problem can be overcome. Finally, we prove in Lemma 5.29 the desired claim that we can, without loss of generality, assume path schemes $\sigma.\lambda^*.\sigma'$ where $|\lambda|$ divides $\mathrm{lcm}(1, \ldots, N)$.

Then, we can refine Corollary 5.4 and establish the upper bound of $2^{O(n)}$ on the period for zigzag automata. Since $M^*_{R^m}$, the encoding of the m-th power of R, is a projection of $\mathcal{M}_R^{\boxtimes m}$, the bound on the period of the sequence $\{\mathcal{M}_R^{\boxtimes m}\}_{m\geq 0}$ is a valid bound on the period of the sequence $\{M^*_{R^m}\}_{m\geq 0}$ and consequently, it is a bound on the period of a difference bounds relation R (Theorem 5.31). In Theorem 5.32, we show that this result extends rather easily to the period of octagonal relations. Moreover, Theorem 5.32 shows how the bound of $\|R\|^2 \cdot 2^{\mathcal{O}(N)}$ on the prefix of and octagonal relation can be inferred.

Repeating Paths – Decomposition and Optimality

Repeating essential paths can be seen as building blocks of each repeating path, as the following proposition states. Note that each essential repeating path is either forward, backward or an elementary cycle.

Proposition 5.12 *Each repeating path ρ can be decomposed into a set of essential repeating paths $\mathcal{F}(\rho)$ such that*

$$w(\rho) \;=\; \sum_{\mu\in\mathcal{F}_{\triangleright}(\rho)} w(\mu) + \sum_{\mu\in\mathcal{F}_{\triangleleft}(\rho)} w(\mu) + \sum_{\mu\in\mathcal{F}_{\circ}(\rho)} w(\mu)$$

where $\mathcal{F}_{\triangleright}(\rho), \mathcal{F}_{\triangleleft}(\rho), \mathcal{F}_{\circ}(\rho) \subseteq \mathcal{F}(\rho)$ are the maximal subset of forward, backward, and cyclic paths, respectively. Moreover,

$$\begin{array}{lcll} \|\rho\| & = & \sum_{\mu\in\mathcal{F}_{\triangleright}(\rho)} \|\mu\| - \sum_{\mu\in\mathcal{F}_{\triangleleft}(\rho)} \|\mu\| & \text{if } \rho \text{ is forward,} \\ \|\rho\| & = & \sum_{\mu\in\mathcal{F}_{\triangleleft}(\rho)} \|\mu\| - \sum_{\mu\in\mathcal{F}_{\triangleright}(\rho)} \|\mu\| & \text{if } \rho \text{ is backward,} \\ \|\rho\| & = & \sum_{\mu\in\mathcal{F}_{\triangleleft}(\rho)} \|\mu\| = \sum_{\mu\in\mathcal{F}_{\triangleright}(\rho)} \|\mu\| = 0 & \text{if } \rho \text{ is a cycle.} \end{array}$$

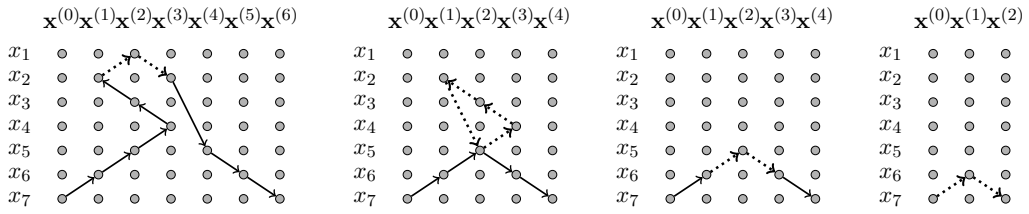
Proof: Let ρ be arbitrary path and denote $\rho_0 = \rho$. For each $i \geq 0$, we define ρ_{i+1} inductively as follows. Let μ_i be an arbitrary essential repeating subpath of ρ_i, i.e. $\rho_i = \theta_i.\mu_i.\theta'_i$ for some θ_i, θ'_i. Then, construct ρ_{i+1} by erasing μ_i from ρ_i, i.e. $\rho_{i+1} = \theta_i.\theta'_i$. Clearly, this decomposition terminates since ρ_{k+1} is empty for some $k \geq 0$. Then, $\mathcal{F}(\rho) = \{\mu_0, \ldots, \mu_k\}$. Next, let us define $D_i \in \mathbb{Z}$, $i = k+1, \ldots, 0$ inductively as follows: $D_{k+1} = 0$ and for each $i = k, \ldots, 0$, define

$$\begin{array}{ll} D_i = D_{i+1} + \|\mu_i\| & \text{if } \mu_i \text{ is forward,} \\ D_i = D_{i+1} - \|\mu_i\| & \text{if } \mu_i \text{ is backward,} \\ D_i = D_{i+1} & \text{if } \mu_i \text{ is an elementary cycle.} \end{array}$$

Clearly, for each $1 \leq i \leq k$,

$$\begin{array}{llllll} \|\rho_i\| = D_i & \textbf{iff} & \rho_i \text{ is forward} & \textbf{iff} & D_i > 0, \\ \|\rho_i\| = -D_i & \textbf{iff} & \rho_i \text{ is backward} & \textbf{iff} & D_i < 0, \\ \|\rho_i\| = 0 & \textbf{iff} & \rho_i \text{ is a cycle} & \textbf{iff} & D_i = 0. \end{array}$$

Recall that $\rho = \rho_0$. Thus, if ρ if forward, then $\|\rho\| = \sum_{\mu\in\mathcal{F}_{\triangleright}(\rho)} \|\mu\| - \sum_{\mu\in\mathcal{F}_{\triangleleft}(\rho)} \|\mu\|$ and if ρ is backward, then $\|\rho\| = \sum_{\mu\in\mathcal{F}_{\triangleleft}(\rho)} \|\mu\| - \sum_{\mu\in\mathcal{F}_{\triangleright}(\rho)} \|\mu\|$. Clearly, if ρ is a cycle, then $\|\rho\| = \sum_{\mu\in\mathcal{F}_{\triangleleft}(\rho)} \|\mu\| = \sum_{\mu\in\mathcal{F}_{\triangleright}(\rho)} \|\mu\| = 0$. □



(a) ρ_0 and its subpath μ_0 (b) ρ_1, μ_1 (c) ρ_2, μ_2 (d) ρ_3, μ_3

Figure 5.4: Decomposition of a repeating path to essential repeating paths.

Example 5.13 *Consider a path ρ depicted in Figure 5.4(a). Figures 5.4(a-d) illustrate a decomposition of ρ into essential repeating paths. Essential subpaths $\mathcal{F}(\rho) = \{\mu_0, \ldots, \mu_3\}$ selected during the decomposition are dotted.*

We next show that the average weight of a repeating path ρ is equal to the average weight of an arbitrary power of ρ.

Proposition 5.14 *Let ρ be a repeating path and let $d \geq 1$. Then, $\overline{\overline{w}}(\rho^d) = \overline{\overline{w}}(\rho)$.*

Proof: Observe that $\overline{\overline{w}}(\rho^d) = \frac{d \cdot w(\rho)}{d \cdot \|\rho\|} = \frac{w(\rho)}{\|\rho\|} = \overline{\overline{w}}(\rho)$. □

Next, we introduce a notion of *optimal* path.

Definition 5.15 *Let $\mathbf{z} \in \mathbf{x}_{/\sim}$ be an equivalence class of $\sim$ and let $S^{\mathbf{z}}_{\triangleright}$ ($S^{\mathbf{z}}_{\triangleleft}$) be the set of all forward (backward) repeating paths in $\mathcal{G}_R$ of the form $x_i \rightsquigarrow x_i$, for some $x_i \in \mathbf{z}$. A path $\rho \in S^{\mathbf{z}}_{\triangleright}$ is* $\triangleright$-optimal *if and only if $\overline{\overline{w}}(\rho) \leq \overline{\overline{w}}(\rho)$ for all $\rho' \in S^{\mathbf{z}}_{\triangleright}$. Similarly, a path $\rho \in S^{\mathbf{z}}_{\triangleleft}$ is* $\triangleleft$-optimal *if and only if $\overline{\overline{w}}(\rho) \leq \overline{\overline{w}}(\rho)$ for all $\rho' \in S^{\mathbf{z}}_{\triangleleft}$.*

We next show that the average weight of optimal paths are determined by average weights of *critical* essential repeating paths. Thus, we first characterize these paths. For each equivalence class $\mathbf{z} \in \mathbf{x}_{/\sim}$, we define the set of essential repeating forward paths $P_{\triangleright}(\mathbf{z})$, minimal average weight of these paths $C_{\triangleright}(\mathbf{z})$, and a subset of *fw-critical* paths $P^c_{\triangleright}(\mathbf{z})$ as follows. Note that we allow a path to cross nodes $\mathbf{x}^{(\ell)}$, where $\ell < 0$, for notational convenience.

$$\begin{array}{lcl} P_{\triangleright}(\mathbf{z}) & = & \{\rho : x_i^{(\ell)} \rightsquigarrow x_i^{(\ell')} \mid \ell' > \ell, \rho \text{ is an essential, repeating path in } \mathcal{G}_R^m, m \geq 0\} \\ C_{\triangleright}(\mathbf{z}) & = & \min\{\|\rho\| \mid \rho \in P_{\triangleright}(\mathbf{z})\} \\ P^c_{\triangleright}(\mathbf{z}) & = & \{\rho \in P_{\triangleright}(\mathbf{z}) \mid \|\rho\| = C_{\triangleright}(\mathbf{z})\} \end{array}$$

Similarly, we define $P_{\triangleleft}(\mathbf{z}), C_{\triangleleft}(\mathbf{z}), P^c_{\triangleleft}(\mathbf{z})$ for backward paths.

The following lemma gives a precise characterization of optimal paths, based on properties of critical paths defined above.

Lemma 5.16 *Let $\mathbf{z} \subseteq \mathbf{x}$ be an equivalence class of $\sim$ and let $\rho : x \rightsquigarrow x$, $x \in \mathbf{z}$, be a repeating path in $\mathcal{G}_R$.*

1. *If ρ is forward, then $\overline{\overline{w}}(\rho) \geq C_{\triangleright}(\mathbf{z})$. Moreover, ρ is $\triangleright$-optimal if and only if $\overline{\overline{w}}(\rho) = C_{\triangleright}(\mathbf{z})$ if and only if*
   a) *$\overline{\overline{w}}(\mu) = C_{\triangleright}(\mathbf{z})$ for each forward path $\mu \in \mathcal{F}(\rho)$,*
   b) *$\overline{\overline{w}}(\mu) = -C_{\triangleright}(\mathbf{z})$ for each backward path $\mu \in \mathcal{F}(\rho)$,*
   c) *$w(\mu) = 0$ for each cycle $\mu \in \mathcal{F}(\rho)$.*

   Moreover, if $\mathcal{F}(\rho)$ contains a backward path and $\overline{\overline{w}}(\rho) = C_{\triangleright}(\mathbf{z})$, then $C_{\triangleright}(\mathbf{z}) = -C_{\triangleleft}(\mathbf{z})$.

2. *If ρ is backward, then $\overline{\overline{w}}(\rho) \geq C_{\triangleleft}(\mathbf{z})$. Moreover, ρ is $\triangleleft$-optimal if and only if $\overline{\overline{w}}(\rho) = C_{\triangleleft}(\mathbf{z})$ if and only if*
   a) *$\overline{\overline{w}}(\mu) = -C_{\triangleleft}(\mathbf{z})$ for each forward path $\mu \in \mathcal{F}(\rho)$,*
   b) *$\overline{\overline{w}}(\mu) = C_{\triangleleft}(\mathbf{z})$ for each backward path $\mu \in \mathcal{F}(\rho)$,*
   c) *$w(\mu) = 0$ for each cycle $\mu \in \mathcal{F}(\rho)$.*

Moreover, if $\mathcal{F}(\rho)$ contains a forward path and $\overline{\overline{w}}(\rho) = C_\triangleleft(\mathbf{z})$, then $C_\triangleleft(\mathbf{z}) = -C_\triangleright(\mathbf{z})$.

3. *If ρ is a cycle, then $\overline{\overline{w}}(\rho) \geq 0$. Moreover, $w(\rho) = 0$ if and only if*
   a) *$\overline{\overline{w}}(\mu) = C_\triangleright(\mathbf{z}) = -C_\triangleleft(\mathbf{z})$ for each forward path $\mu \in \mathcal{F}(\rho)$,*
   b) *$\overline{\overline{w}}(\mu) = C_\triangleleft(\mathbf{z}) = -C_\triangleright(\mathbf{z})$ for each backward path $\mu \in \mathcal{F}(\rho)$,*
   c) *$w(\mu) = 0$ for each cycle $\mu \in \mathcal{F}(\rho)$.*

Proof: We give the proof for the case when ρ is forward. Proofs for other cases are similar.

Let $S_\triangleright, S_\triangleleft, S_\circ \subseteq \mathcal{F}(\rho)$ be the sets of all forward paths, backward paths, and cycles in $\mathcal{F}(\rho)$, respectively. Clearly, $\overline{\overline{w}}(\mu) \geq C_\triangleright(\mathbf{z})$ for each $\mu \in S_\triangleright$. As a corollary of Lemma 5.23, $C_\triangleleft(\mathbf{z}) + C_\triangleright(\mathbf{z}) \geq 0$ and thus, $\overline{\overline{w}}(\mu) \geq C_\triangleleft(\mathbf{z}) \geq -C_\triangleright(\mathbf{z})$ for each $\mu \in S_\triangleleft$. Since R is $*$-consistent, then $w(\nu) \geq 0$ for each $\mu \in S_\circ$. Thus, for each $\mu \in S_\triangleright$, there exists $d_\mu \geq 0$ such that $\overline{\overline{w}}(\mu) = C_\triangleright(\mathbf{z}) + d_\mu$. Similarly, for each $\mu \in S_\triangleleft$, there exists $d_\mu \geq 0$ such that $\overline{\overline{w}}(\mu) = -C_\triangleright(\mathbf{z}) + d_\mu$. Let us define:

$$w(S_\triangleright) = \sum_{\mu \in S_\triangleright} w(\nu) \qquad w(S_\triangleleft) = \sum_{\mu \in S_\triangleleft} w(\nu) \qquad w(S_\circ) = \sum_{\mu \in S_\circ} w(\mu)$$

We derive:

$$\begin{array}{rcl} w(S_\triangleright) & = & \sum_{\mu \in S_\triangleright} w(\nu) = \sum_{\mu \in S_\triangleright} \overline{\overline{w}}(\mu)\|\mu\| = \sum_{\mu \in S_\triangleright} (C_\triangleright(\mathbf{z}) + d_\mu)\|\mu\| \\ & = & C_\triangleright(\mathbf{z}) \sum_{\mu \in S_\triangleright} \|\mu\| + \sum_{\mu \in S_\triangleright} d_\mu \|\mu\| \\ w(S_\triangleleft) & = & \sum_{\mu \in S_\triangleleft} w(\nu) = \sum_{\mu \in S_\triangleleft} \overline{\overline{w}}(\mu)\|\mu\| = \sum_{\mu \in S_\triangleleft} (-C_\triangleright(\mathbf{z}) + d_\mu)\|\mu\| \\ & = & -C_\triangleright(\mathbf{z}) \sum_{\mu \in S_\triangleright} \|\mu\| + \sum_{\mu \in S_\triangleright} d_\mu \|\mu\| \end{array}$$

Observe that:

$$\begin{array}{rcl} w(S_\triangleright) + w(S_\triangleleft) & = & C_\triangleright(\mathbf{z}) \left(\sum_{\mu \in S_\triangleright} \|\mu\| - \sum_{\mu \in S_\triangleleft} \|\mu\| \right) + \sum_{\mu \in S_\triangleright \cup S_\triangleleft} d_\mu \|\mu\| \\ & = & C_\triangleright(\mathbf{z})\|\rho\| + \sum_{\mu \in S_\triangleright \cup S_\triangleleft} d_\mu \|\mu\| \end{array}$$

The last equality holds since $\|\rho\| = \sum_{\mu \in S_\triangleright} \|\mu\| - \sum_{\mu \in S_\triangleleft} \|\mu\|$. Since $w(\rho) = w(S_\triangleright) + w(S_\triangleleft) + w(S_\circ)$, we infer that

$$\begin{array}{rcl} w(\rho) & = & w(S_\triangleright) + w(S_\triangleleft) + w(S_\circ) \\ & = & C_\triangleright(\mathbf{z})\|\rho\| + \sum_{\mu \in S_\triangleright} d_\mu \|\mu\| + \sum_{\mu \in S_\triangleleft} d_\mu \|\mu\| + \sum_{\mu \in S_\circ} w(\mu) \end{array}$$

Consequently,

$$\overline{\overline{w}}(\rho) \quad = \quad C_\triangleright(\mathbf{z}) + \frac{\sum_{\mu \in S_\triangleright} d_\mu \|\mu\| + \sum_{\mu \in S_\triangleleft} d_\mu \|\mu\| + \sum_{\mu \in S_\circ} w(\mu)}{\|\rho\|}.$$

Since the fraction in the above equation is non-negative, then $\overline{\overline{w}}(\rho) \geq C_\triangleright(\mathbf{z})$. Moreover, $\overline{\overline{w}}(\rho) = C_\triangleright(\mathbf{z})$ if and only if $d_\nu = 0$ for each $\nu \in S_\triangleright$, $d_\nu = 0$ for each $\nu \in S_\triangleleft$, and $w(\nu) = 0$ for each $\nu \in S_\circ$ if and only if $\overline{\overline{w}}(\nu) = C_\triangleright(\mathbf{z})$ for each $\nu \in S_\triangleright$, $\overline{\overline{w}}(\nu) = -C_\triangleright(\mathbf{z})$ for each $\nu \in S_\triangleleft$, and $w(\nu) = 0$ for each $\nu \in S_\circ$.

Suppose that $\mathcal{F}_{\triangleleft}(\rho) \neq \emptyset$ and $\overline{\overline{w}}(\rho) = C_{\triangleright}(\mathbf{z})$. Recall that $\overline{\overline{w}}(\mu) \geq C_{\triangleleft}(\mathbf{z}) \geq -C_{\triangleright}(\mathbf{z})$ for each $\mu \in S_{\triangleleft}$. By the above arguments, $\overline{\overline{w}}(\mu) = -C_{\triangleright}(\mathbf{z})$. Thus, $-C_{\triangleright}(\mathbf{z}) \geq C_{\triangleleft}(\mathbf{z}) \geq -C_{\triangleright}(\mathbf{z})$ and consequently, $C_{\triangleleft}(\mathbf{z}) = -C_{\triangleright}(\mathbf{z})$. □

The following technical proposition is later used for proving properties of connecting paths.

Proposition 5.17 *Let $\mathbf{z} \subseteq \mathbf{x}$ be an equivalence class of $\sim$ and let $\rho : x_i \rightsquigarrow x_i$, $x_i \in \mathbf{z}$, be an optimal forward repeating path in $\mathcal{G}_R$. Then, there exist an optimal essential forward path $\rho' : x_j \rightsquigarrow x_j$ and essential paths $\xi : x_i \rightsquigarrow x_j$, $\zeta : x_j \rightsquigarrow x_i$ such that*

- $\overline{\overline{w}}(\xi.\zeta) = C_{\triangleright}(\mathbf{z})$ *if $\xi.\zeta$ is forward,*
- $\overline{\overline{w}}(\xi.\zeta) = -C_{\triangleright}(\mathbf{z})$ *if $\xi.\zeta$ is backward,*
- $w(\xi.\zeta) = 0$ *if $\xi.\zeta$ forms a cycle.*

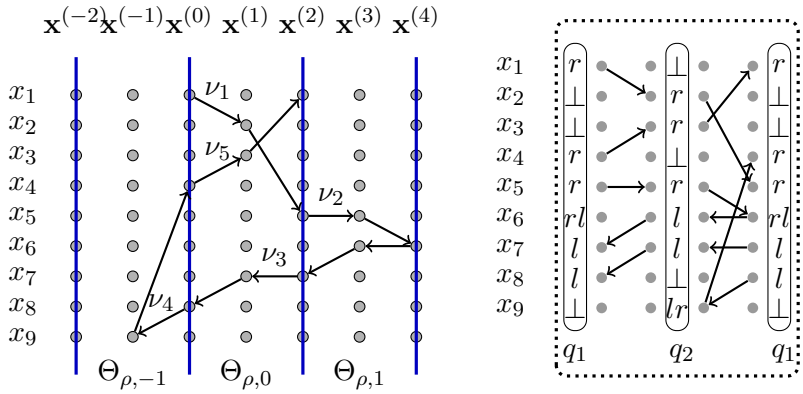
Moreover, $\rho' \in \mathcal{F}(\rho)$ and $\mathcal{F}(\xi.\zeta) \subseteq \mathcal{F}(\rho)$.

Proof: Let $\rho_0, \ldots, \rho_k$ and $\mu_0, \ldots, \mu_k \subseteq \mathcal{F}(\rho)$ be paths constructed in a decomposition of ρ into essential repeating paths as in the proof of Proposition 5.12. Clearly, there exists $0 \leq m \leq k$ such that $\mu_m : x_j \rightsquigarrow x_j$ is forward. Let θ, θ' be paths such that $\rho_m = \theta.\mu_m.\theta'$. Let us decompose θ by erasing its essential repeating subpaths, obtaining $\theta_0, \ldots, \theta_\ell$. Similarly, we decompose θ' and obtain $\theta'_0, \ldots, \theta'_{\ell'}$. Note that we can without loss of generality assume that $\rho_{m+n} = \theta_n.\theta'$ for all $0 \leq n \leq \ell$ and that $\rho_{m+\ell+n} = \theta_\ell.\theta'_n$ for all $0 \leq n \leq \ell'$. Let $\xi = \theta_\ell, \zeta = \theta'_{\ell'}$. Since ρ is fw-optimal, then clearly $\mathcal{F}(\xi.\zeta) = \mathcal{F}(\theta_\ell.\theta'_{\ell'}) \subseteq \mathcal{F}(\rho)$. Applying Lemma 5.16, we get the remaining properties of $\theta_\ell.\theta'_{\ell'}$ stated in this proposition. □

Anatomy of Zigzag Cycles

We now inspect the structure of cycles in zigzag automata. In particular, we show that each $*$-acyclic path encoded in a zigzag cycle is a concatenation of several *zigzag-segments*:

Definition 5.18 *Let $\lambda = q_0 \xrightarrow{G_1} q_1 \xrightarrow{G_2} \ldots \xrightarrow{G_p} q_p$, where $q_0 = q_p$, be a zigzag cycle of length $|\lambda| = p$, where $G_1, \ldots, G_p$ are subgraphs of $\mathcal{G}_R$ that label edges appearing in λ. Let G be a subgraph of $\mathcal{G}_R^p$ constructed as $G = G_1.G_2 \ldots G_p$. Each path θ in G that is maximal in its length is called a* zigzag-segment.



(a) path ρ (b) zigzag cycle λ_ρ

Figure 5.5: Segmentation of a repeating path to zigzag-segments (a) and construction of a corresponding zigzag cycle (b).

Given a zigzag cycle λ of length $|\lambda| = p$, we write its segments as paths of the form $x_i^{(0)} \rightsquigarrow x_j^{(p)}$, $x_i^{(p)} \rightsquigarrow x_j^{(0)}$, $x_i^{(0)} \rightsquigarrow x_j^{(0)}$, or $x_i^{(p)} \rightsquigarrow x_j^{(p)}$.

Example 5.19 *Consider a zigzag cycle λ_ρ depicted in Figure 5.5(b). λ_ρ has five zigzag-segments:*

$$\nu_1 : x_1^{(0)} \to x_2^{(1)} \to x_5^{(2)} \qquad \nu_2 : x_5^{(0)} \to x_5^{(1)} \to x_6^{(2)} \to x_6^{(1)} \to x_7^{(0)}$$
$$\nu_3 : x_7^{(2)} \to x_7^{(1)} \to x_8^{(0)} \qquad \nu_4 : x_8^{(2)} \to x_9^{(1)} \to x_4^{(2)} \qquad \nu_5 : x_4^{(0)} \to x_3^{(1)} \to x_1^{(2)}$$

□

Each zigzag cycle λ, $|\lambda| = p$, encodes a set of forward $*$-acyclic paths of the form $x_i^{(0)} \rightsquigarrow x_i^{(p)}$ and a set of backward $*$-acyclic paths of the form $x_i^{(p)} \rightsquigarrow x_i^{(0)}$. For simplicity, let us first examine cycles which encode one forward $*$-acyclic path $\rho : x_{i_0}^{(\ell_0)} \to \ldots \to x_{i_m}^{(\ell_m)}$, $i_0 = i_m$, $p = \ell_m - \ell_0$, and no backward path. We will describe how λ_ρ, a unique zigzag cycle that encodes ρ, can be built. We first define:

$$L_\rho = |\min\{\ell_k - \ell_0 \mid 0 \le k \le m\}| \qquad \overline{L}_\rho = \left\lceil \frac{L_\rho}{\|\rho\|} \right\rceil$$
$$R_\rho = |\max\{\ell_k - \ell_0 \mid 0 \le k \le m\}| \qquad \overline{R}_\rho = \left\lceil \frac{R_\rho}{\|\rho\|} \right\rceil$$

Intuitively, L_ρ (R_ρ) is the left (right) extent of ρ relative to $x_{i_0}^{(\ell_0)}$.

Example 5.20 *(ctd.) Given a path $\rho : x_1^{(0)} \rightsquigarrow x_1^{(2)}$ depicted in Figure 5.5(a), we compute:*

$$L_\rho = |\min\{-1, \ldots, 3\}| = 1 \quad \overline{L}_\rho = \lceil \tfrac{1}{2} \rceil = 1$$
$$R_\rho = |\max\{-1, \ldots, 3\}| = 3 \quad \overline{R}_\rho = \lceil \tfrac{3}{2} \rceil = 2$$

□

Next, let decompose ρ into zigzag segments in the following manner. For each $-\overline{L}_\rho \leq j < \overline{R}_\rho$, we define $\Theta_{\rho,j}$, the set of maximal (in their length) subpath of ρ which cross only variables

$$\{\mathbf{x}^{(k)} \mid j \cdot \|\rho\| \leq k \leq (j+1) \cdot \|\rho\|\}.$$

Then, λ_ρ consists of zigzag-segments $\bigcup_{k=-\overline{L}_\rho}^{\overline{R}_\rho - 1} \Theta_{\rho,k}$. Similar definitions can be made for backward paths. This construction can be generalized for zigzag cycles encoding a set of forward and backward paths.

Example 5.21 *(ctd.) Given a path $\rho : x_1^{(0)} \rightsquigarrow x_1^{(2)}$ depicted in Figure 5.5(a), we computed $L_\rho = 1$, $\overline{L}_\rho = 1$, $R_\rho = 3$, $\overline{R}_\rho = 2$. Then, $\Theta_{\rho,-1}$ is a set of maximal subpaths of ρ crossing only $\mathbf{x}^{(-2)} \cup \mathbf{x}^{(-1)} \cup \mathbf{x}^{(0)}$, thus $\Theta_{\rho,-1} = \{\nu_4\}$. Similarly, $\Theta_{\rho,0}$ is a set of maximal subpaths of ρ crossing only $\mathbf{x}^{(0)} \cup \mathbf{x}^{(1)} \cup \mathbf{x}^{(2)}$, thus $\Theta_{\rho,0} = \{\nu_1, \nu_3, \nu_5\}$. Finally, $\Theta_{\rho,1}$ is a set of maximal subpaths of ρ crossing only $\mathbf{x}^{(2)} \cup \mathbf{x}^{(3)} \cup \mathbf{x}^{(4)}$, thus $\Theta_{\rho,1} = \{\nu_2\}$. Clearly, the zigzag cycle in Figure 5.5(b) encodes segments $\Theta_{\rho,-1} \cup \Theta_{\rho,0} \cup \Theta_{\rho,1}$.* □

The following proposition states that the average weight of a cycle λ in zigzag automaton is the sum of average weight of $*$-acyclic paths that are encoded in the label of λ.

Proposition 5.22 *Let λ be a zigzag cycle that encodes $*$-acyclic paths $\rho_1, \ldots, \rho_n$. Then $\overline{w}(\lambda) = \overline{\overline{w}}(\rho_1) + \cdots + \overline{\overline{w}}(\rho_n)$.*

Proof: Since $|\lambda| = \|\rho_1\| = \cdots = \|\rho_n\|$, we infer:

$$\overline{w}(\lambda) \quad = \quad \frac{w(\rho_1)+\cdots+w(\rho_n)}{|\lambda|} = \sum_{i=1}^{n} \frac{w(\rho_i)}{|\lambda|} = \sum_{i=1}^{n} \frac{w(\rho_i)}{\|\rho_i\|} = \sum_{i=1}^{n} \overline{\overline{w}}(\rho_i)$$

□

Basic Path Schemes with Simple Zigzag Cycles

This section refines the statement of Lemma 5.1 by proving that the cycle λ from each basic path scheme can be assumed to be simple, without loss of generality. The next lemma proves that the sum of average weights of a forward repeating and a backward repeating path in $\mathcal{G}_R^m$, $m \geq 1$, from the same equivalence class of the $\sim$ relation is non-negative, whenever R is $*$-consistent.

Lemma 5.23 *Let R be a $*$-consistent difference bounds relation and let $\rho_i = x_i \rightsquigarrow x_i$ be a forward repeating and $\rho_j = x_j \rightsquigarrow x_j$ be a backward repeating path in $\mathcal{G}_R$ such that $x_i \sim x_j$. Then, $\overline{\overline{w}}(\rho_i) + \overline{\overline{w}}(\rho_j) \geq 0$.*

Proof: Suppose that $\overline{\overline{w}}(\rho_i) + \overline{\overline{w}}(\rho_j) < 0$. Let us define:

$$p = \mathrm{lcm}(\|\rho_i\|, \|\rho_j\|), \;\; d_i = \frac{p}{\|\rho_i\|}, \;\; d_j = \frac{p}{\|\rho_j\|}, \;\; \gamma_i = (\rho_i)^{d_i}, \;\; \gamma_j = (\rho_j)^{d_j}.$$

By Proposition 5.14, $\overline{\overline{w}}(\rho_i) = \overline{\overline{w}}(\gamma_i)$ and $\overline{\overline{w}}(\rho_j) = \overline{\overline{w}}(\gamma_j)$. Thus, $\overline{\overline{w}}(\gamma_i) + \overline{\overline{w}}(\gamma_j) < 0$. Furthermore, since $\|\gamma_i\| = \|\gamma_j\| = p$, then $p \cdot \overline{w}(\gamma_i) + p \cdot \overline{w}(\gamma_j) = w(\gamma_i) + w(\gamma_j) < 0$. Since $x_i \sim x_j$, there exist essential paths

$$\theta_{ij} = x_i^{(0)} \rightsquigarrow x_j^{(q)} \text{ and } \theta_{ji} = x_j^{(0)} \rightsquigarrow x_i^{(r)}$$

where $0 \leq \|Q\|, |r| < N$. Let $n \geq 0$ be a parameter. We build (refer to Figure 5.6)

$$\xi = \gamma_i^n.\theta_{ij}.\gamma_j^{2n}.\theta_{ji}$$

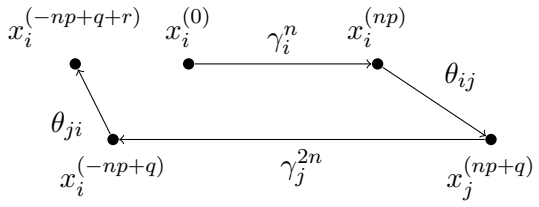


Figure 5.6: Building ξ

Clearly, ξ is of the form $\xi : x_i^{(0)} \rightsquigarrow x_i^{(-np+q+r)}$. By choosing $n > \lceil \frac{q+r}{p} \rceil$, we make sure that $-np + r + s < 0$. We repeat the path p-times and obtain $\xi^p : x_i^{(0)} \rightsquigarrow x_i^{(p(-np+q+r))}$. Since $|\gamma_i| = p$ and p divides $p(-np + q + r)$, we build $\zeta = \gamma_i^{(np-r-s)}$ which is of the form $\zeta : x_i^{(p(-np+q+r))} \rightsquigarrow x_i^{(0)}$. Clearly, $\xi^p.\zeta$ forms a cycle with weight

$$np \cdot w(\gamma_i) + p \cdot w(\theta_{ij}) + 2np \cdot w(\gamma_j) + p \cdot w(\theta_{ji}) + (np - q - r) \cdot w(\gamma_i)$$

which simplifies to

$$2np \cdot (w(\gamma_i) + w(\gamma_j)) - (q + r) \cdot w(\gamma_i) + p \cdot (w(\theta_{ij}) + w(\theta_{ji})).$$

Since we assumed that $w(\gamma_i) + w(\gamma_j) < 0$, by choosing a sufficiently large n, we obtain a negative cycle in $\mathcal{G}_R^\omega$. Thus, R is not $*$-consistent, contradiction. □

We continue with a technical lemma.

Lemma 5.24 *Let $G = \langle V, E, w \rangle$ be a weighted digraph, and $u, v \in V$ be two vertices. Let $\theta_1 = \sigma_1.\lambda_1^*.\sigma_1'$ be a basic path scheme and $\rho_1 = \sigma_1.\lambda_1^{b_1}.\sigma_1'$, $b_1 \geq 0$ be a minimal path from u to v such that $|\rho_1| \geq \|V\|^4$. Further, let $\theta_2 = \sigma_2\lambda_2^*.\sigma_2'$ be a path scheme (not necessarily basic), and $\rho_1' = \sigma_1.\lambda_1^{b_1'}.\sigma_1'$ and $\rho_2' = \sigma_2\lambda_2^{b_2'}.\sigma_2'$, where $b_1' > b_1$ and $b_2' \geq 0$, be paths from u to v, and let $L > 0$ be integer such that*

$$\begin{array}{ll} |\rho_1| + L = |\rho_1'| = |\rho_2'|, & \overline{w}(\lambda_1) = \overline{w}(\lambda_2), \\ |\lambda_1| \text{ and } |\lambda_2| \text{ divide } L, & w(\rho_1') = w(\rho_2'). \end{array}$$

Then, there exists a basic path scheme $\theta_3 = \sigma_3.\lambda_2^*.\sigma_3'$ and a path $\rho_3 = \sigma_3.\lambda_2^{b_3}.\sigma_3'$, $b_3 \geq 0$ from u to v such that $w(\rho_3) = w(\rho_1)$ and $|\rho_3| = |\rho_1|$.

Proof: We can use the same techniques as in the proofs of Lemma 4.3 and 4.4 and for a given path ρ'_2, we construct a basic path scheme $\theta_3 = \sigma_3.\lambda_2^*.\sigma'_3$ and a path $\rho'_3 = \sigma_3.\lambda_2^{b'_3}.\sigma'_3$, $b'_3 \geq 0$ from u to v such that $|\rho'_3| = |\rho'_2|$ and $w(\rho'_3) \leq w(\rho'_2)$. Thus, $w(\rho'_2) = w(\rho'_3) + D$ for some $D \geq 0$. Observe that

$$|\sigma_3.\sigma'_3| \leq \|V\|^4 \leq |\rho_1|.$$

By requirements of the lemma and by construction of ρ'_3, $|\rho_1| + L = |\rho'_1| = |\rho'_2| = |\rho'_3|$. Since $|\lambda_2|$ divides L, there exists a path $\rho_3 = \sigma_3.\lambda_2^{b_3}.\sigma'_3$, where $b_3 \geq 0$, such that $|\rho_3| = |\rho_1|$. Furthermore,

$$\begin{array}{rcl} w(\rho_1) & = & w(\rho'_1) - \overline{w}(\lambda_1) \cdot L, \text{ and} \\ w(\rho_3) & = & w(\rho'_3) - \overline{w}(\lambda_2) \cdot L. \end{array}$$

The lemma requires that $w(\rho'_1) = w(\rho'_2)$. Combining it with $w(\rho'_2) = w(\rho'_3) + D$, we obtain that $w(\rho'_1) = w(\rho'_3) + D$. The lemma also requires that $\overline{w}(\lambda_2) = \overline{w}(\lambda_1)$. Thus,

$$w(\rho'_1) - \overline{w}(\lambda_1) \cdot L = w(\rho'_3) - \overline{w}(\lambda_2) \cdot L + D$$

and consequently, $w(\rho_1) = w(\rho_3) + D$. Clearly, $D > 0$ would contradict that ρ_1 is minimal, thus we conclude that $D = 0$ and thus $w(\rho_1) = w(\rho_3)$. □

We finally prove the existence of a basic path scheme where the cycle λ is simple.

Lemma 5.25 *Let $T_R = (Q, \Delta, w)$ be the common transition table of zigzag automata defined for a difference bounds relation $R(\mathbf{x}, \mathbf{x}')$, and $u, v \in Q$ be two control states. Then for every minimal weight path ρ form u to v, such that $|\rho| \geq \|R\| \cdot \|Q\|^6$, there exists a path ρ' from u to v, such that $w(\rho) = w(\rho')$ and $|\rho| = |\rho'|$, and a basic path scheme $\theta = \sigma \cdot \lambda^* \cdot \sigma'$, such that λ is simple, $\rho' = \sigma \cdot \lambda^b \cdot \sigma'$, for some $b \geq 0$. Moreover, there exists $c \mid \frac{lcm(1,\ldots,\|Q\|-1)}{|\lambda|}$ such that $\sigma \cdot \lambda^{b+kc} \cdot \sigma'$ is a minimal weight path from u to v, for all $k \geq 0$.*

Proof: By Lemma 5.1, there exists a basic path scheme $\theta_1 = \sigma_1\lambda_1^*\sigma'_1$ and a path $\rho_1 = \sigma_1\lambda_1^{b_1}\sigma'_1$, $b_1 \geq 0$ from u to v that have all the properties of this lemma except that λ_1 might not be simple. First, we build a path scheme $\theta_2 = \sigma_2\lambda_2^*\sigma'_2$ which might not be basic, but where λ_2 is simple, $w(\lambda_2) = w(\lambda_1)$, $|\lambda_2| = |\lambda_1|$, and $w(\sigma_1.\lambda_1^{b_1+q+k}) = w(\sigma_2.\lambda_2^k.\sigma'_2)$ for some $q > 0$ and for all $k \geq 0$. In other words, if θ_1 is followed by a minimal path of length $L \geq |\sigma_2.\sigma'_2|$, then θ_2 is followed by a minimal path of length L too.

Let $\mathbf{z}_1, \ldots, \mathbf{z}_m$ be equivalence classes of $\sim$. Let q be a control state of the zigzag automaton such that λ_1 is a cycle that starts and ends in q. Let $L, R \subset \{1, \ldots, N\}$ be the set of l-indices and r-indices of q, respectively, and partition them according to equivalence classes $\mathbf{z}_1, \ldots, \mathbf{z}_m$, thus obtaining $L_1, \ldots, L_m$ and $R_1, \ldots, R_m$. For each $1 \leq i \leq m$, let $\rho_{i,1}, \ldots, \rho_{i,m_i}$ denote the set of repeating paths encoded in λ_1 that cross variables in $\mathbf{z}_i$. Finally, let Θ' denote the paths in σ'_1 that connect r-indices with l-indices and Θ denote the paths in σ_1 that connect l-indices with r-indices. Let $\Theta(i) \subseteq \Theta$ and $\Theta'(i) \subseteq \Theta'$ be paths that cross only variables in $\mathbf{z}_i$. We define $p = \max\left\{\lceil\frac{|\sigma_1|}{|\lambda_1|}\rceil, \lceil\frac{|\sigma'_1|}{|\lambda_1|}\rceil\right\}$. Let $\rho_{i,1} \prec \cdots \prec \rho_{i,m_i}$ be the order in which subpaths $\rho_{i,1}, \ldots, \rho_{i,m_i}$ are visited in the

path encoded in $\sigma_1.\lambda_1.\sigma'_1$. Then, by construction of zigzag automata, $\rho_{i,j}$ is forward if and only if $\rho_{i,j+1}$ is backward. Figures 5.7(a) and (b) illustrate these definitions.

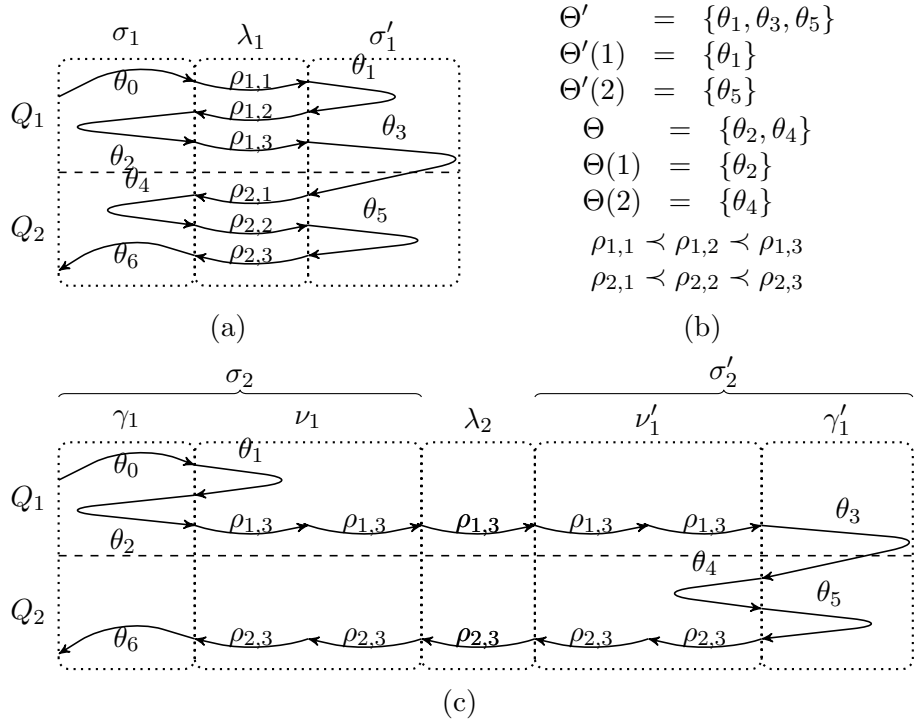


Figure 5.7: Illustration of a construction of a path scheme with simple cycle.

Given an equivalence class $\mathbf{z}_i$, $1 \leq i \leq m$, we first give a construction ensuring that there is at most one repeating path in λ_2 that crosses variables in $\mathbf{z}_i$.

We next build $\gamma, \nu, \lambda_2, \nu', \gamma'$, which are initialized as follows: $\gamma = \sigma_1$, $\nu = (\lambda_1)^p$, $\lambda_2 = \lambda_1$, $\nu' = (\lambda_1)^p$, $\gamma' = \sigma'_1$. Further, we erase all paths from ν, λ_2, ν' that cross some variable in $\mathbf{z}_i$. We finish construction of $\gamma, \nu, \lambda_2, \nu', \gamma'$ for one of the following cases (note that cases 1 and 2 (3 and 4) are symmetrical):

1. $\rho_{i,1}$ **is fw and** m_i **is even.** Erase $\Theta(i)$ from γ' and add it to ν.
2. $\rho_{i,1}$ **is bw and** m_i **is even.** Erase $\Theta'(i)$ from γ and add it to ν'.
3. $\rho_{i,1}$ **is fw and** m_i **is odd.** Do the actions for Case 1. Further, add ρ_{i,m_i} to λ_2 and add $(\rho_{i,m_i})^p$ to both ν and ν'.
4. $\rho_{i,1}$ **is bw and** m_i **is odd.** Do the actions for Case 2. Further, add ρ_{i,m_i} to λ_2 and add $(\rho_{i,m_i})^p$ to both ν and ν'.

The construction is finished by building $\sigma_2 = \gamma.\lambda_1^{b_1}\nu$ and $\sigma'_2 = \nu'.\gamma'$. Figure 5.7(c) illustrates the construction: Case 3 applies for the equivalence class $\mathbf{z}_1$, while the Case 4 applies for $\mathbf{z}_2$.

We prove several properties on weights of $\sigma_2, \lambda_2, \sigma_2'$ for case 3 (proofs for cases 1, 2, and 4 are similar). Let us define

$$W = \sum_{\theta \in \Theta(i)} w(\theta) \qquad V = \sum_{1 \le j < m_i} w(\rho_{i,j})$$

Following equalities follow from the construction:

$$|\sigma_2.\sigma_2'| - |\sigma_1.\sigma_1'| = |\lambda_1^{b_1}| + |\nu| + |\nu'| = |\lambda_1| \cdot (b_1 + 2p)$$

$$\begin{array}{rclcrcl} w(\gamma) & = & w(\sigma_1) & & w(\nu') & = & p \cdot (w(\lambda_1) - V) \\ w(\nu) & = & p \cdot (w(\lambda_1) - V) + W & & w(\gamma') & = & w(\sigma_1') - W \end{array}$$

Then,

$$\begin{array}{rcl} w(\sigma_2.\sigma_2') = w(\gamma.\lambda_1^{b_1}.\nu.\nu'.\gamma') & = & w(\sigma_1) + (b_1 + 2p) \cdot w(\lambda_1) + w(\sigma_1') - 2pV, \\ w(\sigma_1.\lambda_1^{b_1+2p}.\sigma_1') & = & w(\sigma_1) + (b_1 + 2p) \cdot w(\lambda_1) + w(\sigma_1'). \end{array}$$

For Case 3, m_i is odd. Since $\rho_{i,j}$ is fw and $\rho_{i,j+1}$ is bw for odd $j < m_i$, by Lemma 5.23, $w(\rho_j) + w(\rho_{j+1}) \ge 0$ and thus $V \ge 0$. By construction, $w(\lambda_2) = w(\lambda_1) - V$. Since $V > 0$ would imply that $w(\lambda_2) < w(\lambda_1)$ and thus, that $\sigma_1.\lambda_1^{b_1+k}.\sigma_1'$ is not minimal for all $k \ge 0$, we infer that $V = 0$ and therefore, $w(\lambda_2) = w(\lambda_1)$ and

$$w(\sigma_2.\sigma_2') = w(\sigma_1.\lambda_1^{b_1+2p}.\sigma_1').$$

Note that the above construction of σ_2, λ_2, and σ_2' can be easily extended to deal with all equivalence classes of $\sim$ at once. Then, λ_2 is simple and the above equality still hold. Note that if the construction steps for Case 3 or Case 4 generate a path (encoded in ν or ν') with cycles, we erase all the cycles. Their weights must be non-negative, since we consider $*$-consistent relations. They also cannot be strictly positive since then $\sigma_1.\lambda_1.\sigma_1'$ would not be minimal for all $k \ge b_1 + 2p$, contradiction. Thus erasing them changes weight of neither σ_2 nor σ_2' and hence the above equality still hold. Let us define $b_2' = 0$, $b_1' = b_1 + 2p$, $\rho_2' = \sigma_2.\lambda_2^{b_2'}\sigma_2'$, $\rho_1' = \sigma_1.\lambda_1^{b_1'}\sigma_1'$ and observe that

$$\begin{array}{rcl} L = |\rho_2'| - |\rho_1| & = & (|\sigma_2.\sigma_2'| - |\sigma_1.\sigma_1'|) + |\lambda_2^{b_2'}| - |\lambda_1^{b_1}| \\ & = & |\lambda_1| \cdot (b_1 + 2p) + 0 - |\lambda_1| \cdot b_1 \\ & = & |\lambda_1| \cdot (b_1 + 2p - b_1) = |\lambda_1| \cdot 2p \end{array}$$

and thus, $|\lambda_1| = |\lambda_2|$ divides L. Next, we apply Lemma 5.24 which guarantees existence of a path scheme $\theta_3 = \sigma_3.\lambda_2^*.\sigma_3'$ and a path $\rho_3 = \sigma_3.\lambda_2^{b_3}.\sigma_3'$, for some $b_3 \ge 0$, such that $w(\rho_3) = w(\rho_1)$ and $|\rho_3| = |\rho_1|$. The existence of $c \mid \frac{\mathrm{lcm}\,1,\ldots,\|V\|}{|\lambda_2|}$ such that $\sigma_3.\lambda_2^{b_3+kc}.\sigma_3'$ is minimal for all $k \ge 0$ follows from the proof of Lemma 5.1, since we can choose $\pi_i.\lambda_i^*.\pi_i' = \theta_3$. □

Basic Path Schemes with Cycles Bounded by $\mathrm{lcm}(1,\ldots,N)$

This section refines the statement of Lemma 5.1 by proving that the length of the cycle λ from each basic path scheme divides $\mathrm{lcm}(1,\ldots,N)$. This fact is essential in proving the single exponential bound on the period of octagonal relations. We begin with a lemma that is later used to deal with the problem of synchronization of connecting paths which was discussed earlier. Recall that for a path $\rho : x_{i_0}^{(\ell_0)} \to \ldots \to x_{i_m}^{(\ell_m)}$ in $\mathcal{G}_R^\omega$ such that $i_0 = i_m$ and $\ell_m \neq \ell_0$, we defined its left (right) extent relative to $x_{i_0}^{(\ell_0)}$ as

$$L_\rho = |\min\{\ell_k - \ell_0 \mid 0 \le k \le m\}| \qquad \overline{L}_\rho = \left\lceil \frac{L_\rho}{\|\rho\|} \right\rceil$$
$$R_\rho = |\max\{\ell_k - \ell_0 \mid 0 \le k \le m\}| \qquad \overline{R}_\rho = \left\lceil \frac{R_\rho}{\|\rho\|} \right\rceil$$

Lemma 5.26 *Let $\tau' = \mu^r.\xi.\nu^s.\zeta.\mu^t$ be a connecting path such that*

$$r = r_1 + r_2 \quad s = s_1 + \cdots + s_5 \quad t = t_1 + t_2, \text{ where}$$

$$\begin{array}{lcl@{\qquad}lcl} s_1 & \ge & L_\nu + R_\mu + N + 1 & t_1 & \ge & L_\mu + R_\nu + N + 1 \\ s_5 & \ge & R_\nu + L_\mu + N + 1 & r_2 & \ge & R_\mu + L_\nu + N + 1 \end{array}$$

$$\begin{array}{lcl@{\qquad}lcl@{\qquad}lcl} s_2 & \ge & \overline{R}_\nu - 1 & & & & t_2 & \ge & 1 + \overline{R}_\nu \\ s_4 & \ge & \overline{L}_\nu & s_3 & \ge & 2N & r_1 & \ge & 1 + \overline{L}_\nu \end{array}$$

Let τ be a path built by erasing all cycles from τ' and let

- *λ_μ be a zigzag cycle that encodes μ,*
- *λ_ν be a zigzag cycle that encodes ν,*
- *λ_τ be a zigzag cycle that encodes τ.*

Then, λ_τ can be written as $\lambda_\tau = \lambda_\mu.\pi_1.\lambda_\nu'^{\,s_3-2N}.\pi_2.\lambda_\mu$ where π_1, π_2 are some zigzag paths, $|\lambda_\nu| = |\lambda_\nu'|$, $w(\lambda_\nu) = w(\lambda_\nu')$, and $|\lambda_\mu.\pi_1| = \|\mu^{r_1+r_2}\| + \|\nu^{s_1+s_2}\| + N$.

Proof: Let us denote the subpaths of τ' as

$$\begin{array}{rcccccccccccccccccccc} & & \alpha_1 & & \alpha_2 & & \alpha_3 & & \alpha_4 & & \alpha_5 & & \alpha_6 & & \alpha_7 & & \alpha_8 & & \alpha_9 & & \alpha_{10} & & \alpha_{11} \\ & & \| & & \| & & \| & & \| & & \| & & \| & & \| & & \| & & \| & & \| & & \| \\ \tau' & = & \mu^{r_1} & . & \mu^{r_2} & . & \xi & . & \nu^{s_1} & . & \nu^{s_2} & . & \nu^{s_3} & . & \nu^{s_4} & . & \nu^{s_5} & . & \zeta & . & \mu^{t_1} & . & \mu^{t_2} \end{array}$$

First, we show that the constraint on s_1 ensures that subpaths $\alpha_1.\alpha_2.\alpha_3$ and α_5 do not share a node. Let $M_1 = \|\alpha_1.\alpha_2\|$ and observe that

$$vars(\alpha_1.\alpha_2) \subseteq \bigcup_{k=-L_\mu}^{M_1+R_\mu} \mathbf{x}^{(k)}, \qquad vars(\alpha_3) \subseteq \bigcup_{k=M_1-N}^{M_1+N} \mathbf{x}^{(k)}.$$

The property on $vars(\alpha_1.\alpha_2)$ holds since $L_{\rho^k} \leq L_\rho$ and $R_{\rho^k} \leq R_\rho$ for each path ρ and $k \geq 1$. The property on $vars(\alpha_3)$ follows from the fact that ξ is essential. Hence,

$$vars(\alpha_1.\alpha_2.\alpha_3) \subseteq \bigcup_{k=\min(-L_\mu,-N)}^{M_1+\max(R_\mu,N)} \mathbf{x}^{(k)} \subseteq \bigcup_{k=\min(-L_\mu,-N)}^{M_1+R_\mu+N} \mathbf{x}^{(k)}.$$

Similarly, letting $M_2 = M_1 + \|\alpha_3.\alpha_4\|$, we observe that

$$vars(\alpha_5) \subseteq \bigcup_{k=M_2-L_\nu}^{M_2+\|\alpha_5\|+R_\nu} \mathbf{x}^{(k)}.$$

We can now infer a condition that guarantees that subpaths $\alpha_1.\alpha_2.\alpha_3$ and α_5 do not share a node:

$$\begin{array}{rcll} M_1 + R_\mu + N & < & M_2 - L_\nu & \\ R_\mu + N & < & \|\alpha_3.\alpha_4\| - L_\nu & \\ \|\alpha_4\| & > & L_\nu + R_\mu + N - \|\alpha_3\| & \\ \|\alpha_4\| & \geq & L_\nu + R_\mu + N + 1 & \text{(sufficient, since } \|\alpha_3\| \geq 0) \\ s_1 & = & L_\nu + R_\mu + N + 1 & \text{(sufficient, since } \|\alpha_4\| = \|\nu\| \cdot s_1) \end{array}$$

Similarly, one infers constraints on s_5, t_1, and r_2. These constrains guarantee that all sharings of nodes (in other words, cycles) may occur only in $\alpha_2, \alpha_4, \alpha_8$, or α_{10} and thus, that no cycle appears in $\alpha_5.\alpha_6.\alpha_7$, α_1 and α_{11}.

Next, we prove an auxiliary statement that λ_τ can be written as $\lambda_\tau = \lambda_\mu.\pi_1.\lambda_2^{s_3}.\pi_2.\lambda_\mu$ where π_1, π_2 are some zigzag paths, and $|\lambda_\mu.\pi_1| = \|\mu^{r_1+r_2}.\xi\nu^{s_1+s_2}\|$. Figures 5.8 and 5.9 illustrate the proof.

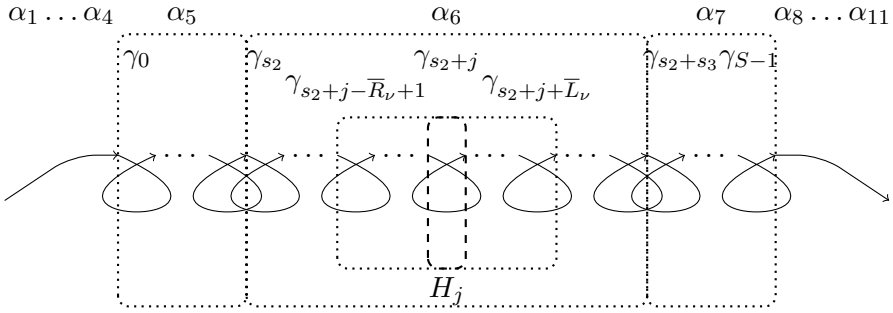
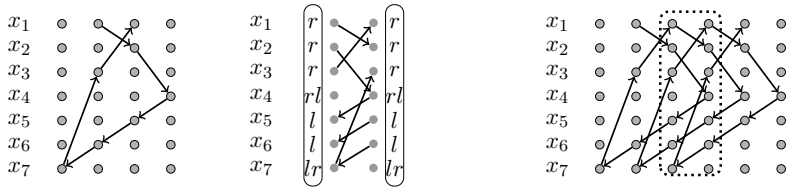


Figure 5.8: $\lambda_\nu^{s_3}$ as a subpath of λ_τ.



(a) path ν (b) zigzag cycle λ_ν (c) H_j, $\gamma_{s_2+j-1}, \gamma_{s_2+j}, \gamma_{s_2+j+1}$

Figure 5.9: Obtaining λ_ν by iterating ν. $\overline{L}_\nu = 1, \overline{R}_\nu = 2$.

Let $G_0G_1\dots G_{|\lambda_\tau|-1}$ be the labeling of λ_τ and let $\alpha_5.\alpha_6.\alpha_7 = \gamma_0.\gamma_1\dots\gamma_{S-1}$, where $S = s_2 + s_3 + s_4$ and $\nu = \gamma_0 = \gamma_1 = \dots = \gamma_{S-1}$. For each $0 \leq j < s_3$, define $K_j = \|\alpha_1\dots\alpha_5.\nu^j\|$, $H_j = G_{K_j+1}\dots G_{K_j+\|\nu\|}$ and observe that for each $-\overline{R}_\nu < k \leq \overline{L}_\nu$, γ_{s_2+j+k} contributes[1] to H_j with $\Theta_{\nu,k}$. Thus, H_j consists of zigzag-segments

$$\bigcup_{j=-\overline{L}_\nu+1}^{\overline{R}_\nu} \Theta_{\nu,j}$$

which are clearly zigzag-segments of λ_ν. Thus, H_j is the labeling of λ_ν for each $0 \leq j < s_3$ and consequently, λ_τ can be decomposed into $\sigma_1.\lambda_\nu^{s_3}.\sigma_2$ for some paths σ_1, σ_2. Moreover, since $K_0 = \|\alpha_1\dots\alpha_5\| = \|\mu^{r_1+r_2}.\xi\nu^{s_1+s_2}\|$, then

$$|\sigma_1| = K_0 = \|\mu^{r_1+r_2}.\xi\nu^{s_1+s_2}\|.$$

By a similar argument, one can show that λ_τ can be decomposed into $\lambda_\mu.\sigma_3.\lambda_\mu$ for some path σ_3, by viewing the path $\tau' = \alpha_1\dots\alpha_{11}$ shifted as $\alpha_6\dots\alpha_{11}\alpha_1\dots\alpha_5$. Combining decompositions $\sigma_1.\lambda_\nu^{s_3}.\sigma_2$ and $\lambda_\tau = \lambda_\mu.\sigma_3.\lambda_\mu$, we obtain the required decomposition $\lambda_\tau = \lambda_\mu.\pi_1.\lambda_2^{s_3}.\pi_2.\lambda_\mu$ for some π_1, π_2, where

$$|\lambda_\mu.\pi_1| = |\sigma_1| = \|\mu^{r_1+r_2}.\xi\nu^{s_1+s_2}\|.$$

Finally, we prove that λ_τ can be written as $\lambda_\tau = \lambda_\mu.\pi_1.{\lambda'_\nu}^{s_3-2N}.\pi_2.\lambda_\mu$ where π_1, π_2 are some zigzag paths, $|\lambda_\nu| = |\lambda'_\nu|$, $w(\lambda_\nu) = w(\lambda'_\nu)$, and $|\lambda_\mu.\pi_1| = \|\mu^{r_1+r_2}\| + \|\nu^{s_1+s_2}\| + N$. Let us define

$$D_0 = \|\mu^{r_1+r_2}\| + \|\nu^{s_1+s_2}\|,\ \ D_1 = \|\mu^{r_1+r_2}.\xi.\nu^{s_1+s_2}\|,\ \ D_2 = \|\mu^{r_1+r_2}\| + \|\nu^{s_1+s_2}\| + N.$$

Since ξ is essential, $D_2 \geq D_1$ and $D_0 - N \leq D_1 \leq D_0 + N$. Thus, by noticing that $D_2 = D_0 + N$, we infer that $0 \leq D_2 - D_1 \leq 2N$. Clearly, there exists $0 \leq d < \|\nu\|$ and $k \geq 0$ such that $D_2 + d = D_1 + k\|\nu\|$. Thus, $k = \lceil \frac{D_2-D_1}{\|\nu\|} \rceil$. Combining this with $0 \leq D_2 - D_1 \leq 2N$, we establish a bound $k \leq 2N$.

Next, observe that a decomposition of λ_τ into $\lambda_\mu.\pi_3.\lambda_\nu^{s_3-k}.\pi_4.\lambda_\mu$, where $|\lambda_\mu.\pi_3| = D_2 + d = D_1 + k\|\nu\|$, is possible by a similar argument as previously. The only difference is that j ranges over $k \leq j < s_3$ instead of $0 \leq j < s_3$. Thus, we need to guarantee that $s_3 - k \geq 0$. This can be achieved by requiring that $s_3 \geq 2N$, since the maximal value of k is $k = 2N$. This gives the stricter condition on s_3 in this lemma that guarantees the decomposition into $\lambda_\mu.\pi_3.\lambda_\nu^{s_3-2N}.\pi_4.\lambda_\mu$, where $|\lambda_\mu.\pi_3| = D_2 + d = D_1 + k\|\nu\|$.

Let $G_1\dots G_n$ be the labeling of λ_ν. The decomposition in the previous paragraph implies that the label of $(\lambda_\tau)_{D_2+d\dots|\lambda_\tau|-1}$ has a prefix $(G_1\dots G_n)^{s_3-2N}$. Further, the label of $(\lambda_\tau)_{D_1\dots|\lambda_\tau|-1}$ has a prefix $(G_1\dots G_n)^{s_3}$. Since $D_2 + d - D_1 = k\|\nu\|$, it follows that the label of $(\lambda_\tau)_{D_1\dots D_2-1}$ is $(G_1\dots G_n)^k$. Furthermore, we infer that the label of $(\lambda_\tau)_{D_2\dots|\lambda_\tau|-1}$ has a prefix $(G_{d+1}\dots G_nG_1\dots G_d)^{s_3-2N}$. Since $G_1\dots G_n$ is the labeling of the cycle $\lambda_\nu : q \xrightarrow{G_1\dots G_d} q' \xrightarrow{G_{d+1}\dots G_n} q$, there clearly exists a cycle $\lambda'_2 : q' \xrightarrow{G_{d+1}\dots G_n}$

[1]Hence the bounds on s_2 and s_4: $s_2 \geq \overline{R}_\nu - 1$, $s_4 \geq \overline{L}_\nu$.

$q \xrightarrow{G_1 \ldots G_d} q'$. Thus, λ can be decomposed into $\lambda_\tau = \sigma_1.(\lambda'_\nu)^{s_3-2N}.\sigma_2$, where $|\sigma_1| = D_2 = \|\mu^{r_1+r_2}\| + \|\mu^{s_1+s_2}\| + N$. Clearly, $|\lambda_\nu| = |\lambda'_\nu|$ and $w(\lambda_\nu) = w(\lambda'_\nu)$. □

Informally, the following technical proposition states that the relative lengths of two repeating paths can be synchronized by iterating each repeating path with itself several times.

Proposition 5.27 *Let γ_1, γ_2 be repeating paths and let $c_1, c_2 \geq 1$. Then, there exists $c'_1 = c_1 \cdot k_1$, $c'_2 = c_2 \cdot k_2$ for some $k_1, k_2 \geq 1$ such that $\|\gamma_1^{c'_1}\| = \|\gamma_2^{c'_2}\|$.*

Proof: Let $L = \mathrm{lcm}(c_1 \cdot \|\gamma_1\|, c_2 \cdot \|\gamma_2\|)$, $c'_1 = \frac{L}{\|\gamma_1\|}$, $c'_2 = \frac{L}{\|\gamma_2\|}$. Since $c_1 \cdot \|\gamma_1\|$ divides L, then c_1 divides $\frac{L}{\|\gamma_1\|}$ too and thus, there exists $k_1 \geq 1$ such that $c_1 \cdot k = c'_1$. Similarly, there exists $k_2 \geq 1$ such that $c_2 \cdot k = c'_2$. Further, $\|\gamma_1^{c'_1}\| = \|\gamma_2^{c'_2}\| = L$, since $\|\gamma_1^{c'_1}\| = \|\gamma_1\| \cdot c'_1 = \|\gamma_1\| \cdot \frac{L}{\|\gamma_1\|} = L$ and similarly, $\|\gamma_2^{c'_2}\| = L$. □

We finally prove that we can, without loss of generality, consider basic path schemes with cycles that are simple and moreover, the length of which divides $\mathrm{lcm}(1, \ldots, N)$. For the proof of the lemma, we need the notion of *optimal* cycle.

Definition 5.28 *a simple cycle λ is* optimal *if and only if each forward path encoded in λ is fw-optimal and each backward path encoded in λ is bw-optimal.*

Lemma 5.29 *Let $T_R = (Q, \Delta, w)$ be the common transition table of zigzag automata defined for a difference bounds relation $R(\mathbf{x}, \mathbf{x}')$, and $u, v \in Q$ be two control states. Then for every minimal weight path ρ form u to v, such that $|\rho| \geq \|R\| \cdot \|V\|^6$, there exists a path ρ' from u to v, such that $w(\rho) = w(\rho')$ and $|\rho| = |\rho'|$, and a basic path scheme $\theta = \sigma \cdot \lambda^* \cdot \sigma'$, such that λ is simple and $|\lambda|$ divides $\mathrm{lcm}(1, \ldots, N)$, $\rho' = \sigma \cdot \lambda^b \cdot \sigma'$, for some $b \geq 0$. Moreover, there exists $c \mid \frac{\mathrm{lcm}(1,\ldots,N)}{|\lambda|}$ such that $\sigma \cdot \lambda^{b+kc} \cdot \sigma'$ is a minimal weight path from u to v, for all $k \geq 0$.*

Proof: By Lemma 5.25, there exists a basic path scheme $\theta_1 = \sigma_1.\lambda_1^*.\sigma'_1$ where λ_1 is simple and a path $\rho_1 = \sigma_1.\lambda_1^{b_1}.\sigma'_1$, for some $b_1 \geq 0$, such that $w(\rho_1) = w(\rho)$ and $|\rho_1| = |\rho|$. In this proof, we assume that λ_1 encodes two forward paths μ_1, μ_2 and no backward path. The extension to arbitrary number of forward and backward paths is straightforward. Let μ_j be of the form $\mu_j : x_{i_j} \rightsquigarrow x_{i_j}$ for each $j \in \{1, 2\}$. and let us denote the equivalence class of x_{i_j} as $\mathbf{z}_j = [x_{i_j}]_\sim$.

Case 1: λ_1 is optimal. By Proposition 5.17, given μ_j, there exists $\nu_j \in P^c_\triangleright(\mathbf{z}_j)$ of the form $\nu_j : x_{k_j} \rightsquigarrow x_{k_j}$ and two essential paths $\xi_j : x_{i_j} \rightsquigarrow x_{k_j}$ and $\zeta_j : x_{k_j} \rightsquigarrow x_{i_j}$. For each $j \in \{1, 2\}$, we build a connecting path τ'_j as follows:

$$
\begin{array}{rcccccccccccccccccccc}
 & & \alpha_1 & & \alpha_2 & & \alpha_3 & & \alpha_4 & & \alpha_5 & & \alpha_6 & & \alpha_7 & & \alpha_8 & & \alpha_9 & & \alpha_{10} & & \alpha_{11} \\
 & & \| & & \| & & \| & & \| & & \| & & \| & & \| & & \| & & \| & & \| & & \| \\
\tau'_1 & = & \mu_1^{r_1} & . & \mu_1^{r_2} & . & \xi_1 & . & \nu_1^{s_1} & . & \nu_1^{s_2} & . & \nu_1^{s_3} & . & \nu_1^{s_4} & . & \nu_1^{s_5} & . & \zeta_1 & . & \mu_1^{t_1} & . & \mu^{t_2} \\
\tau'_2 & = & \mu_2^{t_3} & . & \mu_2^{t_4} & . & \xi_2 & . & \nu_2^{w_1} & . & \nu_2^{w_2} & . & \nu_2^{w_3} & . & \nu_2^{w_4} & . & \nu_2^{w_5} & . & \zeta_2 & . & \mu_2^{t_1} & . & \mu^{t_2} \\
 & & \| & & \| & & \| & & \| & & \| & & \| & & \| & & \| & & \| & & \| & & \| \\
 & & \beta_1 & & \beta_2 & & \beta_3 & & \beta_4 & & \beta_5 & & \beta_6 & & \beta_7 & & \beta_8 & & \beta_9 & & \beta_{10} & & \beta_{11}
\end{array}
$$

In addition to the conditions of Lemma 5.26, we require that $|\alpha_k| = |\beta_k|$ for all $k \in \{1, 2, 4, 5\}$. These additional constraints can be satisfied too, by Proposition 5.27. Clearly, the new coefficients still satisfy the conditions in Lemma 5.26. Finally, we define $s_3 = 2N + \frac{L}{\|\nu_1\|}$, $w_3 = 2N + \frac{L}{\|\nu_2\|}$, where $L = \text{lcm}(\|\nu_1\|, \|\nu_2\|)$.

By Lemma 5.26, there exists a zigzag cycles λ_{τ_1} and λ_{τ_2} that encode τ_1 and τ_2 that were obtained by erasing all cycles in τ_1' and τ_2', respectively, and that can be decomposed into

$$\begin{aligned} \lambda_{\tau_1} &= \lambda_{\mu_1}.\pi_1.{\lambda'_{\nu_1}}^{s_3-2N}.\pi_2.\lambda_{\mu_1} \\ \lambda_{\tau_2} &= \lambda_{\mu_2}.\pi_3.{\lambda'_{\nu_2}}^{w_3-2N}.\pi_4.\lambda_{\mu_2} \end{aligned}$$

where

$$|\lambda'_{\nu_j}| = |\lambda_{\nu_j}| \text{ and } w(\lambda'_{\nu_j}) = w(\lambda_{\nu_j}) \text{ and}$$

$$\|\lambda_{\mu_1}.\pi_1\| = \|\alpha_1.\alpha_2\| + \|\alpha_4.\alpha_5\| + N \text{ and } \|\beta_1.\beta_2\| + \|\beta_4.\beta_5\| + N = \|\lambda_{\mu_2}.\pi_3\|$$

Since $|\alpha_k| = |\beta_k|$ for all $k \in \{1, 2, 4, 5\}$, we infer that $\|\lambda_{\mu_1}.\pi_1\| = \|\lambda_{\mu_2}.\pi_3\|$.

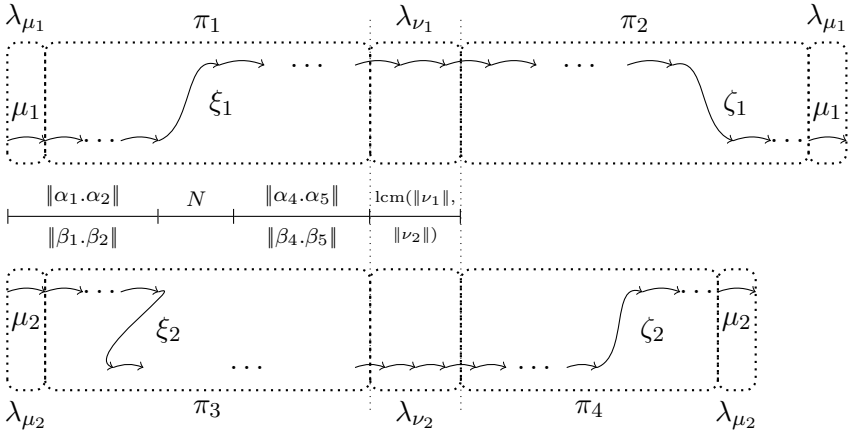


Figure 5.10: Synchronization of connecting paths τ_1 and τ_2.

Note that $|\lambda_{\tau_1}| = |\lambda_{\tau_2}|$ is not true in general. For this reason, we need to make an extra step. Let $M = \text{lcm}(|\lambda_{\tau_1}|, |\lambda_{\tau_2}|)$, $m_1 = \frac{M}{|\lambda_{\tau_1}|}$, $m_2 = \frac{M}{|\lambda_{\tau_2}|}$. Since $|\lambda_{\tau_1}^{m_1}| = |\lambda_{\tau_2}^{m_2}|$, and since the paths in λ_{τ_1} and λ_{τ_2} use disjoint variables, we can build a cycle λ by gluing $\lambda_{\tau_1}^{m_1}$ with $\lambda_{\tau_2}^{m_2}$ and obtain $\lambda = \lambda_1.\pi_1'.\lambda_2'.\pi_2'.\lambda_1$, where

$$\lambda_1 = \begin{bmatrix} \lambda_{\mu_1} \\ \lambda_{\mu_2} \end{bmatrix} \pi_1' = \begin{bmatrix} \pi_1 \\ \pi_3 \end{bmatrix} \lambda_2' = \begin{bmatrix} {\lambda'_{\nu_1}}^{s_3-2N} \\ {\lambda'_{\nu_2}}^{w_3-2N} \end{bmatrix} \pi_2' = \begin{bmatrix} \pi_2.\lambda_{\mu_1}.(\lambda_{\tau_1})^{m_1-1} \\ \pi_4.\lambda_{\mu_2}.(\lambda_{\tau_2})^{m_2-1} \end{bmatrix}$$

Note that the construction of λ_2' is correct since

$$|{\lambda'_{\nu_1}}^{s_3-2N}| = |\lambda_{\nu_1}^{s_3-2N}| = \|\nu_1\|(s_3 - 2N) = \|\nu_1\| \left(\frac{L}{\|\nu_1\|} + 2N - 2N\right) = \text{lcm}(\|\nu_1\|, \|\nu_2\|)$$

and similarly, $|{\lambda'_{\nu_2}}^{w_3-2N}| = \text{lcm}(\|\nu_1\|, \|\nu_2\|)$. Thus, $|\lambda_2'| = \text{lcm}(\|\nu_1\|, \|\nu_2\|)$. Note that λ_1 is optimal by assumption. Further, λ_2 encodes paths $\nu_1^{s_3}$ and $\nu_2^{w_3}$ which are optimal, by the

fact that ν_1, ν_2 are optimal and by Proposition 5.14. Thus, λ_2 is optimal by construction. Since $\|\nu_1\|, \|\nu_1\| \leq N$, then $|\lambda_2|$ divides $\mathrm{lcm}(1, \ldots, N)$. By Proposition 5.17, $\nu_j \in \mathcal{F}(\mu_j)$ and $\mathcal{F}(\xi_j.\zeta_j) \subseteq \mathcal{F}(\mu_j)$. Thus, $\mathcal{F}(\tau_j) \subseteq \mathcal{F}(\mu_j)$. Since τ_j is forward, then $\overline{\overline{w}}(\tau_j) = C_\triangleright(\mathbf{z}_j)$, by Lemma 5.16. Consequently, τ_j is optimal. By Lemma 5.26, λ_{τ_j} encodes τ'_j that was obtained from τ'_j by erasing all its cycles. These cycles are non-negative, since R is $*$-consistent. Next suppose that at least one is strictly positive. Then,

$$\overline{\overline{w}}(\tau_j) < \overline{\overline{w}}(\tau'_j) = \overline{\overline{w}}(\mu_j) = \overline{\overline{w}}(\nu_j) = C_\triangleright(\mathbf{z}_j).$$

However, by Lemma 5.16, $\overline{\overline{w}}(\tau_j) \geq C_\triangleright(\mathbf{z}_j)$, contradiction. Thus, $\overline{\overline{w}}(\tau_j) = C_\triangleright(\mathbf{z}_j)$ too. Consequently, $\lambda_1, \lambda_2, \lambda$ are optimal too, by Definition 5.28. By Proposition 5.22,

$$\overline{w}(\lambda_1) = \overline{w}(\lambda_2) = \overline{w}(\lambda) = C_\triangleright(\mathbf{z}_1) + C_\triangleright(\mathbf{z}_2).$$

We next construct a path scheme $\theta_2 = \sigma_2.{\lambda'_2}^*.\sigma'_2$, where

$$\sigma_2 = \sigma_1.\lambda_1^{b_1}.\lambda_1.\pi'_1 \qquad \sigma'_2 = \lambda'_2.\pi'_2.\lambda_1.\lambda^{(|\lambda_1|\cdot|\lambda'_2|-1)}.\sigma'_1$$

Next, letting $b'_2 = |\lambda_1|$, we construct $\rho'_2 = \sigma_2.{\lambda'_2}^{b'_2}.\sigma'_2$. Recalling that $\rho_1 = \sigma_1.\lambda_1^{b_1}.\sigma'_1$, we compute

$$\begin{array}{rcl} D = |\rho'_2| - |\rho_1| & = & |\lambda_1.\pi'_1.\lambda'_2.\pi'_2.\lambda_1.\lambda^{|\lambda_1|\cdot|\lambda'_2|-1}| + |{\lambda'_2}^{|\lambda_1|}| \\ & = & |\lambda^{|\lambda_1|\cdot|\lambda'_2|}| + |\lambda_1| \cdot |\lambda'_2| \\ & = & |\lambda_1| \cdot |\lambda'_2| \cdot (|\lambda| + 1) \end{array}$$

Letting $b'_1 = b_1 + \frac{D}{|\lambda_1|} + |\lambda'_2|$, we construct $\rho'_1 = \sigma_1.\lambda_1^{b'_1}.\sigma'_1$. Clearly, $|\rho'_1| = |\rho'_2|$. We infer that

$$\begin{array}{rcl} w(\rho'_2) - w(\rho_1) & = & |\lambda_1| \cdot |\lambda'_2| \cdot |\lambda| \cdot \overline{w}(\lambda) + |\lambda_1| \cdot |\lambda_2| \cdot \overline{w}(\lambda_2) \\ & = & D \cdot (C_\triangleright(\mathbf{z}_1) + C_\triangleright(\mathbf{z}_2)) \\ w(\rho'_1) - w(\rho_1) & = & |\lambda_1| \cdot (\frac{D}{|\lambda_1|} + |\lambda'_2|) \cdot \overline{w}(\lambda_1) \\ & = & D \cdot (C_\triangleright(\mathbf{z}_1) + C_\triangleright(\mathbf{z}_2)) \end{array}$$

and thus, $w(\rho'_2) = w(\rho'_1)$. Clearly, $|\lambda_1|$ and $|\lambda'_2|$ divides D. We apply Lemma 5.24 which guarantees existence of a path scheme $\theta_3 = \sigma_3.{\lambda'_2}^*.\sigma'_3$ and a path $\rho_3 = \sigma_3.{\lambda'_2}^{b_3}.\sigma'_3$, for some $b_3 \geq 0$, such that $w(\rho_3) = w(\rho)$ and $|\rho_3| = |\rho|$. Thus, the lemma holds for $\theta = \theta_3$ and $\rho' = \rho_3$.

Case 2: λ_1 is not optimal. Since $\mu_1 \in P_\triangleright(\mathbf{z}_1)$, then $P^c_\triangleright(\mathbf{z}_1) \neq \emptyset$. Let choose $\nu_1 \in P^c_\triangleright(\mathbf{z}_1)$ and assume its form is $\mu_1 : x_{k_1} \rightsquigarrow x_{k_1}$. Further, let $\xi_1 : x_{i_1} \rightsquigarrow x_{k_1}$, $\zeta_1 : x_{k_1} \rightsquigarrow x_{i_1}$ be arbitrary essential paths. Similarly for μ_2, we construct ν_2, ξ_2, ζ_2. We construct $\lambda, \theta_2, \rho'_1, \rho'_2$ and compute D in the same way as in Case 1. Clearly, λ'_2 is optimal, λ_1 is not optimal and thus $\overline{w}(\lambda_1) > \overline{w}(\lambda'_2) = C_\triangleright(\mathbf{z}_1) + C_\triangleright(\mathbf{z}_2)$. Since $|\lambda_1|$ and $|\lambda'_2|$ divide $|\rho'_2| - |\rho_1|$, then $\sigma_1.\lambda_1^{b_1+ck}.\sigma'_2$ is not minimal for some $k \geq 0$. Contradiction with our assumption on $\theta_1 = \sigma_1.\lambda_1^*.\sigma'_1$.

We have proved that Case 2 is not possible, and that for Case 1, there exists a path $\rho_3 = \sigma_3.{\lambda'_2}^{b_3}.\sigma'_3$ where $w(\rho_3) = w(\rho)$, $|\rho_3| = |\rho|$, λ'_2 is optimal and $|\lambda'_2|$ divides $\mathrm{lcm}(1, \ldots, N)$. Let us denote $\rho' = \rho_3$, $\sigma = \sigma_3$, $\sigma' = \sigma'_3$, $\lambda = \lambda'_2$, $b = b_3$. It remains to prove that there exists $c \mid \frac{\mathrm{lcm}(1,\ldots,N)}{|\lambda'_2|}$ such that $\sigma.\lambda^{b+kc}.\sigma'$ is a minimal path from u to v for all $k \geq 0$. The

proof is almost identical to that of Lemma 4.4. The only difference is that we can now consider only basic path schemes $\sigma.\lambda^*.\sigma'$ where λ is simple and $|\lambda|$ divides $\mathrm{lcm}(1,\ldots,N)$. Thus, the proof can use $\mathrm{lcm}(1,\ldots,N)$ instead of $\mathrm{lcm}(1,\ldots,\|V\|-1)$ everywhere. This in turn implies the existence of $c \mid \mathrm{lcm}(1,\ldots,N)$ such that $w(\sigma_i.\lambda^{kc}.\sigma_i')$ is minimal for all $k \geq 0$. □

The single exponential bound on the period of difference bounds relations follows easily from Lemma 5.29.

Corollary 5.30 *Let $\mathbf{x} = \{x_1,\ldots,x_N\}$ be a set of variables. Given a difference bounds relation $R(\mathbf{x},\mathbf{x}')$, the period of $R(\mathbf{x},\mathbf{x}')$ is bounded by $2^{\mathcal{O}(N)}$.*

Proof: Let $T_R = (Q,\Delta,w)$ be the common transition table of zigzag automata defined for a difference bounds relation $R(\mathbf{x},\mathbf{x}')$ and let c be the period of T_R. By Lemma 5.29, $c \mid lcm(1,\ldots,N)$. Applying Lemma 4.5, it follows that c is bounded by $2^{\mathcal{O}(N)}$. □

We finally summarize the complexity results on difference bounds relations.

Theorem 5.31 *Let $\mathbf{x} = \{x_1,\ldots,x_N\}$ be a set of variables. Given a difference bounds relation $R(\mathbf{x},\mathbf{x}')$, its period is bounded by $2^{O(N)}$ and its prefix is bounded by $\|R\| \cdot 2^{O(N)}$.*

Proof: Follows from Corollary 5.3 and Corollary 5.30. □

5.2 Octagonal Relations

Let $R(\mathbf{x},\mathbf{x}')$ be an octagonal relation and $\overline{R}(\mathbf{y},\mathbf{y}')$ be its difference bounds representation. Using the results on bounds on the prefix (Corollary 5.3) and the period (Corollary 5.30) of $\overline{R}(\mathbf{y},\mathbf{y}')$, we infer, using Lemma 4.20, the bounds on the prefix and period of the relations $R(\mathbf{x},\mathbf{x}')$ itself.

Theorem 5.32 *Let $\mathbf{x} = \{x_1,\ldots,x_N\}$ be a set of variables. Given a relation $R(\mathbf{x},\mathbf{x}') \in \mathcal{R}_{oct}$, its period is bounded by $2^{O(N)}$ and its prefix is bounded by $\|R\|^2 \cdot 2^{O(N)}$.*

Proof: Let $\overline{R}(\mathbf{y},\mathbf{y}')$ be the difference bounds representation of $R(\mathbf{x},\mathbf{x}')$ and let $\mathcal{G}_{\overline{R}} = \langle Q,\Delta,w\rangle$ be the zigzag automaton of $\overline{R}(\mathbf{y},\mathbf{y}')$. It follows immediately from Lemma 5.29 that $\mathcal{G}_{\overline{R}}$ has prefix $b = \mu(\mathcal{G}_{\overline{R}}) \cdot \|Q\|^6 = \|\overline{R}\| \cdot 5^{12N}$ and period $c = \mathrm{lcm}(1,\ldots,2N)$. Consequently, the prefix and period of $\{M^*_{\overline{R}^m}\}_{m\geq 0}$ and of $\overline{R}$ are b and c as well, respectively.

The prefix and the period of R are defined as the prefix and period of the sequence $\{\sigma(R^m)\}_{m\geq 0}$, by Definition 3.6. By definition of σ for octagonal relations given in Section 4.3, $\{\sigma(R^m)\}_{m\geq 0} = \{M^t_{\overline{R^m}}\}_{m\geq 0}$. By Theorem 2.26, $M^t_{\overline{R^m}} = M^t_{\overline{R}^m}$ for all $m \geq 0$ and

$$(M^t_{\overline{R^m}})_{ij} = \min\left\{(M^*_{\overline{R}^m})_{ij}, \left\lfloor \frac{(M^*_{\overline{R}^m})_{i\bar{i}}}{2} \right\rfloor + \left\lfloor \frac{(M^*_{\overline{R}^m})_{\bar{j}j}}{2} \right\rfloor\right\}$$

for all $m \geq 0$ and for all $1 \leq i,j \leq 4N$.

We next prove the asymptotic bound on period of R. If R is $*$-consistent, its period is twice the period of $\overline{R}$, by Lemma 4.20. Thus the period of R is bounded by $c = 2 \cdot \mathrm{lcm}(1, \ldots, 2N)$ and consequently, it is asymptotically bounded by $2^{O(N)}$, by Lemma 4.5. If R is not $*$-consistent, its period is 1 and the same asymptotic bound applies.

Next, we prove the asymptotic bound on the prefix of a $*$-consistent octagonal relation R. Let us define:

$$\{s_m\}_{m\geq 0} = \left\{(M^*_{\overline{R}^k})_{i,j}\right\}_{m\geq 0} \qquad \{t_m\}_{m\geq 0} = \left\{\left\lfloor \tfrac{(M^*_{\overline{R}^k})_{i,\bar{i}}}{2}\right\rfloor + \left\lfloor \tfrac{(M^*_{\overline{R}^k})_{\bar{j},j}}{2}\right\rfloor\right\}_{m\geq 0}$$

By Lemma 4.20, the periodic sequence $\{t_m\}_{m\geq 0}$ has prefix b and period $c' = 2c$. The sequence $\{s_m\}_{m\geq 0}$ has prefix b and period c, but we can without loss of generality assume that its period is $c' = 2c$. By Lemma 4.20, the sequence $\{\min(s_m, t_m)\}_{m\geq 0}$ has period c and prefix defined as $b' = b + \max_{i=0}^{c-1} K_i c'$ where

$$\begin{aligned}
K_i &= \left\lceil \tfrac{s_{b+i} - t_{b+i}}{\lambda_i^{(t)} - \lambda_i^{(s)}} \right\rceil && \text{if } \lambda_i^{(s)} < \lambda_i^{(t)} \text{ and } t_{b+i} < s_{b+i},\\
K_i &= \left\lceil \tfrac{t_{b+i} - s_{b+i}}{\lambda_i^{(s)} - \lambda_i^{(t)}} \right\rceil && \text{if } \lambda_i^{(t)} < \lambda_i^{(s)} \text{ and } s_{b+i} < t_{b+i},\\
K_i &= 0 && \text{otherwise.}
\end{aligned}$$

Observe that

$$\begin{aligned}
s_b &\geq -b \cdot \|\overline{R}\|,\\
t_b &\leq \max\{(M^*_{\overline{R}^b})_{i,\bar{i}}, (M^*_{\overline{R}^b})_{\bar{j},j}\} \leq b \cdot \|\overline{R}\|.
\end{aligned}$$

Thus, if $\lambda_i^{(s)} > \lambda_i^{(t)}$ and $t_{b+i} > s_{b+i}$, then

$$K_i = \left\lceil \tfrac{t_{b+i} - s_{b+i}}{\lambda_i^{(s)} - \lambda_i^{(t)}} \right\rceil \leq t_{b+i} - s_{b+i} \leq 2 \cdot b \cdot \|\overline{R}\|.$$

Similarly, we infer that $K_i \leq 2 \cdot b \cdot \|\overline{R}\|$ if $\lambda_i^{(s)} < \lambda_i^{(t)}$ and $t_{b+i} < s_{b+i}$. Hence, $b' = b + 2 \cdot b \cdot \|\overline{R}\| \cdot c'$ is the prefix of $\{\min(s_m, t_m)\}_{m\geq 0}$ and thus of R. The asymptotic bound $\|R\|^2 \cdot 2^{\mathcal{O}(N)}$ on b' follows.

Finally, we prove the asymptotic bound on the prefix of R that is not $*$-consistent. Let b and c be the prefix and period of $\{M^*_{\overline{R}^m}\}_{m\geq 0}$ as inferred previously. By Theorem 2.20, either $M^*_{\overline{R}^\ell}$ is inconsistent or $\left\lfloor \tfrac{(M^*_{\overline{R}^\ell})_{i,\bar{i}}}{2}\right\rfloor + \left\lfloor \tfrac{(M^*_{\overline{R}^\ell})_{\bar{i},i}}{2}\right\rfloor < 0$ for some $\ell \geq 0$, $1 \leq i \leq 4N$. For the former case, ℓ (and thus the prefix of R) is bounded by $\|R\| \cdot 2^{\mathcal{O}(N)}$, by Corollary 5.3. Now consider the latter case. Let us denote

$$\{s_m\}_{m\geq 0} = \left\{\left\lfloor \tfrac{(M^*_{\overline{R}^m})_{i,\bar{i}}}{2}\right\rfloor + \left\lfloor \tfrac{(M^*_{\overline{R}^m})_{\bar{i},i}}{2}\right\rfloor\right\}_{m\geq 0}$$

and let $\ell \geq 0$ and $1 \leq i \leq 4N$ be such that $s_\ell < 0$. If $\ell \leq b$, we immediately get the required asymptotic bound. If $\ell > b$, then by Lemma 5.29, there exist path schemes in the zigzag automaton $\sigma_1.\lambda_1^*.\sigma_1'$ and $\sigma_2.\lambda_2^*.\sigma_2'$ such that $(M^*_{\overline{R}^\ell})_{i,\bar{i}} = w(\sigma_1.\lambda_1^{b_1}.\sigma_1')$ for some $b_1 \geq 0$ and $(M^*_{\overline{R}^\ell})_{\bar{i},i} = w(\sigma_2.\lambda_2^{b_2}.\sigma_2')$ for some $b_2 \geq 0$ and moreover, letting $c_1 = \frac{c}{|\lambda_1|}$ and $c_2 = \frac{c}{|\lambda_2|}$, the paths $\sigma_1.\lambda_1^{b_1+kc_1}.\sigma_1'$ and $\sigma_2.\lambda_2^{b_2+kc_2}.\sigma_2'$ are minimal for all $k \geq 0$. By

Lemma 4.20, the sequence $\{s_m\}_{m\geq 0}$ has prefix b and period $2c$. Moreover, its rate is $w(\lambda_1^{c_1}) + w(\lambda_2^{c_2})$. Clearly, $w(\lambda_1^{c_1}) + w(\lambda_2^{c_2}) < 0$, since otherwise $s_\ell < 0$ would not be possible. Observe that

$$s_b \leq \max\left\{(M^*_{\overline{R}^b})_{i,\bar{i}}, (M^*_{\overline{R}^b})_{\bar{i},i}\right\} \leq b \cdot \|\overline{R}\|.$$

Then,

$$\ell \leq b \cdot \|\overline{R}\| + \left\lceil \frac{b\cdot\|\overline{R}\|}{-\left(w(\lambda_1^{c_1})+w(\lambda_2^{c_2})\right)} \right\rceil \cdot c \leq b \cdot \|\overline{R}\| \cdot c.$$

Thus, ℓ and consequently the prefix of R are asymptotically bounded by $\|R\|^2 \cdot 2^{O(n)}$. □

5.3 Finite Monoid Affine Relations

An affine relation $R \in \mathbb{Z}^N \times \mathbb{Z}^N$ is defined by a linear arithmetic constraint of the form $\mathbf{x}' = A\mathbf{x} + \mathbf{b}$ where $A \in \mathbb{Z}^{N\times N}$ is a square matrix, and $\mathbf{b} \in \mathbb{Z}^N$ is a column vector. The relation is said to have the finite monoid property if the set $\{A^0, A^1, \ldots\}$ of matrix powers of A is finite. The cardinality of this set of called the *monoid size of* R, and denoted by $[R]$. Finite monoid affine relations are periodic, and the prefix b and period c of a relation R are such that $b + c = [R]$. In this section, we show that the monoid size of a finite monoid relation is bounded by $2^{\mathcal{O}(N^{\log_{10} 11})}$, in other words, it is simply exponential in the number of variables. The developments in this section are closely related to decidability of the finite monoid property as mentioned in Theorem 2.28.

If R has the finite monoid property, then $[R]$ is the smallest integer p from the above theorem. This is because, due to the two conditions of Theorem 2.28, for every $k > 0$, we have that $A^{kp} = A^p$. Moreover, if p is the minimal integer satisfying these conditions, all powers $A^0, A^1, \ldots, A^{p-1}$ are pairwise distinct.

In the rest of this section, we give an upper bound for the smallest integer p that satisfies the conditions of Theorem 2.28. Notice first that every eigenvalue of A^p is of the form λ^p where λ is an eigenvalue of A. Since, by the first condition, λ^p is either zero or one, the only non-zero roots of the characteristic polynomial of A must be roots of the unity. But then $P_A(x)$ is a product of x^k, for some $k < N$, and several cyclotomic polynomials, call them $F_{i_1}, \ldots, F_{i_m}$. Clearly, the degrees of these polynomials are smaller than the degree of P_A, which, in turn, is smaller or equal to N. Let $i_0 = lcm(i_1, \ldots, i_m)$. Then every root λ of P_A has the property $\lambda^{i_0} = 1$, i.e. the first condition from Theorem 2.28 is met for $p = i_0^\ell$, for any integer $\ell > 0$. Moreover, this condition does not hold for any $0 < q < i_0$, or $i_0^\ell < q < i_0^{\ell+1}$, for all $\ell > 0$. But then the second condition of Theorem 2.28, if it holds for some p which is a multiple of i_0, it must hold also for $p = i_0$.

The only remaining question is how big $i_1, \ldots, i_m$ are. The idea is that we do not need to consider cyclotomic polynomials of degree higher than the degree of P_A, which is turn is at most N. A tight bound on the degree of a cyclotomic polynomial is given by the following theorem:

Theorem 5.33 *For every two integers $n > 0$ and $d \geq 0$, such that $n > 210\left(\frac{d}{48}\right)^{\log_{10} 11}$, the degree of $F_n(x)$ is higher than d.*

Proof: See Theorem 8.46 in [Boi99]. □

Since, by Theorem 5.33, we have $0 < i_1, \ldots, i_m < 210\left(\frac{N}{48}\right)^{\log_{10} 11}$, it follows by Lemma 4.5 that $lcm(i_1, \ldots, i_m)$, and implicitly, the minimal integer p satisfying Theorem 2.28, is bounded by $2^{\mathcal{O}(N^{\log_{10} 11})}$. This gives the bound on the size of the monoid for R, which in turn equals the sum $b + c$ between the prefix and the period of R. In conclusion, Algorithm 1 runs in time at most $2^{\mathcal{O}(N^{\log_{10} 11})}$.

6 Computing Termination Pre-conditions of Integer Relations

In this chapter, we address the problem of conditional termination, which is that of defining the *weakest termination set*—the set of initial configurations from which a given program terminates. First, we define its dual, the *weakest non-termination set*—the set of initial configurations from which a non-terminating execution exists—as the greatest fixpoint of the pre-image of the transition relation. This definition enables the representation of this set whenever (1) the closed form of the relation of the loop is definable in a logic that has quantifier elimination and (2) the greatest fixpoint of the pre-image of the relation can be computed as the infimum of the Kleene sequence. We show that these conditions are met for three classes of relations: difference bounds, octagonal, and finite monoid affine relations, and that the weakest non-termination set of relations from these classes is an effectively computable Presburger formula. This entails the decidability of the termination problem for these classes. The Presburger formula defining the weakest non-termination set is defined using the closed form of a relation. This induces a method for computation of the weakest non-termination set which inherits the asymptotic complexity of the algorithm computing the closed form, which we proved to be EXPTIME in Chapter 5.

We study the classes of difference bounds and octagonal relations further and observe that the representation of powers of a relation by zigzag automata provides a better argument to decide termination of these classes. More precisely, we notice that well-foundedness of a difference bounds or an octagonal relation manifests in the existence of a negative cycle in the corresponding zigzag automaton, which makes the limit of the Kleene sequence empty. These observations lead to a PTIME algorithm computing the weakest non-termination sets for difference bounds and octagonal relations that avoids the closed form computations and complex quantifier eliminations. Further, the structure of the negative cycle in a zigzag automaton can be used to prove a result on existence of linear ranking functions. We show that we can construct a witness relation R' for each difference bounds or octagonal relation R such that R and R' have equal sets of infinite runs and, moreover, that R' has a linear ranking function if it is well-founded.

Finally, we study the class of linear affine relations and give a method of under-approximating the termination precondition for a non-trivial subclass of polynomially bounded affine relations.

Related Work. The literature on program termination is vast. Most work focuses however on universal termination, such as the techniques for synthesizing linear ranking functions of Sohn and Van Gelder [SVG91] or Podelski and Rybalchenko [PR04a], and

the more sophisticated method of Bradley, Manna and Sipma [BMS05], which synthesizes lexicographic polynomial ranking functions, suitable when dealing with disjunctive loops. However, not every terminating program (loop) has a linear (polynomial) ranking function. In this chapter, we show that for an entire class of non-deterministic linear relations, defined using octagons, termination is always witnessed by a computable octagonal relation that has a linear ranking function.

Another line of work considers the decidability of termination for simple (conjunctive) linear loops. Initially, Tiwari [Tiw04] showed decidability of termination for affine linear loops interpreted over *reals*, while Braverman [Bra06] refined this result by showing decidability over *rationals* and over *integers*, for homogeneous relations of the form $C_1\mathbf{x} > 0 \ \wedge \ C_2\mathbf{x} \geq 0 \ \wedge \ \mathbf{x}' = A\mathbf{x}$. The non-homogeneous integer case seems to be much more difficult as it is closely related to the open *Skolem's Problem* [HHHK05]: given a linear recurrence $\{u_i\}_{i\geq 0}$, determine whether $u_i = 0$ for some $i \geq 0$.

To our knowledge, the first work on proving non-termination of simple loops is reported in [GHM$^+$08]. The notion of *recurrent sets* occurs in this work, however, without the connection with fixpoint theory, which is introduced in the present work. Finding recurrent sets in [GHM$^+$08] is complete with respect to a predefined set of templates, typically linear systems of rational inequalities.

The work which is closest to ours is probably that of Cook et al. [CGLA$^+$08]. In that paper, the authors develop an algorithm for deriving termination preconditions by first guessing a ranking function candidate (typically the linear term from the loop condition) and then inferring a supporting assertion which guarantees that the candidate function decreases with each iteration. The step of finding a supporting assertion requires a fixpoint iteration in order to find an invariant condition. Unlike our work, the authors of [CGLA$^+$08] do not address issues related to completeness: the method is not guaranteed to find the weakest precondition for termination, even in cases when this set can be computed. On the other hand, it is applicable to a large range of programs extracted from real-life software. To compare our method with theirs, we tried the examples available in [CGLA$^+$08]. For those which are polynomially bounded affine relations, we used our under-approximation method and have computed termination preconditions, which turn out to be slightly more general than the ones reported in [CGLA$^+$08].

Roadmap. Section 6.1 gives the fixpoint characterization of the weakest non-termination set and states conditions under which this set can be computed as the limit of the Kleene sequence. Sections 6.2 and 6.3 prove that these conditions are met for octagonal relations (and implicitly for difference bounds relations) and for finite monoid affine relations, respectively. Section 6.2.1 presents a PTIME algorithm computing the weakest non-termination set for octagonal relations (and implicitly for difference bounds relations). Section 6.3.1 presents a method that computes termination preconditions for polynomially bounded affine relations. We defer all experiments with our methods for termination analysis to Chapter 8.

6.1 Preconditions for Non-termination

We first recall the notions of $*$-*consistent* and *well-founded* relations.

Definition 6.1 *A relation $R \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$ defined by $R(\mathbf{x}, \mathbf{x}')$ is said to be* $*$-consistent *if and only if, for any $m > 0$, there exists a sequence of valuations $\{\nu_i \in \mathbb{Z}^{\mathbf{x}}\}_{i=0}^{m}$, such that $\nu_i, \nu_{i+1} \models R(\mathbf{x}, \mathbf{x}')$, for all $i = 0, \ldots, m-1$. $R(\mathbf{x}, \mathbf{x}')$ is said to be* well-founded *if and only if there is no infinite sequence of valuations $\{\nu_i : \mathbb{Z}^{\mathbf{x}}\}_{i \geq 0}$, such that $\nu_i, \nu_{i+1} \models R(\mathbf{x}, \mathbf{x}')$, for all $i \geq 0$.*

Notice that if a relation is not $*$-consistent, then it is also well-founded. However the dual is not true. For instance, the relation $R = \{(n, n-1) \mid n > 0\}$ is both $*$-consistent and well-founded. Also notice that a relation R is $*$-consistent if and only if R^i is consistent for all $i \geq 0$.

Definition 6.2 *A set $S \subseteq \mathbb{Z}^N$ is said to be a* non-termination precondition *for $R(\mathbf{x}, \mathbf{x}')$ if, for each $s \in S$ there exists an infinite sequence of valuations $\{\nu_i : \mathbb{Z}^{\mathbf{x}}\}_{i \geq 0}$ such that $s = \nu_0(\mathbf{x})$ and $\nu_i, \nu_{i+1} \models R(\mathbf{x}, \mathbf{x}')$, for all $i \geq 0$.*

If $S_0, S_1, \ldots$ are all non-termination preconditions for R, then the (possibly infinite) union $\bigcup_{i=0,1,\ldots} S_i$ is a non-termination precondition for R as well. The set $\mathrm{wnt}(R) = \bigcup\{S \in \mathbb{Z}^N \mid S \text{ is a non-termination precondition for } R\}$ is called the *weakest non-termination precondition* for R. A relation R is well-founded if and only if $\mathrm{wnt}(R) = \emptyset$. A set S such that $S \cap \mathrm{wnt}(R) = \emptyset$ is called a *termination precondition*.

Definition 6.3 *A set $S \subseteq \mathbb{Z}^N$ is said to be* recurrent *for a relation $R \in \mathbb{Z}^N \times \mathbb{Z}^N$ if and only if $S \subseteq \mathrm{pre}_R(S)$.*

Notice that if S is a recurrent set for a relation R, then for each $s \in S$ there exists $s' \in S$ such that $(s, s') \in R$.

Proposition 6.4 *Let $S_0, S_1, \ldots \in \mathbb{Z}^N$ be a (possibly infinite) sequence of sets, all of which are recurrent for a relation $R \in \mathbb{Z}^N \times \mathbb{Z}^N$. Then their union $\bigcup_{i=0,1,\ldots} S_i$ is recurrent for R as well.*

Proof: For each i we have $S_i \subseteq \mathrm{pre}_R(S_i) \subseteq \mathrm{pre}_R(\bigcup_{j=0,1,\ldots} S_j)$. The last inclusion is by the monotonicity of pre_R. Hence $\bigcup_{j=0,1,\ldots} S_j \subseteq \mathrm{pre}_R(\bigcup_{j=0,1,\ldots} S_j)$. □

The set $\mathrm{wrs}(R) = \bigcup\{S \in \mathbb{Z}^N \mid S \text{ is a recurrent set for } R\}$ is called the *weakest recurrent set* for R. By Proposition 6.4, $\mathrm{wrs}(R)$ is recurrent for R. The following lemma shows that in fact, this is exactly the set of valuations from which an infinite iteration is also possible.

Lemma 6.5 *Given a relation $R \in \mathbb{Z}^N \times \mathbb{Z}^N$, the weakest recurrent set for R equals its weakest non-termination precondition.*

Proof: "$\mathrm{wrs}(R) \subseteq \mathrm{wnt}(R)$" Let $s_0 \in \mathrm{wrs}(R)$ be a valuation. Then there exists $s_1 \in \mathrm{wrs}(R)$ such that $(s_0, s_1) \in R$. Applying this argument infinitely many times, one can construct an infinite sequence $s_0, s_1, s_2, \ldots$ such that $(s_i, s_{i+1}) \in R$, for all $i \geq 0$. Hence $s_0 \in \mathrm{wnt}(R)$.

"$\mathrm{wnt}(R) \subseteq \mathrm{wrs}(R)$" Let $s_0 \in \mathrm{wnt}(R)$ be a valuation and let $s_0, s_1, s_2, \ldots$ be arbitrary infinite sequence such that $(s_i, s_{i+1}) \in R$, for all $i \geq 0$. Clearly, $s_1 \in \mathrm{wnt}(R)$ too. Consequently, $s_0 \in \mathrm{pre}_R(\mathrm{wnt}(R))$ for each state $s_0 \in \mathrm{wnt}(R)$ and hence, $\mathrm{wnt}(R) \subseteq \mathrm{pre}_R(\mathrm{wnt}(R))$. Thus, $\mathrm{wnt}(R)$ is a recurrent set and hence $\mathrm{wnt}(R) \subseteq \mathrm{wrs}(R)$. □

Next we define the weakest recurrent set as the greatest fixpoint of the transition relation's pre-image.

Lemma 6.6 *Given a relation $R \in \mathbb{Z}^N \times \mathbb{Z}^N$, the weakest recurrent set for R is the greatest fixpoint of the function $X \mapsto \mathrm{pre}_R(X)$.*

Proof: By the Knaster-Tarski Fixpoint Theorem, $\mathrm{gfp}(\mathrm{pre}_R) = \bigcup\{S \mid S \subseteq \mathrm{pre}_R(S)\} = \mathrm{wrs}(R)$. □

The following two lemmas give conditions under which $\mathrm{wrs}(R)$ can be computed as the infimum of the Kleene sequence $\{\mathrm{pre}_R^m(\mathbb{Z}^N)\}_{m\geq 0}$.

Lemma 6.7 *Let $R(\mathbf{x}, \mathbf{x}')$ be a relation and $n_2 > n_1 \geq 0$ such that $\mathrm{pre}_R^{n_1}(\mathbb{Z}^N) = \mathrm{pre}_R^{n_2}(\mathbb{Z}^N)$. Then, $\mathrm{wrs}(R) = \bigcap_{m\geq 0} \mathrm{pre}_R^m(\mathbb{Z}^N) = \mathrm{pre}_R^{n_1}(\mathbb{Z}^N) = \mathrm{pre}_R^{n_2}(\mathbb{Z}^N)$.*

Proof: First observe that $\mathrm{wrs}(R) = \mathrm{wnt}(R) \subseteq \bigcap_{m\geq 0} \mathrm{pre}_R^m(\mathbb{Z}^N)$, by Lemma 6.5 and by the fact that $\mathrm{wnt}(R) \subseteq \mathrm{pre}_R^m(\mathbb{Z}^N)$ for each $m \geq 0$. Second, observe that $\mathrm{pre}_R^n(\mathbb{Z}^N) = \mathrm{pre}_R^{n_1}(\mathbb{Z}^N)$ for all $n \geq n_1$, by the hypothesis of the lemma and by monotonicity of pre_R. Hence, $\bigcap_{m\geq 0} \mathrm{pre}_R^m(\mathbb{Z}^N) = \bigcap_{0\leq m\leq n_1} \mathrm{pre}_R^m(\mathbb{Z}^N) = \mathrm{pre}_R^{n_1}(\mathbb{Z}^N) = \mathrm{pre}_R^{n_2}(\mathbb{Z}^N)$ is the greatest fixpoint of pre_R and it is thus equal to $\mathrm{wrs}(R)$, by Lemma 6.6. □

Lemma 6.8 *Let $R \in \mathbb{Z}^N \times \mathbb{Z}^N$ be a relation such that either*

1. *pre_R is $\cap$-continuous, or*
2. *$\mathrm{pre}_R^{n_2}(\mathbb{Z}^N) = \mathrm{pre}_R^{n_1}(\mathbb{Z}^N)$ for some $n_2 > n_1 \geq 0$, or*
3. *$\bigcap_{m\geq 0} \mathrm{pre}_R^m(\mathbb{Z}^N) = \emptyset$.*

Then, $\mathrm{wrs}(R) = \bigcap_{m\geq 0} \mathrm{pre}_R^m(\mathbb{Z}^N)$.

Proof: If the condition 1 holds, one can apply the Kleene Fixpoint Theorem and conclude that $\mathrm{wrs}(R) = \mathrm{gfp}(\mathrm{pre}_R) = \bigcap_{m\geq 0} \mathrm{pre}_R^m(\mathbb{Z}^N)$. If the condition 2 holds, one can apply Lemma 6.7 and conclude that the lemma holds. For the condition 3 holds, observe that $\mathrm{wnt}(R) \subseteq \mathrm{pre}_R^m(\mathbb{Z}^N)$ for each $m \geq 0$. Consequently,

$$\mathrm{wrs}(R) = \mathrm{wnt}(R) \subseteq \bigcap_{m\geq 0} \mathrm{pre}_R^m(\mathbb{Z}^N) = \emptyset$$

Hence, $\mathrm{wrs}(R) = \bigcap_{m\geq 0} \mathrm{pre}_R^m(\mathbb{Z}^N) = \emptyset$ and the lemma holds. □

The Lemma 6.8 suggests a method for computing weakest recurrent sets of relations that satisfy one of the conditions stated in the lemma. In subsequent sections, we show that Lemma 6.8 is applicable for both octagonal and finite-monoid affine relations. Thus, $\mathrm{wrs}(R) = \mathrm{gfp}(\mathrm{pre}_R) = \bigcap_{m>0} \mathrm{pre}_R^m(\mathbb{Z}^N) = \bigcap_{m>0} \mathrm{pre}_{R^m}(\mathbb{Z}^N)$ for these classes. In Chapter 3, we showed that the closed form $\widehat{R}(\mathbf{x}, \mathbf{x}', k)$ is definable for these classes. Using the closed form $\widehat{R}(\mathbf{x}, \mathbf{x}', k)$ of R, one can now define $\mathrm{wrs}(R)$ as a Presburger formula:

$$\mathrm{wrs}(R) \equiv \forall k \geq 0 \; \exists \mathbf{x}' \; . \; \widehat{R}(k, \mathbf{x}, \mathbf{x}') \tag{6.1}$$

Because Presburger arithmetic has quantifier elimination, $\mathrm{wrs}(R)$ can be defined in Presburger arithmetic too. Decidability of Presburger arithmetic entails decidability of the termination problem for octagonal and finite-monoid affine relations.

Example. Consider an octagonal relation $R(x, x') \equiv x \geq 0 \wedge x' = x - 1$. The closed form of $R(x, x')$ is $\widehat{R}(\mathbf{x}, \mathbf{x}', k) \equiv x \geq k - 1 \wedge x' = x - k$. Quantifier elimination yields $\mathrm{wrs}(R) \equiv \forall k > 0 \; \exists x' \; . \; x \geq k - 1 \wedge x' = x - k \equiv \forall k \geq 0 \; . \; x \geq k - 1 \equiv \mathrm{false}$. Hence the relation R is well-founded. □

6.2 Octagonal Relations

For a set $\mathbf{v}$ of variables, let $U(\mathbf{v}) = \{\pm v_1 \pm v_2 \mid v_1, v_2 \in \mathbf{v}\}$ denote the set of octagonal terms over $\mathbf{v}$. As a first remark, by the periodicity of the sequence $\{M_{R^i}^t\}_{i \geq 0}$, proved in Chapter 4.3, the closed form of the subsequence $\{R^{b+c\ell}\}_{\ell \geq 0}$ (of $\{R^i\}_{i \geq 0}$) can be defined as:

$$\widehat{R}_{b,c}(\ell, \mathbf{x}, \mathbf{x}') \equiv \bigwedge_{u \in U(\mathbf{x} \cup \mathbf{x}')} u \leq a_u \ell + d_u \tag{6.2}$$

where $a_u = (\Lambda_0)_{ij}$, $d_u = (M_{R^b}^t)_{ij}$ for all octagonal terms $u = y_i - y_j$. This is indeed the case, since the matrix sequence $\{M_{R^{b+c\ell}}^t\}_{\ell \geq 0}$ is periodic i.e., $M_{R^{b+c\ell}}^t = M_{R^b}^t + \ell \Lambda_0$, for all $\ell \geq 0$. Moreover, it follows that the parametric DBM encoding of $\widehat{R}_{b,c}(\ell, \mathbf{x}, \mathbf{x}')$ is tightly closed.

Lemma 6.9 *Let R be a $*$-consistent octagonal relation with prefix b, period c and let $\widehat{R}_{b,c}(\ell, \mathbf{x}, \mathbf{x}')$ be the closed form of $\{R^{b+c\ell}\}_{\ell \geq 0}$ as defined in (6.2). Then,* $\mathrm{wrs}(R) = \bigcap_{k \geq 0} \mathrm{pre}_R^k(\mathbb{Z}^N)$. *Moreover,* $\mathrm{wrs}(R) = \emptyset$ *if there exists $u \in U(\mathbf{x})$ s.t. $a_u < 0$. Otherwise,* $\mathrm{wrs}(R) = R^{-b}(\mathbb{Z}^N)$.

Proof: Notice that the function pre_R is monotonic and thus, $\mathrm{pre}_R^{k_1}(\mathbb{Z}^n) \supseteq \mathrm{pre}_R^{k_2}(\mathbb{Z}^n)$, for $k_1 \leq k_2$. Consequently, we have that $\bigcap_{k>0} \mathrm{pre}_R^k(\mathbb{Z}^N) = \bigcap_{\ell \geq 0} \mathrm{pre}_R^{b+c\ell}(\mathbb{Z}^N)$. The latter set can now be defined using the closed form of the subsequence (6.2) i.e.,

$$\bigcap_{k>0} \mathrm{pre}_R^k(\mathbb{Z}^N) \equiv \forall \ell \geq 0 \; \exists \mathbf{x}' \; . \; \widehat{R}_{b,c}(\ell, \mathbf{x}, \mathbf{x}')$$

Since the parametric DBM encoding of $\widehat{R}_{b,c}(\ell, \mathbf{x}, \mathbf{x}')$ is tightly closed, if follows that the existential quantifier $\exists \mathbf{x}'$ can be eliminated by simply deleting all atomic propositions involving primed variables from (6.2). Thus, we obtain:

$$\begin{array}{rcl} \bigcap_{k>0} \mathrm{pre}_R^k(\mathbb{Z}^N) & \equiv & \forall \ell \geq 0 \ \bigwedge_{u \in U(\mathbf{x})} u \leq a_u \ell + d_u \\ & \equiv & \bigwedge_{u \in U(\mathbf{x})} u \leq \inf \{a_u \ell + d_u \mid \ell \geq 0\} \end{array}$$

where, for a set $S \subseteq \mathbb{Z}$, $\inf S$ denotes the minimal element of S, if one exists, or $-\infty$, otherwise. We have

$$\inf \{a_u \ell + d_u \mid \ell \geq 0\} = \begin{cases} -\infty & \text{if } a_u < 0, \\ d_u & \text{otherwise.} \end{cases}$$

Hence $\bigcap_{k>0} \mathrm{pre}_R^k(\mathbb{Z}^N)$ is the empty set, if $a_u < 0$ for some $u \in U(\mathbf{x})$. In this case, condition 3 of Lemma 6.8 holds. Otherwise, we obtain $\bigcap_{k>0} \mathrm{pre}_R^k(\mathbb{Z}^N) \equiv \bigwedge_{u \in U(\mathbf{x})} u \leq d_u$. However, this is exactly the set $\mathrm{pre}_R^b(\mathbb{Z}^N)$, by (6.2). In this case, condition 2 of Lemma 6.8 holds. Thus, we can apply Lemma 6.8 in both cases and conclude that $\mathrm{wrs}(R) = \bigcap_{k>0} \mathrm{pre}_R^k(\mathbb{Z}^N)$. To summarize, $\mathrm{wrs}(R) = \emptyset$ if $a_u < 0$ for some $u \in U(\mathbf{x})$. Otherwise, $\mathrm{wrs}(R) = R^{-b}(\mathbb{Z}^N)$. □

An immediate consequence is that the termination problem is decidable and that the weakest termination set is an effectively computable Presburger formula.

Theorem 6.10 *The termination problem is decidable for octagonal relations. Moreover, the weakest non-termination set of an octagonal relation is an effectively computable octagonal constraint.*

Proof: By Lemma 6.9, the weakest non-termination set of an octagonal relation is either empty or $R^{-b}(\mathbb{Z}^N)$. Moreover, Lemma 6.9 gives means to compute this set. Thus, the termination problem can be decided by checking whether $\mathrm{wrs}(R) = \emptyset$. □

Lemma 6.9 can be used to compute $\mathrm{wrs}(R)$ for an octagonal relation R. First we need to check the $*$-consistency of R, using the method reported in Chapter 4.3. Second, we compute the closed form (6.2) and check for the existence of a term $u \in U(\mathbf{x})$ such that $a_u < 0$, in which case $wrs(R) = \emptyset$ and R is well-founded. Finally, if this is not the case, then we compute $\mathrm{wrs}(R) = R^{-b}(\mathbb{Z}^N)$. On the complexity side, the computation of the closed form has asymptotic time complexity of $\|R\|^2 \cdot 2^{O(N)}$, as proved in Chapter 5. The rest of the computation is linear in the size of the closed form. Thus, the method runs in $\|R\|^2 \cdot 2^{O(N)}$ asymptotic time.

6.2.1 Computing WNT in Polynomial Time

Lemma 6.9 can be used to build an EXPTIME algorithm for computation of the weakest non-termination precondition, as discussed in the previous section. In this section, we give a different termination argument which leads to an efficient polynomial time algorithm that computes the weakest non-termination precondition.

Lemma 6.9 proves that the weakest recurrent set of an octagon is the limit of the sequence $\{R^{-k}(\mathbb{Z}^N)\}_{k\geq 0}$. Note that $R^{-k}(\mathbb{Z}^N)$ is represented as $(M^t_{\overline{R}^k})_{\downarrow \mathbf{y}}$, a projection of $M^t_{\overline{R}^k}$ onto unprimed variables. By Theorem 2.26, $(M^t_{\overline{R}^k})_{\downarrow \mathbf{y}}$ is computed by tightening $(M^*_{\overline{R}^k})_{\downarrow \mathbf{y}}$. Recall that entries of $(M^*_{\overline{R}^k})_{\downarrow \mathbf{y}}$ can be encoded as weights of minimal runs of length $k+2$ in the even forward zigzag automaton $\mathcal{A}^{ef}_{\overline{R}}$ defined in Section 2.3.3. The later developments of this section depend on this encoding and we thus recall it in the following example.

Example. Consider the set of variables $\mathbf{x} = \{x_1, \ldots, \mathbf{x}_4\}$ and a difference bounds relation $R(\mathbf{x}, \mathbf{x}') \equiv x_2 - x'_1 \leq -1 \wedge x_3 - x'_2 \leq 0 \wedge x_1 - x'_3 \leq 0 \wedge x'_4 - x_4 \leq 0 \wedge x'_3 - x_4 \leq 0$. Graph representation $\mathcal{G}_R$ of the relation $R(\mathbf{x}, \mathbf{x}')$ is depicted in Figure 6.1 (a). Figure 6.1 (b) shows $\mathcal{G}_{R^8}$, the 8-times unfolding of $\mathcal{G}_R$. The transition table that is common to all even forward zigzag automata is given in Figure 6.1 (c). An example of a run of $\mathcal{A}^{ef}_R$ recognizing a path of constraints in $\mathcal{G}^8_R$ is given in Figure 6.1 (d). The word accepted by π is a subgraph of $\mathcal{G}^8_R$ shown in Figure 6.1 (b). The cycle $\lambda : q_1 \xrightarrow{G_1} q_2 \xrightarrow{G_2} q_3 \xrightarrow{G_3} q_1$ is taken twice in this run. The weights of the symbols on the run are $w(G_1) = w(G_2) = w(G_4) = 0$ and $w(G_3) = -1$. □

The following proposition relates values of entries in $(M^*_{\overline{R}^k})_{\downarrow \mathbf{y}}$ to weights of runs of length $k+2$ in even forward zigzag automata.

Proposition 6.11 *Let $R(\mathbf{x}', \mathbf{x}')$, $\mathbf{x} = \{x_1, \ldots, x_N\}$, be a $*$-consistent octagonal relation, $\overline{R}(\mathbf{y}, \mathbf{y}')$, $\mathbf{y} = \{y_1, \ldots, y_{2N}\}$, be a difference bounds representation of $R(\mathbf{x}', \mathbf{x}')$, and let $\mathcal{A}^{ef}_{\overline{R}}$ be the even forward zigzag automaton corresponding to $\overline{R}(\mathbf{y}, \mathbf{y}')$. Then, the following assertions are equivalent for all $1 \leq i, j \leq 2N$ and all $m \geq 0$,*

1. *there exists an acyclic path ρ from $y^{(0)}_i$ to $y^{(0)}_j$ in $\mathcal{G}^m_{\overline{R}}$*
2. *there exists a run π in $\mathcal{A}^{ef}_{\overline{R}}$ of length $m+2$ such that $w(\pi) = w(\rho)$ and π is of the form*

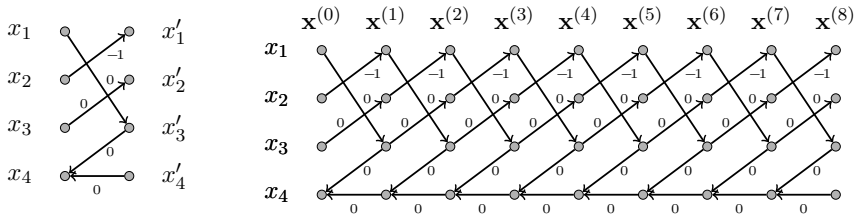
$$\pi = \underbrace{(I^{ef}_{i,j} \to q)}_{\sigma_1} . \underbrace{(q \to \ldots \to q')}_{\sigma_2} . \underbrace{(q' \to F^{ef})}_{\sigma_3} . \underbrace{(F^{ef} \to F^{ef})^n}_{\sigma_4} .$$

where $q, q' \in \{l, r, lr, rl, \perp\}^{2N}$ are control states, $n \geq 0$, $w(\pi) = w(\sigma_2)$, and $w(\sigma_1) = w(\sigma_3) = w(\sigma_4) = 0$.

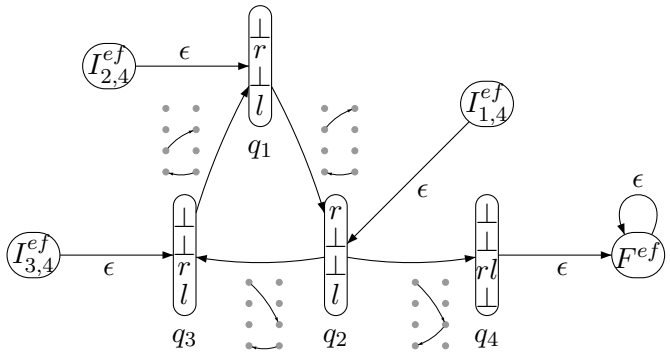
and moreover, $(M^*_{\overline{R}^m})_{i,j} \leq w(\pi)$ for each path π of the above form. Moreover, there exists a path π of the above form such that $w(\pi) = (M^*_{\overline{R}^m})_{i,j}$.

Proof: Follows from construction of $\mathcal{A}^{ef}_{\overline{R}}$ and the fact that $(M^*_{\overline{R}^m})_{i,j}$ is the weight of the minimal weight path from $y^{(0)}_i$ to $y^{(0)}_j$ in $\mathcal{G}^m_{\overline{R}}$. □

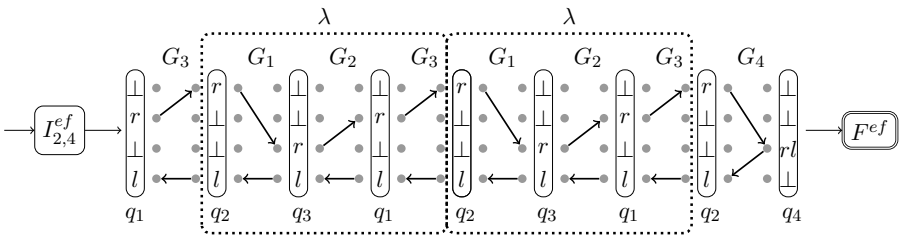
The following lemma gives several equivalent conditions for checking that an octagonal relation is well-founded. They will later be used to design an efficient polynomial



(a) $\mathcal{G}_R$ (b) $\mathcal{G}_R^8$



(c) A^{ef} – transition table for forward even zigzag automata



(d) a run in zigzag automaton over a path in $\mathcal{G}_R^8$.

Figure 6.1: (a) $\mathcal{G}_R$ – graph representation of $R(\mathbf{x}, \mathbf{x}')$. (b) $\mathcal{G}_R^8$ – 8-times unfolding of $\mathcal{G}_R$. (c) Common transition table of even forward zigzag automata. (d) a run of the zigzag automaton over a path in $\mathcal{G}_R^8$. Indices of the initial control state $I^{ef}_{2,4}$ indicate that the run encodes a path from $x_2^{(0)}$ to $x_4^{(0)}$.

time algorithm that computes the weakest recurrent set of an octagon. The conditions also provide basis for the proof of existence of linear ranking functions for well-founded octagonal relations.

Lemma 6.12 *Let $R(\mathbf{x}, \mathbf{x}')$, $\mathbf{x} = \{x_1, \ldots, x_N\}$, be a $*$-consistent octagonal relation with prefix b and $\overline{R}(\mathbf{y}, \mathbf{y}')$, $\mathbf{y} = \{y_1, \ldots, y_{2N}\}$, be the difference bounds encoding of $R(\mathbf{x}, \mathbf{x}')$. Then, the following statements are equivalent.*

1. *R is well-founded*
2. *$R^{-n_1}(\mathbb{Z}^N) \neq R^{-n_2}(\mathbb{Z}^N)$ for some $n_2 > n_1 \geq b$*

3. $R^{-n_1}(\mathbb{Z}^N) \neq R^{-n_2}(\mathbb{Z}^N)$ *for some* $n_2 > n_1 \geq 5^{2N}$

4. *there exists a path* $\sigma.\lambda.\sigma'$ *in* $\mathcal{A}^{ef}_{\overline{R}}$ *of the form* $\sigma : I^{ef}_{i,j} \rightsquigarrow q$, $\lambda : q \rightsquigarrow q$, $\sigma' : q \rightsquigarrow F^{ef}$ *for some* $1 \leq i, j \leq 2N$ *and a control state* q *such that* $w(\lambda) < 0$.

5. $\overline{R}$ *is well-founded*

Proof: $(1 \Rightarrow 2)$ For a proof by contraposition, suppose that $R^{-n_1}(\mathbb{Z}^N) = R^{-n_2}(\mathbb{Z}^N)$ for all $n_2 > n_1 \geq 0$. Then, $\mathrm{wrs}(R) = R^{-b}(\mathbb{Z}^N) \neq \emptyset$. The equality follows from Lemma 6.7, the inequality is due to $*$-consistency of R. Thus, $\mathrm{wrs}(R) \neq \emptyset$ and R is not well-founded, contradiction.

$(1 \Rightarrow 3)$ Similar to $(1 \Rightarrow 2)$.

$(2 \Rightarrow 1)$ For a proof by contraposition, suppose that R is not well-founded. By Lemma 6.9, $\mathrm{wrs}(R) = R^{-b}(\mathbb{Z}^N)$. Since $\mathrm{wrs}(R)$ is a fixpoint, then clearly $\mathrm{wrs}(R) = R^{-b}(\mathbb{Z}^N) = R^{-n}(\mathbb{Z}^N)$ for all $n \geq b$. Consequently, $R^{-n_1}(\mathbb{Z}^N) = \mathrm{wrs}(R) = R^{-n_2}(\mathbb{Z}^N)$ for all $n_2 > n_1 \geq b$.

$(3 \Rightarrow 4)$ Let $n_2 > n_1 \geq 5^{2N}$ such that $R^{-n_1}(\mathbb{Z}^N) \neq R^{-n_2}(\mathbb{Z}^N)$. Then, $\overline{R}^{-n_1}(\mathbb{Z}^{2N}) \neq \overline{R}^{-n_2}(\mathbb{Z}^{2N})$ too by contraposition: For all $m \geq 0$, the tightly closed difference bounds encoding of $R^{-m}(\mathbb{Z}^N)$ is $(M^t_{\overline{R}^m})_{\downarrow \mathbf{y}}$, a restriction of $M^t_{\overline{R}^m}$ to the entries corresponding to unprimed variables. If $\overline{R}^{-n_1}(\mathbb{Z}^{2N}) = \overline{R}^{-n_2}(\mathbb{Z}^{2N})$, then $(M^*_{\overline{R}^{n_1}})_{\downarrow \mathbf{y}} = (M^*_{\overline{R}^{n_2}})_{\downarrow \mathbf{y}}$, by Proposition 2.8. This implies that $(M^t_{\overline{R}^{n_1}})_{\downarrow \mathbf{y}} = (M^t_{\overline{R}^{n_2}})_{\downarrow \mathbf{y}}$, by Theorem 2.26. Consequently, $R^{-n_1}(\mathbb{Z}^N) = R^{-n_2}(\mathbb{Z}^N)$ by Proposition 2.21.

Since $\overline{R}^{-n_1} \neq \overline{R}^{-n_2}$, then $(M^*_{\overline{R}^{n_1}})_{i,j} > (M^*_{\overline{R}^{n_2}})_{i,j}$ for some $1 \leq i \neq j \leq 2N$. By Proposition 6.11, there exists a path π in $Z_{\overline{R}}$ from $I^{ef}_{i,j}$ to F^{ef} such that $w(\pi) = (M^*_{\overline{R}^{n_1}})_{i,j}$. Moreover π has length $n_1 + 2$ and can be written as $\pi = \sigma_1.\sigma_2.\sigma_3.\sigma_4$ where $w(\pi) = w(\pi_2)$ and $w(\sigma_1) = w(\sigma_3) = w(\sigma_4) = 0$. Similarly, where is a path π' from $I^{ef}_{i,j}$ to F^{ef} of length $n_2 + 2$ such that $w(\pi') = (M^*_{\overline{R}^{n_2}})_{i,j}$. $\pi' = \sigma'_1.\sigma'_2.\sigma'_3.\sigma'_4$ where $w(\pi') = w(\pi'_2)$ and $w(\sigma'_1) = w(\sigma'_3) = w(\sigma'_4) = 0$.

We prove that $|\sigma'_2| > n_1$ by contradiction. Suppose that $|\sigma'_2| \leq n_1$ and denote $n = |\sigma'_2|$. The path $\theta = \sigma'_1.\sigma'_2.\sigma'_3$ has length $n + 2$ and thus $(M^*_{\overline{R}^n})_{i,j} \leq w(\theta)$, by Proposition 6.11. We obtain that $(M^*_{\overline{R}^n})_{i,j} \leq w(\pi') = (M^*_{\overline{R}^{n_2}})_{i,j}$, by the fact that $w(\theta) = w(\sigma'_2) = w(\pi')$ and by Proposition 6.11. Since $n \leq n_1 < n_2$, we infer that $(M^*_{\overline{R}^n})_{i,j} = (M^*_{\overline{R}^{n_1}})_{i,j} = (M^*_{\overline{R}^{n_2}})_{i,j}$, by monotonicity of pre_R. Contradiction with $(M^*_{\overline{R}^{n_1}})_{i,j} > (M^*_{\overline{R}^{n_2}})_{i,j}$.

Since $|\sigma'_2| > n_1 \geq 5^{2N}$, there are cycles (at least one) in σ'_2. Observe that:

- None of these cycles can be positive, since positive weight would imply that π' is not a minimal run of length n_2, which contradicts the assumption.

- Not all these cycles are zero-weight, which can be demonstrated by contradiction. Suppose all cycles are zero-weight, erase all of them from σ'_2 and denote the resulting path ρ_2 and its length $n = |\rho_2| \leq 5^{2N}$. Next, build $\theta = \sigma'_1.\rho_2.\sigma'_3$. We infer that $(M^*_{\overline{R}^n})_{i,j} \leq w(\pi') = (M^*_{\overline{R}^{n_2}})_{i,j}$, since $w(\theta) = w(\rho_2) = w(\sigma'_2) = w(\pi')$ and by Proposition 6.11. Since $n \leq n_1 < n_2$, we infer that $(M^*_{\overline{R}^n})_{i,j} = (M^*_{\overline{R}^{n_1}})_{i,j} = (M^*_{\overline{R}^{n_2}})_{i,j}$, by monotonicity of pre_R. Contradiction with $(M^*_{\overline{R}^{n_1}})_{i,j} > (M^*_{\overline{R}^{n_2}})_{i,j}$.

The above proves that there exists at least one negative-weight cycle in σ'_2. Consequently, π' can be split into $\pi' = \sigma.\lambda.\sigma'$ where λ is a negative weight cycle.

$(4 \Rightarrow 5)$ By Proposition 6.11, the length of $\sigma.\sigma'$ is $|\sigma.\sigma'| = m + 2$, $m \geq 0$ and $\sigma.\sigma'$ starts in $I^{ef}_{i,j}$ and ends in F^{ef} for some $1 \leq i, j \leq 2N$. Let $p = |\lambda|$. Clearly, $(M^*_{\overline{R}^{m+kp}})_{i,j} \leq w(\sigma.\lambda^k.\sigma')$ for all $k \geq 0$, by Proposition 6.11. The infinite sequence $\{w(\sigma.\lambda^k.\sigma')\}_{k\geq 0}$ is strictly decreasing and thus $inf\{w(\sigma.\lambda^k.\sigma')\}_{k\geq 0} = -\infty$. Consequently $inf\{(M^*_{\overline{R}^{m+kp}})_{i,j}\}_{k\geq 0} = -\infty$ too. By monotonicity of pre,

$$inf\{(M^*_{\overline{R}^m})_{i,j}\}_{m\geq 0} = inf\{(M^*_{\overline{R}^{m+kp}})_{i,j}\}_{k\geq 0}.$$

Thus, $inf\{(M^*_{\overline{R}^m})_{i,j}\}_{m\geq 0} = -\infty$. Consequently, $\text{wrs}(\overline{R}) = \bigcap_{m\geq 0} \overline{R}^{-m}(\mathbb{Z}^N) = \emptyset$ and $\overline{R}$ is well-founded.

$(5 \Rightarrow 1)$ If $\overline{R}$ is well-founded, then $\bigcap_{m\geq 0} \overline{R}^{-m}(\mathbb{Z}^N) = \emptyset$. Thus, there exist $1 \leq i, j \leq N$ such that $inf\{(M^*_{\overline{R}^m})_{i,j}\}_{m\geq 0} = -\infty$. Since $(M^t_{\overline{R}^m})_{i,j} \leq (M^*_{\overline{R}^m})_{i,j}$ for all $m \geq 0$, by Theorem 2.26, we infer that $inf\{(M^t_{\overline{R}^m})_{i,j}\}_{m\geq 0} = -\infty$ too. Hence, $\bigcap_{m\geq 0} R^{-m}(\mathbb{Z}^N) = \emptyset$ and R is well-founded. □

The computation of the weakest non-termination set based on Lemma 6.9 requires computation of the closed form $\widehat{R}(k, \mathbf{x}, \mathbf{x}')$ which has $\|R\|^2 \cdot 2^{O(n)}$ worst case time complexity, as proved in Chapter 5. As we next show, it turns out that Lemma 6.12 provides a better argument for computation of the weakest recurrent set that avoids computation of the closed form. Algorithm 4 computes the weakest recurrent set for arbitrary octagonal relation in $PTIME$ as proved by Theorem 6.13.

Algorithm 4 Weakest Non-termination Set for Octagons

input An octagonal relation $R \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$
output The weakest non-termination set of R

```
1: function WNT(R)
2:     V ← R
3:     for all i ∈ 1, 2, ... 5N do
4:         V ← V ∘ V
5:     W ← V ∘ R
6:     if V^{-1}(Z^N) = W^{-1}(Z^N) then return V^{-1}(Z^N)
7:     else return ∅
```

Theorem 6.13 *Given an octagonal relation $R \subseteq \mathbb{Z}^N \times \mathbb{Z}^N$, the Algorithm 4 computes* $\text{wrs}(R)$ *in $O(N^4)$ time.*

Proof: Note that for all $k_1 \geq 1$, $k_2 = 2^{\lceil \log_2 k_1 \rceil}$, R^{k_2} can be computed in $\lceil \log_2 k_1 \rceil$ time by iterating the assignment $R \leftarrow R \circ R$. Observe that $5N \geq \log_2(5^{2N}) = 2 \cdot \log_2 5 \cdot n$ for all $n \geq 1$. Thus, $2^{5N} \geq 5^{2N}$ for all $n \geq 1$. Notice that after executing the line 5, $V \equiv R^{n_1}$, $W \equiv R^{n_2}$, where $n_1 = 2^{5N}$ and $n_2 = 2^{5N} + 1$. Clearly, $n_2 > n_1 \geq 5^{2N}$.

Consider the case when R is $*$-consistent. If the test on line 6 succeeds, then $V^{-1} = R^{-n_1} = \text{wrs}(R)$ by Lemma 6.7 and the algorithm returns correct result. If the test fails, then $R^{-n_1}(\mathbb{Z}^N) \neq R^{-n_2}(\mathbb{Z}^N)$ and consequently, $\text{wrs}(R) = \emptyset$ by Lemma 6.12 (the direction $3 \Rightarrow 1$) and the algorithm returns correct result. Next, consider the case when R is not $*$-consistent. If the test on line 6 succeeds, then $V^{-1}(\mathbb{Z}^N) = R^{-n_1}(\mathbb{Z}^N) = \text{wrs}(R)$ by Lemma 6.7 and the algorithm returns correct result. If the test fails, the algorithm returns $\emptyset$ which is correct, since $\text{wrs}(R) = \emptyset$ for each relation that is not $*$-consistent.

On the complexity side, since both octagonal composition (lines 4 and 5) can be computed in $O(N^3)$ time, inclusion (line 6) in $O(N^2)$ time, and pre-image (line 6) is computed as a matrix projection in $O(N^2)$ time, the algorithm runs in $O(N^4)$ time. □

6.2.2 On the Existence of Linear Ranking Functions

A ranking function for a given relation R constitutes a proof of the fact that R is well-founded. We distinguish here two cases. If R is not $*$-consistent, then the well-foundedness of R is witnessed simply by an integer constant $i > 0$ such that $R^i = \emptyset$. Otherwise, if R is $*$-consistent, we need a better argument for well-foundedness. In this section, we show that for any $*$-consistent well-founded octagonal relation $R(\mathbf{x}, \mathbf{x}')$, $\mathbf{x} = \{x_1, \ldots, x_N\}$, with prefix b, the (strengthened) relation defined by $(\exists \mathbf{x}'.R^B) \wedge R$, where $B = \min\{b, 5^{2N}\}$ is well-founded and has a linear ranking function even when R alone does not have one.

Definition 6.14 *Given a relation defined by $R(\mathbf{x}, \mathbf{x}')$, a* linear ranking function *for R is a term $f(\mathbf{x}) = \sum_{i=1}^{n} a_i x_i$ such that for all valuations $\nu, \nu' : \mathbf{x} \to \mathbb{Z}$:*

1. *f is* decreasing*: if $\nu, \nu' \models R(\mathbf{x}, \mathbf{x}')$, then $f(\nu) > f(\nu')$,*
2. *f is* bounded*: if $\nu, \nu' \models R(\mathbf{x}, \mathbf{x}')$, then $f(\nu) > h$ and $f(\nu') > h$ for some $h \in \mathbb{Z}$.*

The main result of this section is the following:

Theorem 6.15 *Let $R(\mathbf{x}, \mathbf{x}')$ be a $*$-consistent octagonal relation, with prefix $b \geq 0$. Then, letting $B = \min\{b, 5^{2N}\}$, R is well-founded if and only if the relation defined by $(\exists \mathbf{x}'.R^B) \wedge R$ is well founded if and only if $(\exists \mathbf{x}'.R^B) \wedge R$ has a linear ranking function.*

The first part of the theorem is proved by the following lemma:

Lemma 6.16 *Let $R(\mathbf{x}, \mathbf{x}')$ be a relation, and $m > 0$ be an integer. Then $\text{wrs}(R) = \emptyset$ if and only if $\text{wrs}(R_m) = \emptyset$, where R_m is the relation defined by $(\exists \mathbf{x}'.R^m) \wedge R$.*

Proof: "$\Rightarrow$" By the fact that $R \Leftarrow (\exists \mathbf{x}'.R^m) \wedge R$ and the monotonicity of wrs. "$\Leftarrow$" We prove the dual. Assume that $\text{wrs}(R) \neq \emptyset$ i.e., there exists an infinite sequence of valuations $\sigma = \{\nu : \mathbf{x} \to \mathbb{Z}\}_{i \geq 0}$ such that $(\nu_i(\mathbf{x}), \nu_{i+1}(\mathbf{x})) \in R$, for all $i \geq 0$. Then all $\nu_i(\mathbf{x})$ belong to the set defined by $\exists \mathbf{x}'.R^m$, hence σ is an infinite sequence for the relation defined by $(\exists \mathbf{x}'.R^m) \wedge R$ as well. □

It remains to prove that the witness relation defined by $(\exists \mathbf{x}'.R^B) \wedge R$ has a linear ranking function, provided that it is well-founded. The proof is organized as follows. We first prove the existence of such function for difference bounds relations. By Lemma 6.12, if the difference bounds relation R is well-founded, then the zigzag automaton Z_R must have a cycle of negative weight. Lemma 6.17 and 6.19 use the structure of this cycle, representing several of the constraints in R, to show the existence of the linear ranking function for the witness relation $(\exists \mathbf{x}'.R^B) \wedge R$. Second, using the result of Lemma 6.12 on equivalence of well-foundedness of an octagon R and its difference bounds representation $\overline{R}$, we prove the existence of linear ranking function for octagons in Lemma 6.22.

Linear Ranking Function for Difference Bounds Relation

We first prove the existence of a linear decreasing function, based on the existence of a negative weight cycle in the zigzag automaton.

Lemma 6.17 *Let $R(\mathbf{x}, \mathbf{x}')$, $\mathbf{x} = \{x_1, \ldots, x_N\}$ be a $*$-consistent and well-founded difference bounds relation with prefix $b \geq 0$. Then, there exists a linear function $f(\mathbf{x})$ such that for all valuations $\nu : \mathbf{x} \to \mathbb{Z}, \nu' : \mathbf{x}' \to \mathbb{Z}$ satisfying $\nu, \nu' \models R(\mathbf{x}, \mathbf{x}')$, we have $f(s) > f(s')$.*

Proof: By Lemma 6.12, there exists a path $\sigma.\lambda.\sigma'$ in $Z_{\overline{R}}$ of the form $\sigma : I^{ef}_{i,j} \rightsquigarrow q$, $\lambda : q_1 \rightsquigarrow q_1$, $\sigma' : q_1 \rightsquigarrow F^{ef}$ for some $1 \leq i, j \leq 2N$ and a control state q such that $w(\lambda) < 0$. Let denote the length of λ by p and let write λ as $\lambda = q_1 \xrightarrow{G_1} q_2 \ldots q_p \xrightarrow{G_p} q_1$. Let $G_j = (\mathbf{x} \cup \mathbf{x}', E_j)$ for all $1 \leq j \leq p$.

Consider the following sum of all constraints represented by edges appearing in the zigzag cycle (note that the sum of weights of these edges equals $w(\lambda)$):

$$\sum_{1 \leq j \leq p} \left(\sum_{(x_i \to x'_j) \in E_j} (x_i - x'_j) + \sum_{(x'_i \to x_j) \in E_j} (x'_i - x_j) \right) \leq w(\lambda) \tag{6.3}$$

The left-hand side of (6.3) can be written equivalently as

$$\sum_{1 \leq j \leq p} \left(\sum_{\substack{1 \leq i \leq n, \\ (q_j)_i = r}} (x_i - x'_i) + \sum_{\substack{1 \leq i \leq n, \\ (q_j)_i = l}} (-x_i + x'_i) + \sum_{\substack{1 \leq i \leq n, \\ (q_j)_i = lr}} (-x_i + x_i) + \sum_{\substack{1 \leq i \leq n, \\ (q_j)_i = rl}} (-x'_i + x'_i) \right) \tag{6.4}$$

and thus, after simplifications ($-x_i + x_i = 0, -x'_i + x'_i = 0$), (6.3) can be written equivalently as

$$\sum_{1 \leq j \leq p} \left(\sum_{\substack{1 \leq i \leq n, \\ (q_j)_i = r}} (x_i - x'_i) + \sum_{\substack{1 \leq i \leq n, \\ (q_j)_i = l}} (-x_i + x'_i) \right) \leq w(\lambda) \tag{6.5}$$

Let f denote the negated sum of all unprimed terms in (6.4) and f' denote the sum of all primed terms in (6.4). Then, clearly $f' = f[\mathbf{x}'/\mathbf{x}]$. Thus, (6.5) can be written as

$$f' - f \leq w(\lambda) \tag{6.6}$$

Notice that since $w(\lambda) < 0$, we establish that $f' - f < 0$ hence f is strictly decreasing. Formally $f(s) > f'(s)$ for all valuations s, s' such that $s, s' \models R(\mathbf{x}, \mathbf{x}')$. Since $(\exists \mathbf{x}'.R^B) \wedge R \Rightarrow R$, then clearly $f(\nu) > f(\nu')$ for all valuations ν, ν' such that $\nu, \nu' \models (\exists \mathbf{x}'.R^B) \wedge R$. □

Example (ctd.). We illustrate the construction of linear decreasing function. R is well-founded and by Lemma 6.12, there exists a path depicted in Figure 6.1 (d) where $w(\lambda) < 0$. We follow the construction from Lemma 6.17 and sum the edges in λ. We obtain $x_1 - x_3' + x_3 - x_2' + x_2 - x_1' + x_4' - x_4 + x_4' - x_4 + x_4' - x_4 \leq -1$, which simplifies to $x_1 + x_2 + x_3 - 3x_4 - (x_1' + x_2' + x_3' - 3x_4) \leq -1$. Letting $f(\mathbf{x}) = -(x_1 + x_2 + x_3 - 3x_4)$, we have that $f(\mathbf{x}) > f(\mathbf{x}')$. □

The next auxiliary lemma proves that if the difference $x_i - x_j$ is bounded in R^k for some $k \geq 1$, it is bounded in R^B too.

Lemma 6.18 *Let $R(\mathbf{x}, \mathbf{x}')$, $\mathbf{x} = \{x_1, \ldots, x_N\}$ be a $*$-consistent difference bounds relation with prefix $b \geq 0$ and period $c > 0$. Then, for any $1 \leq i, j \leq N$ and $k \geq 0$, we have $(M^*_{R^k})_{i,j} < \infty \rightarrow (M^*_{R^B})_{i,j} < \infty$, where $B = \min\{b, 5^N\}$.*

Proof: (Case $0 \leq k \leq B$) By monotonicity of pre_R, $(M^*_{R^k})_{i,j} \geq (M^*_{R^B})_{i,j}$. Thus if $(M^*_{R^k})_{i,j} < \infty$, then clearly $(M^*_{R^B})_{i,j} < \infty$.

(Case $k > b$) Let $p = \lceil \frac{k-b}{c} \rceil$, and $k' = b + pc$. Note that $R^{k'} = \widehat{R}_{b,c}(\mathbf{x}, \mathbf{x}', \ell)[p/\ell]$, where $\widehat{R}_{b,c}(\mathbf{x}, \mathbf{x}', \ell)[p/\ell]$ is the closed form of $\{R^{b+ci}\}_{i \geq 0}$. Since $k' \geq k$, by the same argument as for case $(1 \leq k \leq b)$, $(M^*_{R^{k'}})_{i,j} < \infty$. Since $k' = b + pc$, then $x_i - x_j \leq a\ell + d$, where $a, d \in \mathbb{Z}$, is one of its conjuncts. By definition of $\widehat{R}_{b,c}(\mathbf{x}, \mathbf{x}', \ell)[p/\ell]$, we have $(M^*_{R^b})_{i,j} = a \cdot 0 + d = d < \infty$.

(Case $k > 5^N$) By Proposition 6.11, there exists a path $\pi = \sigma_1.\sigma_2.\sigma_3.\sigma_4$ in $Z_{\overline{R}}$ of length $k + 2$ such that $w(\pi) = (M^*_{R^k})_{i,j}$. Let σ_2' be a path obtained by erasing all cycles from σ_2 and let construct $\pi' = \sigma_1.\sigma_2'.\sigma_3$. Let $h = |\sigma_2'|$. Consequently, $(M^*_{R^h})_{i,j} \leq w(\pi') < \infty$. Since $h \leq 5^N$, then $\infty > (M^*_{R^h})_{i,j} \geq (M^*_{R^{5^N}})_{i,j}$ by monotonicity of pre_R. □

Last, we prove that the function f of Lemma 6.17 is bounded, concluding that it is indeed a ranking function. Since each run in the zigzag automaton recognizes a path from some x_i to some x_j, a run that repeats a cycle can be decomposed into a prefix, the cycle itself and a suffix. The recognized path may traverse the cycle several times, however each exit point from the cycle must match a subsequent entry point. These paths from the exit to the corresponding entries give us the necessary lower bound. In fact, these paths appear already on graphs $\mathcal{G}_{R^i}$ for $i \geq B$, where b is the prefix of R and $B = \min\{b, 5^N\}$. Hence the need for a strengthened witness $(\exists \mathbf{x}'.R^B) \wedge R$, as R alone is not enough for proving boundedness of f.

Lemma 6.19 *Let $R(\mathbf{x}, \mathbf{x}')$, $\mathbf{x} = \{x_1, \ldots, x_N\}$ be a $*$-consistent and well-founded difference bounds relation with prefix $b \geq 0$. Then, letting $B = \min\{b, 5^N\}$, there exists a linear ranking function for $(\exists \mathbf{x}'.R^B) \wedge R$.*

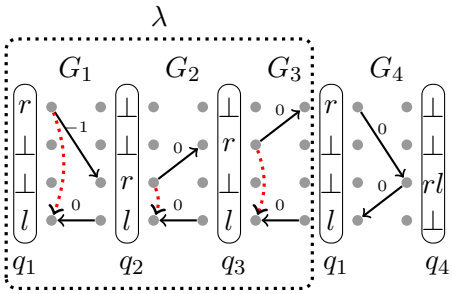


Figure 6.2: Constructing the ranking function.

Proof: Let f be a linear decreasing function from Lemma 6.17. Let $\lambda : q_1 \xrightarrow{G_1} q_2 \dots q_p \xrightarrow{G_p} q_1$ be the negative cycle used to construct f, and λ_F be the suffix from Lemma 6.17. By construction of the zigzag automaton, for any $1 \leq j \leq p$,

$$|\{i \mid (q_j)_i = r\}| = |\{i \mid (q_j)_i = l\}|$$

It follows from (6.5) that each $(q_j)_i = r$ contributes to f with a term $-x_i$ and that each $(q_j)_i = l$ contributes to f with a term $+x_i$ and that each $(q_j)_i \notin \{r, l\}$ doesn't contribute at all. We now demonstrate that for each $1 \leq j \leq p$, there exists a bijective matching $\beta_j : \{i \mid (q_j)_i = r\} \to \{i \mid (q_j)_i = l\}$ such that for any $1 \leq i_1 \leq n$ s.t. $\beta_j(i_1) = i_2$, the difference $x_{i_2} - x_{i_1}$ is bounded in $(\exists \mathbf{x}'.R^B) \wedge R$, formally $(\exists \mathbf{x}'.R^B) \wedge R \Rightarrow (x_{i_2} - x_{i_1} \geq h)$ for some $h \in \mathbb{Z}$.

Let $j \in \{1, \dots, p\}$. By construction of the zigzag automaton, the concatenated graph $G_j G_{j+1} \dots G_p \lambda_F$ connects each $(q_j)_{i_1}$ s.t. $(q_j)_{i_1} = r$ with a unique $(q_j)_{i_2}$ s.t. $(q_j)_{i_2} = l$. This induces the required bijection β_j. Since $G_j G_{j+1} \dots G_p \lambda_F$ is a subgraph of $\mathcal{G}_R^{p+|\lambda_F|}$, it follows that there is a path $\mathbf{x}_{i_1}^{(0)} \rightsquigarrow \mathbf{x}_{i_2}^{(0)}$ in $\mathcal{G}_R^{p+|\lambda_F|}$, in other words, $R^{p+|\lambda_F|} \Rightarrow x_{i_1} - x_{i_2} \leq h$ for some $h \in \mathbb{Z}$. By Lemma 6.18, $R^B \Rightarrow x_{i_1} - x_{i_2} \leq h'$ for some $h' \in \mathbb{Z}$ too. Clearly, $(\exists \mathbf{x}'.R^B) \wedge R \Rightarrow x_{i_1} - x_{i_2} \leq h'$ too. Since $x_{i_1} - x_{i_2} \leq h'$ if and only if $x_{i_2} - x_{i_1} \geq -h'$, we obtain the required property.

Now since $f = \sum_{1 \leq j \leq p} \sum_{\substack{1 \leq i_1, i_2 \leq n \\ \beta_j(i_1) = i_2}} (x_{i_2} - x_{i_1})$ and since we proved that each of the differences $x_{i_2} - x_{i_1}$ in the sum is bounded in $(\exists \mathbf{x}'.R^B) \wedge R$, it follows that f is bounded in $(\exists \mathbf{x}'.R^B) \wedge R$ too, formally $(\exists \mathbf{x}'.R^B) \wedge R \Rightarrow (f \geq h)$ for some $h \in \mathbb{Z}$.

By Lemma 6.17, f is decreasing for R. Thus, f is decreasing fora stronger relation $(\exists \mathbf{x}'.R^B) \wedge R$ too, since $(\exists \mathbf{x}'.R^B) \wedge R \Rightarrow R$. Thus, f is both decreasing and bounded for $(\exists \mathbf{x}'.R^B) \wedge R$ and is a ranking function for $(\exists \mathbf{x}'.R^B) \wedge R$. □

Example (ctd.). We illustrate the boundedness of $f = -(x_1 + x_2 + x_3 - 3x_4)$ (see Figure 6.2). First, compute $B = \min\{b, 5^N\} = \min\{3, 5^4\} = 3$. Since there is a path $\mathbf{x}_2^{(6)} \rightsquigarrow \mathbf{x}_4^{(6)}$ in $G_3 G_4$ (and hence in $\mathcal{G}_R^2$), then $R^2 \Rightarrow (x_2 - x_4 \leq -1)$, and by Lemma 6.18, we obtain

$R^B \Rightarrow (x_2 - x_4 \leq -1)$. Similarly, since there is a path $\mathbf{x}_3^{(5)} \rightsquigarrow \mathbf{x}_4^{(5)}$ in $G_2G_3G_4$ (and hence in $\mathcal{G}_R^3$), we obtain $R^B \Rightarrow (x_3 - x_4 \leq -1)$. Similarly, since there is a path $\mathbf{x}_1^{(4)} \rightsquigarrow \mathbf{x}_4^{(4)}$ in $G_1G_2G_3G_4$ (and hence in $\mathcal{G}_R^4$), we obtain $R^B \Rightarrow (x_1 - x_4 \leq -1)$. Summing up these inequalities, we obtain that $f(\mathbf{x}) = -(x_1 + x_2 + x_3 - 3x_4) \geq 3$ and, thus $(\exists \mathbf{x}'.R^B) \wedge R \Rightarrow (f \geq 3)$.

As an experiment, we have tried the RankFinder [PR04a] tool (complete for linear ranking functions), which failed to discover a ranking function on this example. This comes with no surprise, since no linear decreasing function that is bounded after the first iteration exists. However, RankFinder finds a ranking function for the witness relation $(\exists \mathbf{x}'.R^B) \wedge R$ instead. □

Linear Ranking Function for Octagonal Relation

In the rest of this section, let us fix the sets of variable $\mathbf{x} = \{x_1, \ldots, x_N\}$ and $\mathbf{y} = \{y_1, \ldots, y_{2N}\}$ for some $N \geq 1$. We first prove two technical propositions.

Proposition 6.20 *Let $R(\mathbf{x}, \mathbf{x}')$ be an octagon, $\overline{R}(\mathbf{y}, \mathbf{y}')$ be its difference bounds encoding and let $\overline{f}(\mathbf{y})$ be a linear ranking function for $\overline{R}(\mathbf{y}, \mathbf{y}')$. Then, the function $f \stackrel{def}{=} \overline{f}[x_i/y_{2i-1}, -x_i/y_{2i}]_{i=1}^N$, is a linear ranking function for $R(\mathbf{x}, \mathbf{x}')$.*

Proof: If $\overline{f}$ is decreasing, then $\overline{f}(\overline{\mathbf{v}}) > \overline{f}(\overline{\mathbf{v}}')$ for each $(\overline{\mathbf{v}}, \overline{\mathbf{v}}') \models \overline{R}(\mathbf{y}, \mathbf{y}')$, and thus clearly, for all valuations $(\overline{\mathbf{v}}, \overline{\mathbf{v}}') \models \overline{R}(\mathbf{y}, \mathbf{y}')$ of the form

$$\overline{\mathbf{v}} = (v_1, -v_1, \ldots, v_N, -v_N), \qquad \overline{\mathbf{v}}' = (v_1', -v_1', \ldots, v_N', -v_N').$$

Recall that by Equation (2.1),

$$\phi(\mathbf{x}) \Leftrightarrow (\exists y_2, y_4, \ldots, y_{2N} \,.\, \overline{\phi} \wedge \bigwedge_{i=1}^{N} y_{2i-1} = -y_{2i})[x_i/y_{2i-1}]_{i=1}^N$$

for each octagonal constraint ϕ. Defining $f \stackrel{def}{=} \overline{f}[x_i/y_{2i-1}, -x_i/y_{2i}]_{i=1}^N$, it follows by the above observations that $f(\mathbf{v}) > f(\mathbf{v}')$ for each $(\mathbf{v}, \mathbf{v}') \models R(\mathbf{x}, \mathbf{x}')$. Hence, f is decreasing too. Similarly, we can prove that f is bounded as well. Since f is clearly linear by definition, it follows that it is a linear ranking function for $R(\mathbf{x}, \mathbf{x}')$. □

Proposition 6.21 *Let $R(\mathbf{x}, \mathbf{x}')$ be an octagon, $\overline{R}(\mathbf{y}, \mathbf{y}')$ be its difference bounds encoding and let $\overline{f}(\mathbf{y})$ be a linear ranking function for $(\exists \mathbf{y}'.\overline{R}^m) \wedge \overline{R}$, $m \geq 0$. Then, $f \stackrel{def}{=} \overline{f}[x_i/y_{2i-1}, -x_i/y_{2i}]_{i=1}^n$ is a linear ranking function for $(\exists \mathbf{x}'.R^m) \wedge R$.*

Proof: Let α be a function which syntactically transforms a given octagonal relation $R(\mathbf{x}, \mathbf{x}')$ to its difference bounds representation $\overline{R}(\mathbf{y}, \mathbf{y}')$ induced by Definition 2.18. Let γ be the inverse of α that syntactically transforms $\overline{R}(\mathbf{y}, \mathbf{y}')$ to $R(\mathbf{x}, \mathbf{x}')$ as $R(\mathbf{x}, \mathbf{x}') = \overline{R}(\mathbf{y}, \mathbf{y}')[x_i/y_{2i-1}, -x_i/y_{2i}]_{i=1}^N$. Clearly, $R \equiv \gamma(\alpha(R))$ for each relation and

both α and γ distribute over conjunction. Furthermore, $R^m \Rightarrow \gamma(\alpha(R)^m)$ by Theorem 2.26. Consequently, $(\exists \mathbf{x}'.R^m) \Rightarrow \gamma(\exists \mathbf{y}'.\alpha(R)^m)$.

By the hypothesis, $\overline{f}(\mathbf{y})$ is a linear ranking function for a difference bounds relation $\overline{S}(\mathbf{y}, \mathbf{y}')$ defined as $\overline{S}(\mathbf{y}, \mathbf{y}') \equiv (\exists \mathbf{y}'.\alpha(R)^m) \wedge \alpha(R)$. Let define $S(\mathbf{x}, \mathbf{x}') \equiv \gamma(\overline{S})$. By Proposition 6.20, f is a linear ranking function for S. Since $S(\mathbf{x}, \mathbf{x}') \equiv \gamma(\exists \mathbf{y}'.\alpha(R)^m) \wedge R$ and by earlier arguments, $(\exists \mathbf{x}'.R^m) \Rightarrow \gamma(\exists \mathbf{y}'.\alpha(R)^m)$, we infer that $(\exists \mathbf{x}'.R^m) \wedge R \Rightarrow S$. Consequently, since f is a linear ranking function for S, it is a linear ranking function for $(\exists \mathbf{x}'.R^m) \wedge R$ too. □

Finally, we show that for each $*$-consistent and well-founded octagonal relation, the corresponding witness relation has a linear ranking function.

Lemma 6.22 *Let $R(\mathbf{x}, \mathbf{x}')$ be a $*$-consistent and well-founded octagonal relation with prefix b. Then, letting $B = \min\{b, 5^{2N}\}$, there exists a linear ranking function for $(\exists \mathbf{x}'.R^B) \wedge R$.*

Proof: By Lemma 6.12, $\overline{R}(\mathbf{y}, \mathbf{y}')$ is well-founded too and moreover, $\overline{R}(\mathbf{y}, \mathbf{y}')$ is $*$-consistent by Theorem 2.20. Let $\overline{b}$ be the prefix of $\overline{R}$ and define $\overline{B} = \min\{\overline{b}, 5^{2N}\}$. By Lemma 6.19, there exists a linear ranking function $\overline{f}$ for $(\exists \mathbf{x}'.\overline{R}^{\overline{B}}) \wedge \overline{R}$. By Proposition 6.21, the function $f \stackrel{def}{=} \overline{f}[x_i/y_{2i-1}, -x_i/y_{2i}]_{i=1}^N$ is a linear ranking function for $(\exists \mathbf{x}'.R^{\overline{B}}) \wedge R$. To see that f is a linear ranking function for $(\exists \mathbf{x}'.R^B) \wedge R$ too, consider arbitrary term $y_i - y_j$ considered in the proof of Lemma 6.19. If $b \geq \overline{b}$, then the boundedness argument ($y_i - y_j$ is bounded in R^b) follows by monotonicity of pre_R and is similar to the first case of the proof of Proposition 6.18. If $b < \overline{b}$, one can use a similar argument as in the second case of the proof of Lemma 6.18 and show, using the closed form of R instead of $\overline{R}$, that $y_i - y_j$ is bounded in R^b too. □

6.3 Linear Affine Relations

In this section, we prove that the weakest recurrent set of linear affine relations can be computed as the limit of the Kleene sequence. We further show that this set can be defined in Presburger arithmetic for a subclass of relations with finite monoid property. Next, we relax the finite monoid condition and describe a method for generating sufficient termination conditions, i.e. sets $S \in \mathbb{Z}^n$ such that $S \cap \mathrm{wrs}(R) = \emptyset$, for the class for *polynomially bounded* affine relations.

We first prove that pre_R is $\cap$-continuous for deterministic relations.

Lemma 6.23 *Let $R(\mathbf{x}, \mathbf{x}')$ be a deterministic relation. Then, pre_R is $\cap$-continuous.*

Proof: Let $I = \{1, \ldots, d\}$, $d \in \mathbb{N} \cup \{\infty\}$, and $\{S_i \in \mathbb{Z}^N\}_{i \in I}$ be a potentially infinite collection of sets. We prove that

$$\mathrm{pre}_R(\textstyle\bigcap_{i \in I} S_i) = \bigcap_{i \in I} \mathrm{pre}_R(S_i).$$

"⇒" By monotonicity of pre_R, we have $\mathrm{pre}_R(\bigcap_{i\in I} S_i) \subseteq \mathrm{pre}_R(S_i)$ for all $i \in I$ and hence, $\mathrm{pre}_R(\bigcap_{i\in I} S_i) \subseteq \bigcap_{i\in I} \mathrm{pre}_R(S_i)$. "⇐" Let $v \in \bigcap_{i\in I} \mathrm{pre}_R(S_i)$. Then, there exists $v_i \in S_i$ such that $(v, v_i) \in R$ for all $i \in I$. Since R is deterministic, then $v_1 = v_i$ for all $i \in I$ and hence $v_1 \in \bigcap_{i\in I} S_i$. Consequently, $v \in \mathrm{pre}_R(\bigcap_{i\in I} S_i)$. □

Since linear affine relations are deterministic, the weakest recurrent set of arbitrary linear affine relation R can be computed as the limit of the Kleene sequence $\mathrm{wrs}(R) = \bigcap_{m\geq 0} \mathrm{pre}_R^m(\mathbb{Z}^N)$, by Lemma 6.23 and Lemma 6.8. As discussed in the end of Section 6.1, the weakest recurrent set can be defined using the closed form of R:

$$\mathrm{wrs}(R) \equiv \forall k \geq 0 \;.\; \exists \mathbf{x}' \;.\; \widehat{R}(\mathbf{x}, \mathbf{x}', k)$$

Writing R as $R_u(\mathbf{x}, \mathbf{x}') \wedge \phi(\mathbf{x})$ where $R_u(\mathbf{x}, \mathbf{x}')$ is a deterministic update and $\phi(\mathbf{x})$ is a Presburger guard, we can write the closed form of R, by Lemma 4.28, as

$$\widehat{R}(k, \mathbf{x}, \mathbf{x}') \Leftrightarrow \widehat{R}_u(k, \mathbf{x}, \mathbf{x}') \wedge \forall 0 \leq \ell < k \; \exists \mathbf{y} \;.\; \widehat{R}_u(\ell, \mathbf{x}, \mathbf{y}) \wedge \varphi(\mathbf{y})$$

Then, the definition of the weakest recurrent set of a linear affine relation is (after the elimination of the trailing existential quantifier and renaming ℓ with k and $\mathbf{y}$ with $\mathbf{x}'$):

$$\mathrm{wrs}(R)(\mathbf{x}) \;\equiv\; \forall k \geq 0 \;.\; \exists \mathbf{x}' \;.\; \widehat{R}_u(k, \mathbf{x}, \mathbf{x}') \wedge \varphi(\mathbf{x}') \tag{6.7}$$

The main difficulty with the form (6.7) comes from the fact that the powers of a matrix A cannot usually be defined in a known decidable theory of arithmetic.

In Section 4.4, we proved that $\widehat{R}_u(k, \mathbf{x}, \mathbf{x}')$ is Presburger definable for linear affine relations with finite monoid property. It follows immediately from equation 6.7 that $\mathrm{wrs}(R)$ is Presburger definable and thus, that termination problem is decidable for finite monoid affine relations. We summarize these observations in the following theorem.

Theorem 6.24 *The termination problem is decidable for finite monoid affine relations. Moreover, the weakest termination set of a finite monoid affine relation is an effectively computable Presburger formula.*

6.3.1 Polynomially Bounded Affine Relations

The closed form of polynomially bounded affine relations cannot be defined in Presburger arithmetic any longer, thus we renounce defining $\mathrm{wrs}(R)$ precisely, and content ourselves with the discovery of *sufficient conditions for termination*. Basically, given a linear affine relation R, we aim at finding a disjunction $\phi(\mathbf{x})$ of linear constraints on $\mathbf{x}$, such that $\phi \wedge \mathrm{wrs}(R)$ is inconsistent without explicitly computing $\mathrm{wrs}(R)$. For this, we use several existing results from linear algebra (see, e.g., [Eve03]). In the following, it is convenient to work with the equivalent homogeneous form:

$$\begin{gathered} R(\mathbf{x}, \mathbf{x}') \equiv C_h \mathbf{x}_h \geq \mathbf{0} \;\wedge\; \mathbf{x}'_h = A_h \mathbf{x}_h \\ A_h = \begin{pmatrix} a & \mathbf{b} \\ 0 & 1 \end{pmatrix} \; C_h = \begin{pmatrix} C & -\mathbf{d} \end{pmatrix} \; \mathbf{x}_h = \begin{pmatrix} \mathbf{x} \\ x_{N+1} \end{pmatrix} \end{gathered} \tag{8}$$

The weakest recurrent set of R can be then defined as:

$$\text{wrs}(R) \;\equiv\; \exists x_{N+1} \,.\, \forall k \geq 0 \,.\, C_h A_h^k \mathbf{x}_h \geq \mathbf{0} \;\wedge\; x_{N+1} = 1 \tag{6.9}$$

Definition 6.25 *A function $f : \mathbb{N} \to \mathbb{C}$ is said to be a* C-finite recurrence *if and only if:*

$$f(m+d) = a_{d-1} f(m+d-1) + \ldots + a_1 f(m+1) + a_0 f(n),\ \forall m \geq 0$$

for some $d \in \mathbb{N}$ and $a_0, a_1, \ldots, a_{d-1} \in \mathbb{C}$, with $a_{d-1} \neq 0$. The polynomial $x^d - a_{d-1}x^{d-1} - \ldots a_1 x - a_0$ is called the characteristic polynomial *of f.*

A C-finite recurrence always admits a closed form.

Theorem 6.26 ([Eve03]) *The closed form of a C-finite recurrence is:*

$$f(m) = p_1(m)\lambda_1^m + \ldots + p_s(m)\lambda_s^m$$

where $\lambda_1, \ldots, \lambda_s \in \mathbb{C}$ are non-zero distinct roots of the characteristic polynomial of f, and $p_1, \ldots, p_s \in \mathbb{C}[m]$ are polynomials of degree less than the multiplicities of $\lambda_1, \ldots, \lambda_s$, respectively.

Next, we define the closed form for the sequence of powers of A.

Corollary 6.27 *Given a square matrix $A \in \mathbb{Z}^{N \times N}$, we have:*

$$(A^m)_{i,j} = p_{1,i,j}(m)\lambda_1^m + \ldots + p_{s,i,j}(m)\lambda_s^m$$

where $\lambda_1, \ldots, \lambda_s \in \mathbb{C}$ are non-zero distinct eigenvalues of A, and $p_{1,i,j}, \ldots, p_{s,i,j} \in \mathbb{C}[m]$ are polynomials of degree less than the multiplicities of $\lambda_1, \ldots, \lambda_s$, respectively.

Proof: If $\det(A - xI_n) = x^d - a_{d-1}x^{d-1} - \ldots - a_1 x - a_0$ is the characteristic polynomial of A, then we have

$$A^d - a_{d-1}A^{d-1} - \ldots - a_1 A - a_0 = 0$$

by the Cayley-Hamilton Theorem. If we define $f_{i,j}(m) = (A^m)_{i,j}$, then we have

$$\begin{array}{rcl} A^{m+d} & = & a_{d-1}A^{m+d-1} + \ldots + a_1 A^{m+1} + a_0 A^m \\ f_{i,j}(m+d) & = & a_{d-1}f_{i,j}(m+d-1) + \ldots + a_1 f_{i,j}(m+1) + a_0 f_{i,j}(m) \end{array}$$

By Theorem 6.26, we have that

$$(A^m)_{i,j} = p_{1,i,j}(m)\lambda_1^m + \ldots + p_{s,i,j}(m)\lambda_s^m$$

for some polynomials $p_{1,i,j}, \ldots, p_{s,i,j} \in \mathbb{C}[m]$ of degrees less than the multiplicities of $\lambda_1, \ldots, \lambda_s$, respectively. □

Lemma 6.28 *Given a square matrix $A \in \mathbb{Z}^{N \times N}$, whose non-zero eigenvalues are all roots of the unity. Then $(A^m)_{i,j} \in \mathbb{Q}[m]$, for all $1 \leq i, j \leq N$, are effectively computable polynomials with rational coefficients.*

Proof: Assume from now on that all non-zero eigenvalues $\lambda_1, \ldots, \lambda_s$ of A are such that $\lambda_1^{d_1} = \ldots = \lambda_s^{d_s} = 1$, for some integers $d_1, \ldots, d_s > 0$. The method given in [Boi99] for testing the finite monoid condition for A gives also bounds for $d_1, \ldots, d_s$. Then we have $\lambda_1^L = \ldots \lambda_s^L = 1$, where $L = \text{lcm}(d_1, \ldots, d_s)$. As $d_1, \ldots, d_s$ are effectively bounded, so is L. By Corollary 6.27, we have that, if m is a multiple of L, then $(A^m)_{i,j} = p_{i,j}(m)$ for some effectively computable polynomial $p_{i,j} \in \mathbb{C}[m]$ i.e., for m multiple of L, A^m is polynomially definable. But since $p_{i,j}(m)$ assumes real values in an infinity of points $m = kL,\ k > 0$, it must be that its coefficients are all real numbers, i.e. $p_{i,j} \in \mathbb{R}[m]$. Moreover, these coefficients are the solutions of the integer system:

$$\begin{cases} p_{i,j}(L) & = & (A^L)_{i,j} \\ & \ldots & \\ p_{i,j}((d+1)L) & = & (A^{(d+1)L})_{i,j} \end{cases}$$

Clearly, since $A \in \mathbb{Z}^{N\times N}$, $A^p \in \mathbb{Z}^{N\times N}$, for any $p \geq 0$. Hence $p_{i,j} \in \mathbb{Q}[m]$. □

We turn now back to the problem of defining $\text{wrs}(R)$ for linear affine relations R of the form (6.9). First notice that, if all non-zero eigenvalues of A are roots of the unity, then the same holds for A_h (8). By Lemma 6.28, one can find rational polynomials $p_{i,j}(k)$ defining $(A_h^k)_{i,j}$, for all $1 \leq i, j \leq N$. The condition (6.9) resumes to a conjunction of the form:

$$\text{wrs}(R)(\mathbf{x}) \equiv \bigwedge_{i=1}^{n} \forall k \geq 0\ .\ P_i(k, \mathbf{x}) \geq 0 \tag{6.10}$$

where each $P_i = a_{i,d}(\mathbf{x}) \cdot k^d + \ldots + a_{i,1}(\mathbf{x}) \cdot k + a_{i,0}(\mathbf{x})$ is a polynomial in k whose coefficients are the linear combinations $a_{i,d} \in \mathbb{Q}[\mathbf{x}]$. We are looking after a sufficient condition for termination, which is, in this case, any set of valuations of $\mathbf{x}$ that would invalidate (6.10). The following proposition gives sufficient invalidating clauses for each conjunct above. By taking the disjunction of all these clauses we obtain a sufficient termination condition for R.

Lemma 6.29 *Given a polynomial $P(k, \mathbf{x}) = a_d(\mathbf{x}) \cdot k^d + \ldots + a_1(\mathbf{x}) \cdot k + a_0(\mathbf{x})$, there exists $n > 0$ such that $P(n, \mathbf{x}) < 0$ if, for some $i = 0, 1, \ldots, d$, we have $a_{d-i}(\mathbf{x}) < 0$ and $a_d(\mathbf{x}) = a_{d-1}(\mathbf{x}) = \ldots = a_{d-i+1}(\mathbf{x}) = 0$.*

Proof: Assuming the condition $a_{d-i}(\mathbf{x}) < 0$ and $a_d(\mathbf{x}) = a_{d-1}(\mathbf{x}) = \ldots = a_{d-i+1}(\mathbf{x}) = 0$, for some $0 \leq i \leq d$, we have $P(k, \mathbf{x}) = a_{d-i}(\mathbf{x}) \cdot k^d + \ldots + a_1(\mathbf{x}) \cdot k + a_0(\mathbf{x})$. Since the dominant coefficient $a_{d-i}(\mathbf{x})$ is negative, the polynomial will assume only negative values, from some point on. □

Example. Consider the following program [CGLA+08], and its linear transformation matrix A.

$$\begin{array}{l}\text{while } (x \geq 0)\\ \quad x' = x + y\\ \quad y' = y + z\end{array} \qquad A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \qquad A^k = \begin{pmatrix} 1 & k & \frac{k(k-1)}{2} \\ 0 & 1 & k \\ 0 & 0 & 1 \end{pmatrix}$$

The characteristic polynomial of A is $\det(A - \lambda I_3) = (1-\lambda)^3$, hence the only eigenvalue is 1, with multiplicity 3. Then we compute A^k (see above), and $x' = x + k \cdot y + \frac{k(k-1)}{2} z$ gives the value of x after k iterations of the loop. Hence the (precise) non-termination condition is: $\forall k \geq 0 \;.\; \frac{z}{2} \cdot k^2 + (y - \frac{z}{2}) \cdot k + x \geq 0$. A sufficient condition for termination is: $(z < 0) \vee (z = 0 \wedge y < 0) \vee (z = 0 \wedge y = 0 \wedge x < 0)$ □

We can generalize this method further to the case where all eigenvalues of A are of the form $q \cdot r$, with $q \in \mathbb{R}$ and $r \in \mathbb{C}$ being a root of the unity. The main reason for not using this condition from the beginning is that we are, to this point, unaware of its decidability status. With this condition instead, it is sufficient to consider only the eigenvalues with the maximal absolute value, and the polynomials obtained as sums of the polynomial coefficients of these eigenvalues. The result of Lemma 6.28 and the sufficient condition of Lemma 6.29 carry over when using these polynomials instead.

7 Verification of Programs with Integer Data

In this chapter, we show how the techniques developed in the preceding chapters, which deal with several restricted classes of program loops, can be used in verification of larger programs. We presents several algorithms for reachability and termination analysis of non-recursive programs with integer data. Sections 7.1 and 7.2 present techniques for modular reachability and termination analysis, respectively, while Section 7.3 presents an enhancement of a non-modular reachability analysis algorithm based on predicate abstraction.

The reachability analysis method described in Section 7.1 computes *procedure summaries* by tracking relations and computing loop accelerations, using acceleration techniques from Chapter 3 and Chapter 4. Since the considered programs are non-recursive, we can order the procedures topologically with respect to the call graph of a program. Then, summaries of procedures without call transitions can be computed first. Subsequently, the remaining procedures can be analyzed in the given topological order by plugging the computed summaries at the call sites. Finally, we check whether the error summary of the main procedure is empty.

The termination analysis method described in Section 7.2 computes over-approximations of non-termination sets of procedures. Since we consider non-recursive programs, there cannot be an infinite sequence of procedure calls that never return, and hence, each non-terminating run must loop infinitely in one of the procedures. By first computing a *transition invariant* of each procedure, one can then compute an over-approximation of the set of initial configurations from which a run that loops infinitely in that procedure exists. Next, these sets can be propagated to the main procedure by computing their pre-image via summaries of procedures. We show that such sets can be computed by combining techniques from Chapter 6, which compute the weakest non-termination sets for certain classes of loops, with the reachability analysis techniques from Section 7.1.

Finally, Section 7.3 shows how transitive closure computation can help predicate abstraction to deal with the divergence problem by computing inductive interpolants that rule potentially infinite number of spurious counterexamples in one refinement step. The underlying idea is that by accelerating dynamically detected looping patterns in spurious traces, we can analyze multiple concrete traces at once. We further prove that inductive interpolants that rule out multiple traces can be computed directly from classical Craig interpolants and transitive closures of loops.

A discussion of the related work can be found in the respective sections. We defer all experiments with our reachability and termination analysis techniques to Chapter 8.

7.1 Modular Reachability Analysis

This section describes a reachability analysis techniques that tracks relations of the procedures of non-recursive integer programs in order to construct a procedure summary $(\llbracket\mathcal{P}\rrbracket^f_\exists, \llbracket\mathcal{P}\rrbracket^e_\exists)$ defined in Section 2.2. A key to computing a procedure summary is acceleration of disjunctive loops, for which we design a semi-algorithm that attempts to compute effects of all possible interleavings of the disjuncts. Computed summaries can be plugged in at calling procedures, which can be analyzed subsequently. This process continues until the main procedure is analyzed. Then, presence of an erroneous execution is checked. We start by illustrating the approach on an example in Section 7.1.1. Then, Section 7.1.2 presents an algorithm that computes procedure summaries, and Section 7.1.3 presents a semi-algorithm that computes transitive closures of disjunctive loops.

Related Work. Reachability analysis techniques that compute the set of reachable states precisely [Boi99, FL02, AAB00] or using abstractions, such as abstract interpretation and predicate abstraction methods, have been overviewed in Section 1.1. The precise reachability methods from [Boi99, FL02, AAB00] lack modularity. Similarly, the goal-driven search performed in predicate abstraction-based analyzers is not easily amenable to cope with modular verification since the reason for spuriously reaching an error state might reside in the overapproximation of the behavior of a function call. Since the error location is typically not part of that function, it is usually hard to trace the relation between the cause and effect in order to refine the abstraction in the right way. A method that attempts to apply predicate abstraction to programs composed of (possibly recursive) functions is the method of *nested interpolants* [HHP10]. However, this method lacks modularity as it represents the entire programs by nested word automata [AM06], i.e., computation models which are equivalent to the visible pushdown automata [AM04].

Our work focuses on modular program verification by attempting to compute function summaries, without regard to the calling context. On one side, unlike the techniques based on abstract interpretation, we aim at computing precise summary relations that should not require refinement. On the other side, our method, although modular, is computationally more expensive than the error-driven search of predicate abstraction, mostly due to the lack of abstraction (and refinement) in our method. A combination of these two (apparently antithetical) approaches to program verification seems to be the key to a wider application of program verification in real-life software development. In Section 7.3, we present a method that combines predicate abstraction based verification with a precise reachability analysis technique (acceleration). There, we also discuss methods based on abstractions in more detail. The idea of using relations as a domain of program analysis has been also exploited in [PR05], although with the goal of proving program termination rather than safety, which is the aim of this section.

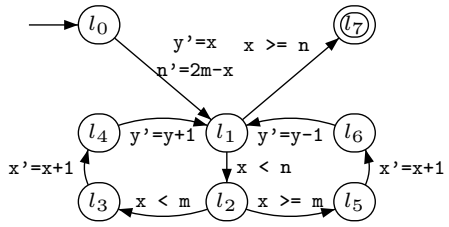
7.1.1 Motivating Example

We give an example of an integer program for which we prove safety (unreachability of the error state) by computing the relation between the values of its variables at the initial and final control locations. Consider the program in Figure 7.1a and the control flow graph of the `fun` function in Figure 7.1b and 7.1c. The function defined in Figure 7.1a

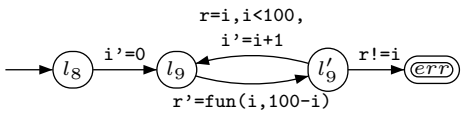
```
int fun(int x, int m) {
l0: int y = x, n = 2*m-x;
l1 : while (x < n) {
l2 :   if (x < m) {
l3 :     x ++;
l4 :     y ++;
       } else {
l5 :     x ++;
l6 :     y --;
       }
     }
l7 : return y;
}

void main() {
l8: for (int i = 0; i < 100; i ++)
l9:   assert(fun(i,100-i) == i);
}
```

(a)



(b)



(c)

Figure 7.1: Example of an integer program and its control flow graph

takes two integer parameters and returns an integer result. The goal of the analysis is to prove that the assertion at line 9 holds whenever the control reaches it. A non-modular method will typically iterate the loop at lines 8 and 9 and analyze the behavior of the `fun` function for each different valuation of the formal parameters `x` and `m`. Our method will instead first infer the error summary relation for the function and then will use this relation in proving all assertions correct at once.

This method is similar to the classical conversion of finite automata into regular expressions. We proceed by eliminating the control states from the control flow graph of the `fun` function (Figure 7.1b) and recording the result of these eliminations on the remaining edges. We start by first eliminating the states without self-loops. Each elimination requires composing all incoming with all outgoing edge relations—the edges labeled by inconsistent relations are not added. First, we eliminate l_3 and l_4, in Figure 7.2a followed by l_5 and l_6, in Figure 7.2b. Next, we eliminate l_2, which causes l_1 to have two self-loops, labeled by the relations $R_1 \Leftrightarrow x < n \wedge x < m \wedge x' = x + 1 \wedge y' = y + 1$ (the left loop in Figure 7.2c) and $R_2 \Leftrightarrow x < n \wedge x \geq m \wedge x' = x + 1 \wedge y' = y - 1$ (the right loop in Figure 7.2c).

At this point, we need to eliminate a control location (l_1) with two self-loops. This step requires the computation of the transitive closure of the disjunctive relation cor-

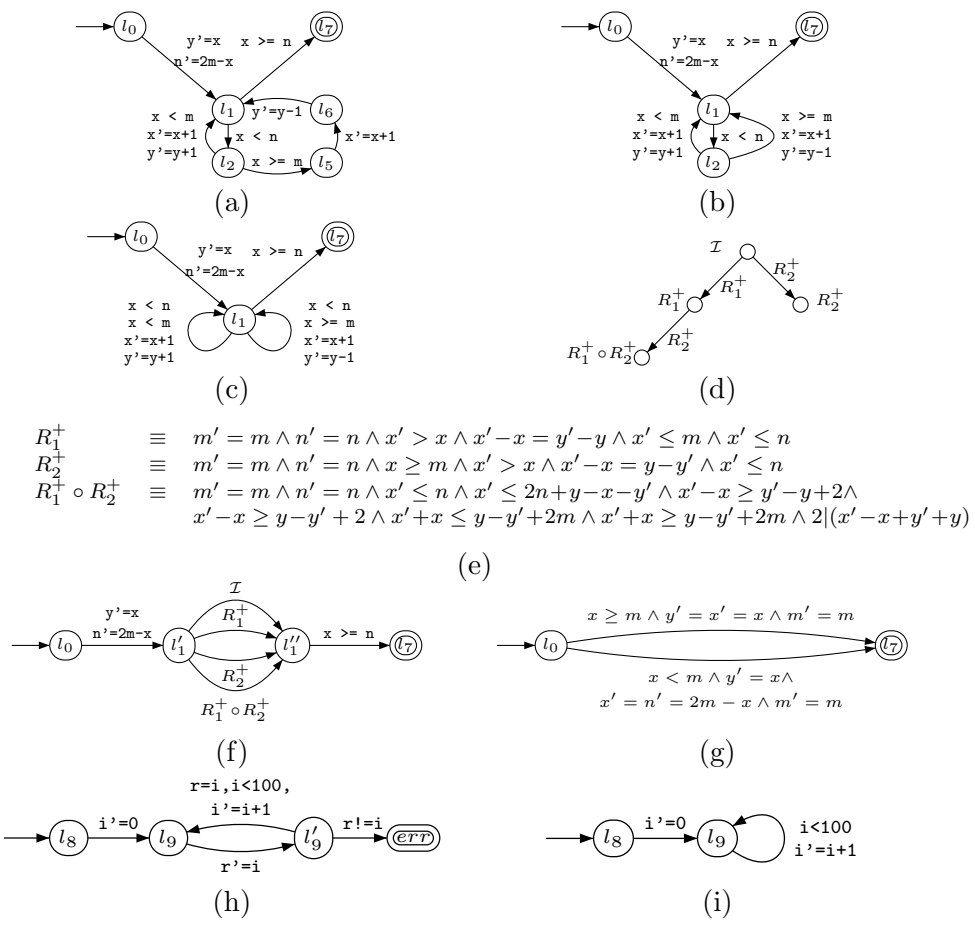


Figure 7.2: Deciding Safety by Elimination of Control Locations

responding to iterating these loops in any possible order. In general, this computation is not bound to yield a result that can be expressed in linear arithmetic (or, for that matter, in any decidable subfragment of first-order arithmetic). Hence, our method has the form of a semi-algorithm.

We first compute transitive closures of the conjunctive relations R_1^+ and R_2^+ (Figure 7.2e). Then we systematically explore all possible interleavings of R_1^+ and R_2^+ by building a tree (breadth-first) whose edges are labeled with the transitive closures of R_1^+ and R_2^+, and each node corresponds to the composition of the transitive closures on the path from the root (labeled with the identity relation $\mathcal{I}$) to the node. Before expanding the tree, the algorithm checks whether the new relation is either (i) inconsistent or (ii) included in the union of the relations labeling the existing nodes. If this test succeeds, the algorithm backtracks, otherwise it adds the new node to the tree. If the transitive closure $(R_1 \vee R_2)^+$ is equivalent to a finite number of interleavings, this construction will terminate, otherwise not. For this example, in Figure 7.2d, the tree construction ends

after three iterations with the result $(R_1 \vee R_2)^+ = R_1^+ \vee R_2^+ \vee R_1^+ \circ R_2^+$, see Figure 7.2e. This is because the composition $R_2^+ \circ R_1^+$ is inconsistent. Next, the location l_1 is split into l_1' and l_1'' (this time, both locations have no self-loops), and there are four edges between l_1' and l_1'' labeled with $\mathcal{I}$, R_1^+, R_2^+ and $R_1^+ \circ R_2^+$, see Figure 7.2f. We further eliminate the locations l_1' and l_1'', see Figure 7.2g. Finally, we eliminate variables not appearing in the signature of the function and obtain the error summary relation of the `fun` function: $(x \geq m \wedge y' = x) \vee (x < m \wedge y' = x) \equiv (y' = x)$.

This relation is now used to check the validity of the assertion at line 9, in Figure 7.1a. Since the call to `fun` on line 9 can be replaced by the summary relation computed above, we can represent the `main` function by a control flow graph and apply the state elimination method in order to establish that the error state (corresponding to a violation of the assertion) is unreachable. First, we substitute the call arguments and return values to the error summary function, see Figure 7.2h. Then we eliminate control state l_9'. Since the composition of the transitions $l_9 \xrightarrow{r'=i} l_9'$ and $l_9' \xrightarrow{r \neq i} err$ is unsatisfiable, the error state is unreachable (Figure 7.2i). We can now conclude that the program is safe with respect to the assertion at line 9.

7.1.2 Computing Program Summaries

Since we consider programs that are not recursive, the call graph of $\mathcal{P}$, $CG(\mathcal{P})$ is a dag, and therefore one can choose a topological ordering of the procedures $\langle P_{i_1}, \ldots, P_{i_n} \rangle$ such that for all $1 \leq k < \ell \leq n$, there is no path from P_{i_k} to P_{i_ℓ} in $CG(\mathcal{P})$.

The Algorithm 5 computes Presburger formulae defining the summaries of the procedures in the given topological order, starting with the procedures that do not have calls to other procedures. Since the program is not recursive, the fixpoint $[\![\mathcal{P}]\!]$ is reached in at most n steps because each procedure needs to be evaluated only once. Once the summary of a procedure is computed, it is used to replace the call transitions involving that procedure in every procedure which is higher in the topological order w.r.t. $CG(\mathcal{P})$ (function ProcSummary, lines 3 and 4). Additionally, the algorithm checks for error traces by also computing the error summary of each procedure and checking the resulting formula for satisfiability. The problem is thus reduced to computing the summary of a procedure without call transitions.

Algorithm 6 implements the function ProcSummaryNoCalls, used to generate the (error) summary of a procedure without call transitions. The idea of this algorithm is to eliminate control states which are neither initial, error, or final, while introducing new transitions labeled with compositions of relations between the remaining states. We iterate the following until no more states can be eliminated: For each control state with (possibly zero) self-loops labeled with relations $R_1, \ldots, R_k$, we compute the transitive closure $T = (R_1 \vee \ldots \vee R_k)^*$. By convention, if $k = 0$, then T is the identity relation. Then we compose the relation of each incoming transition $q_1 \xrightarrow{R} q$ with T and with the relation of each outgoing transition $q \xrightarrow{Q} q_2$ and replace the pair of incoming and outgoing transitions with the transition $q_1 \xrightarrow{P \circ T \circ Q} q_2$, which avoids q. Finally, we eliminate q and

all transitions involving it from the procedure. For an example of a run of this algorithm, the reader may refer to Figure 7.2 (a), (b), (c), (f), and (g).

The termination of Algorithm 6 is clearly determined by the termination of the transitive closure computation function DisjTransitiveClosure since at each iteration of the main loop, the boolean variable `changed` is set to true only if at least one control state is eliminated. Since the set of control states is finite, this implies the termination of the main loop provided that the semi-algorithm DisjTransitiveClosure terminates.

7.1.3 Computing Transitive Closures of Disjunctive Relations

In this section, we present a semi-algorithm computing transitive closures of disjunctive relations. Given a (possibly empty) set of Presburger definable relations $\{R_1, \ldots, R_k\}$, Algorithm 7, if it terminates, returns the reflexive and transitive closure of the disjunctive relation $R = R_1 \vee \ldots \vee R_k$. We assume in the following that the transitive closure R_i^+ of each relation in the set is Presburger definable. The algorithm enumerates all unrollings of the relations $R_1, \ldots, R_k$ and computes increasingly larger underapproximations of R^+. If two successive such underapproximations are equivalent, the algorithm terminates and

Algorithm 5 Program Summary Algorithm

input a program $\mathcal{P} = \langle \mathbf{x}_g, \{P_{i_1}, \ldots, P_{i_n}\}, P_m \rangle$ ordered w.r.t. $CG(\mathcal{P})$
output a summary $([\![\mathcal{P}]\!]^f_\exists, [\![\mathcal{P}]\!]^e_\exists)$ of the program

1: **function** ProgramSummary($\mathcal{P} = \langle \mathbf{x}_g, \{P_{i_1}, \ldots, P_{i_n}\}, P_m \rangle$)
2: $([\![\mathcal{P}]\!]^f_\exists, [\![\mathcal{P}]\!]^e_\exists) \leftarrow (\langle \emptyset, \ldots, \emptyset \rangle, \langle \emptyset, \ldots, \emptyset \rangle)$
3: **for** $k = 1, \ldots, n$ **do**
4: $([\![P_{i_k}]\!]^f_\exists, [\![P_{i_k}]\!]^e_\exists) \leftarrow$ ProcSummary($P_{i_k}, [\![\mathcal{P}]\!]^f_\exists, [\![\mathcal{P}]\!]^e_\exists$)
5: **if** V_m^e is satisfiable **then**
6: **report** "program is unsafe"

input a procedure $P = \langle \mathbf{x}, \overrightarrow{\mathbf{x}}^{in}, \overrightarrow{\mathbf{x}}^{out}, Q, q_0, q_f, q_e, \Delta \rangle$, and
a program summary $([\![\mathcal{P}]\!]^f_\exists, [\![\mathcal{P}]\!]^e_\exists)$
output a summary of P w.r.t. $(\mathcal{V}_f, \mathcal{V}_e)$

1: **function** ProcSummary(P, $(\mathcal{V}_f, \mathcal{V}_e)$)
2: **for each** $q \xrightarrow{\overrightarrow{\mathbf{z}}' = call_j(\overrightarrow{\mathbf{t}})} q' \in \Delta$ **do**
3: $R_f \leftarrow$ PlugSummary$(i, j, [\![\mathcal{P}_i]\!]^f_\exists, \overrightarrow{\mathbf{t}}, \overrightarrow{\mathbf{z}}')$
4: $R_e \leftarrow$ PlugSummary$(i, j, [\![\mathcal{P}_i]\!]^e_\exists, \overrightarrow{\mathbf{t}}, \overrightarrow{\mathbf{z}}')$
5: $\Delta \leftarrow \left(\Delta \setminus \{q \xrightarrow{\overrightarrow{\mathbf{z}}' = call_j(\overrightarrow{\mathbf{t}})} q'\}\right) \cup \{q \xrightarrow{R_f} q',\ q \xrightarrow{R_e} q_e\}$
6: **return** ProcSummaryNoCalls (P)

1: **function** PlugSummary$(i, j, R, \overrightarrow{\mathbf{t}}, \overrightarrow{\mathbf{z}}')$
2: **return** $R\big[(\overrightarrow{\mathbf{t}})_k / (\overrightarrow{\mathbf{x}}_i^{in})_k\big]_{k=1..|\overrightarrow{\mathbf{t}}|} \big[(\overrightarrow{\mathbf{z}})'_k / (\overrightarrow{\mathbf{x}}_i^{out})'_k\big]_{k=1..|\overrightarrow{\mathbf{z}}|} \wedge Id_{(\mathbf{x}_i \setminus \overrightarrow{\mathbf{x}}_i^{out})}$

Algorithm 6 Procedure Summary Algorithm

input a procedure $P = \langle \mathbf{x}, \overrightarrow{\mathbf{x}}^{in}, \overrightarrow{\mathbf{x}}^{out}, Q, q_0, q_f, q_e, \Delta \rangle$ without call transitions
output The summary relations $([\![P]\!]^f_\exists, [\![P]\!]^e_\exists)$

1: **function** ProcSummaryNoCalls(P)
2: changed $\leftarrow$ true
3: **while** changed **do**
4: changed $\leftarrow$ false
5: **for each** $q \in Q \setminus \{q_0, q_f, q_e\}$ with self-loops $q \xrightarrow{R_1} q, \ldots, q \xrightarrow{R_k} q$ **do**
6: $T \leftarrow$ DisjTransitiveClosure($R_1, \ldots, R_k$)
7: **for each** $q_1 \xrightarrow{P} q$ and $q \xrightarrow{Q} q_2$ such that $q_1 \neq q,\ q_2 \neq q$ **do**
8: **if** $\forall q_1 \xrightarrow{R'} q_2 \in \Delta \ .\ P \circ T \circ Q \not\Rightarrow R'$ **then**
9: $\Delta \leftarrow \Delta \cup \{q_1 \xrightarrow{P \circ T \circ Q} q_2\}$
10: changed $\leftarrow$ true
11: $Q \leftarrow Q \setminus \{q\}$
12: $\Delta \leftarrow \Delta \setminus \big(\{q \xrightarrow{R_1} q, \ldots, q \xrightarrow{R_k} q\} \cup \{q' \xrightarrow{R'} q, q \xrightarrow{R''} q'' \mid q', q'' \in Q\}\big)$
13: **return** $\big($Project($\bigvee\{R \mid q_0 \xrightarrow{R} q_f\})$, Project($\bigvee\{R \mid q_i \xrightarrow{R} q_e\})\big)$

1: **function** Project(R)
2: **return** $\exists(\mathbf{x}_i \cup \mathbf{x}'_i) \setminus (\overrightarrow{\mathbf{x}}^{in}_i \cup \overrightarrow{\mathbf{x}}'^{out}_i)\ .\ R$

returns the precise transitive closure.

Algorithm 7 builds breadth-first a tree structure in which each edge corresponds to a relation R_i, and each node corresponds to the composition of the transitive closures of all relations along the path from the root to itself. The algorithm backtracks either when the composition becomes unsatisfiable, or when it is included in the union of the relations corresponding to the nodes which have been already constructed (i.e. if the test on line 7 fails). As an optimization, if a node N is obtained from its parent by composing it with the transitive closure R_i^+, it is not necessary to add the child corresponding to R_i to N since this child would be automatically subsumed by N (line 9). The result of the algorithm is the disjunction of all relations corresponding to nodes in the tree (line 11). For an example of a terminating execution of this algorithm, the reader may refer to Figure 7.2 (d) and (e).

Lemma 7.1 *Let $S = \{R_1, \ldots, R_k\}$ be the input to the Algorithm 7. If the algorithm terminates, then the output is $(R_1 \vee \ldots \vee R_k)^*$.*

Proof: The reflexive and transitive closure of a relation $R = R_1 \vee \ldots \vee R_k$ is the limit of the increasing sequence defined by $S_0 = Id$, and $S_{i+1} = S_i \vee S_i \circ R$, i.e. $R^* = \bigcup_{i=0}^{\infty} S_i$. Let $T_0, T_1, T_2 \ldots$ be the sequence of trees, as generated by the algorithm. Clearly, $T_0 \subseteq T_1 \subseteq T_2 \subseteq \ldots$. We will show that, (1) for each $i \geq 0$, there exists $j \geq 0$, such that

Algorithm 7 Reflexive and Transitive Closure of Disjunctive Relations

input a set of relations $S = \{R_1, \ldots, R_k\}$
output The reflexive and transitive closure $(R_1 \vee \ldots \vee R_k)^*$
$Queue, Tree \leftarrow \emptyset$

```
1: function DisjTransitiveClosure(S)
2:     if S = ∅ then
3:         return Id
4:     Queue.add(Id, ⊥)
5:     while !Queue.empty() do
6:         ⟨N, P⟩ ← Queue.remove()
7:         if N is satisfiable and N ⇏ ⋁Tree then
8:             Tree ← Tree ∪ {N}
9:             for each R ∈ S such that R ≠ P do
10:                Queue.add(N ∘ R⁺, R)
11:    return ⋁Tree
```

$S_i \Rightarrow \bigvee T_j$, and that (2) $\bigvee T_j \Rightarrow R^*$ for all $j \geq 0$. If the algorithm terminates, the sequence T_j has a maximal element T_J, and since for all $i \geq 0$, $S_i \Rightarrow \bigvee T_J$, we have that $R^* \Rightarrow \bigvee T_J$, and therefore $\bigvee T_J = R^*$.

To show (1), let us consider the disjunction S_i for some $i \geq 0$, and let $R = R_{i_1} \circ \ldots \circ R_{i_t}$ be some maximal satisfiable sequence of compositions in S_i. Let $i_{k_1}, \ldots, i_{k_s}$ be the subsequence of $i_1, \ldots, i_t$ obtained by replacing each maximal subsequence $i_j = i_{j+1} = \ldots$ by i_j. Since R is satisfiable, then the path $R^+_{i_{k_1}} \circ \ldots \circ R^+_{i_{k_s}}$ is also satisfiable. As a first remark, every prefix $R^+_{i_{k_1}} \circ \ldots \circ R^+_{i_{k_j}}$, $j \leq s$ is satisfiable. Let us consider the maximal prefix $R^+_{i_{k_1}} \circ \ldots \circ R^+_{i_{k_j}}$ which is represented by a node in some tree $T_0, T_1, \ldots$, and let T_ℓ be the first tree in which this path appears. If $j = s$, then the path appears explicitly in the tree, hence $R \Rightarrow \bigvee T_{j_R}$. Otherwise, if $j < s$, the path $R^+_{i_{k_1}} \circ \ldots \circ R^+_{i_{k_j}} \circ R^+_{i_{k_{j+1}}}$ fails the test on line 7, hence it is subsumed by T_ℓ, i.e. $R^+_{i_{k_1}} \circ \ldots \circ R^+_{i_{k_j}} \circ R^+_{i_{k_{j+1}}} \Rightarrow \bigvee T_\ell$. If $j + 1 = s$, we choose T_ℓ. Otherwise, repeat for all $h = j + 2, \ldots, s$:

- For all maximal paths P in T_ℓ, if $P \circ R_{i_{k_{j+1}}} \circ \ldots \circ R_{i_{k_h}} \not\Rightarrow T_\ell$, then add this path to T_ℓ. The resulting tree will eventually be generated by the algorithm as T_m for some $m > \ell$, and let the new T_ℓ be T_m.

When the iteration ends, we let $T_R = T_\ell$, and we have $R \Rightarrow \bigvee T_\ell$. Since the number of maximal satisfiable sequences in S_i is finite, we can choose the largest such tree T_{j_R} and obtain that $S_i \Rightarrow \bigvee T_{j_R}$. To prove (2), observe that each path $R^+_{j_1} \circ R^+_{j_2} \circ \ldots \circ R^+_{j_k}$ in some T_j implies R^*, hence $\bigvee T_j \Rightarrow R^*$. □

As previously mentioned, Algorithm 5 computing a program summary terminates for non-recursive programs provided that each call to the DisjTransitiveClosure function terminates. In general, this is, however, not true due to the undecidability of the safety problem for integer programs [Min67].

Under- and Overapproximations of Transitive Closures

Termination of the disjunctive closure algorithm can be imposed in two ways, depending on the goal of the analysis. If the goal is proving correctness of the system, i.e. unreachability of the error states, one may resort to overapproximation of the disjunctive loop relations $R_1 \vee \ldots \vee R_k$ by a single, weaker, relation $R^\sharp$ (i.e. $R_1 \vee \ldots \vee R_k \Rightarrow R^\sharp$) such that $T^\sharp = R^{\sharp *}$ is effectively Presburger definable. For instance, if each R_i is a convex polyhedron, one can use Integer Linear Programming [Sch86] to compute the *integer octagonal hull* $R^\sharp$. If using $T^\sharp$ to eliminate the state q in Algorithm 6 suffices to prove unreachability of error states in the program, then this constitutes a valid correctness proof, since every trace of the original program is a trace of the abstract program.

Then, Algorithm 7 can be modified into Algorithm 8, which guarantees termination. The crux of Algorithm 8 is that the expansion of a tree node situated at a depth beyond a certain threshold is done by composition with $R^{\sharp +}$ (line 14), instead of R_i^+ for some $i = 1, \ldots, k$ as in the case of nodes below the threshold (line 11). Since $R_1 \vee \ldots \vee R_k \Rightarrow R^\sharp$, then for any sequence $i_1, \ldots, i_m \in \{1, \ldots, k\}$, the relation $R_{i_1}^+ \circ \ldots \circ R_{i_m}^+$ is subsumed by $R^{\sharp +}$. This prevents future descendants of nodes above the threshold to be added to the queue (the test on line 7 will fail for them), causing termination of the main loop. The returned value in this case is an overapproximation of the reflexive and transitive closure $(R_1 \vee \ldots \vee R_k)^*$.

Algorithm 8 Abstract Reflexive and Transitive Closure of Disjunctive Relations

input a set of relations $S = \{R_1, \ldots, R_k\}$
output a relation R such that $(R_1 \vee \ldots \vee R_k)^* \Rightarrow R$
$Queue, Tree \leftarrow \emptyset$

```
1:  function AbsTransitiveClosure(S)
2:      if S = ∅ then
3:          return Id
4:      Queue.add(Id, ⊥, 0)
5:      while !Queue.empty() do
6:          ⟨N, P, depth⟩ ← Queue.remove()
7:          if N is satisfiable and N ⇏ ⋁Tree then
8:              Tree ← Tree ∪ {N}
9:              if depth ≤ Threshold then
10:                 for each R ∈ S such that R ≠ P do
11:                     Queue.add(N ∘ R⁺, R, depth + 1)
12:             else
13:                 R♯ ← AbstractDisjRelation(S)
14:                 Queue.add(N ∘ R♯⁺, R, depth + 1)
15:     return ⋁Tree
```

On the other hand, if the goal of the analysis is to find errors in the program, the disjunctive closure algorithm can be stopped after a given number of steps, the result

being an underapproximation of the transitive closure, i.e. a relation $T^\flat \Rightarrow (R_1 \vee \ldots \vee R_k)^*$. Even if they cannot be used to certify correctness of systems, underapproximations play an important role in finding errors within complex systems since every error trace found using underapproximations is a valid error trace of the program.

A direct consequence of the proof of Lemma 7.1 is that the disjunction of the nodes in each tree built by the DisjTransitiveClosure function is an underapproximation of the transitive closure. Thus the algorithm can be stopped if, e.g., the number of nodes reaches a predefined threshold, and the result will be a relation stronger than $(R_1 \vee \ldots \vee R_k)^*$. This relation can be used in Algorithm 6 to underapproximate the summary of a procedure and thus the summary of a program. If insufficient, the underapproximation can be improved by letting the DisjTransitiveClosure algorithm run longer.

7.2 Modular Termination Analysis

This section presents a method that computes (an over-approximation of) the weakest non-termination set $[\![P_m]\!]^{nt}$ of the main procedure P_m of a non-recursive program $\mathcal{P} = \langle \mathbf{x}_g, \{P_1, \ldots, P_n\}, P_m \rangle$. Its negation is thus a termination precondition of the program.

The method can be summarized as follows. For simplicity, suppose first that R is the (disjunctive) transition relation of a procedure without call transitions and that R encodes the program counter. Our method first computes (1) a *transition invariant*, i.e. a relation $R_1^\# \vee \ldots \vee R_m^\#$ which over-approximates the transitive closure of R restricted to reachable states, and (2) a *reachability relation*, i.e., a relation which over-approximates $(R^+ \wedge Init)$. Next, we compute the weakest non-termination set $\mathrm{WNT}(R_i^\#)$ of each disjunct $R_i^\#$, by applying methods from Chapter 6. We prove that computing the pre-image of $\mathrm{WNT}(R_1^\#) \vee \ldots \vee \mathrm{WNT}(R_m^\#)$ via the reachability relation gives an over-approximation of the weakest non-termination set of the procedure. We apply Algorithm 4 to compute the weakest non-termination sets $\mathrm{WNT}(R_i^\#)$. A transition invariant and a reachability relation can be computed by a slight adaptation of the techniques described in Section 7.1.

Next, we show that the above approach can be generalized to non-recursive programs with multiple procedures. Since each infinite run of non-recursive program loops infinitely in one of its procedures, one can compute over-approximations of non-termination sets for each procedures by applying ideas from the previous paragraph, and then propagate these sets to the main procedure by computing their pre-image via summaries of procedures.

The results of this section can be seen as a generalization of a result by Podelski and Rybalchenko [PR04b] who proved that disjunctive well-foundedness of a transition invariant of a program P entails termination of that program. We extend their result to computation of over-approximations of non-termination sets.

7.2.1 Transition Invariants and Non-termination Sets

In this section, we present a method that computes, for a given procedure P_i of a non-recursive program, (an over-approximation of) the set of initial configurations from which runs that loop infinitely in P_i exist. Recall from Section 2.2.2 that $\mathcal{V}_i = \{\nu : \mathbf{x}_i \cup \mathbf{x}_g \to \mathbb{Z}\}$ denotes the set of valuations of variables visible in procedure P_i, that $\mathcal{S}_i = \{S_i : Q_i \times Q_i \to \mathcal{V}_i \times \mathcal{V}_i\}$ denotes the set of summaries compatible with P_i, that $[\![\mathcal{P}]\!] = ([\![P_1]\!], \ldots, [\![P_n]\!]) \in \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$ denotes the reachability summary in $k \geq 1$ steps, and that $[\![\mathcal{P}]\!]^* = ([\![P_1]\!]^*, \ldots, [\![P_n]\!]^*)) \in \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$ denotes the reachability summary in $k \geq 0$ steps. Then, the set of valuations in a control state $q \in Q_i$ of a procedure P_i that are reachable from the initial control state $q_{0,i}$ can be defined as the post-image of $\mathcal{V}_i$ via $[\![P_i]\!]^*(q_{0,i}, q)$, formally $\big([\![P_i]\!]^*(q_{0,i}, q)\big)(\mathcal{V}_i)$. With this notation, the *precise transition invariant* $[\![\mathcal{P}]\!]^{TInv}$ of a program $\mathcal{P}$ is defined as

$$[\![\mathcal{P}]\!]^{TInv} = ([\![P_1]\!]^{TInv}, \ldots, [\![P_n]\!]^{TInv})$$

where for each procedure P_i and each $q, q' \in Q_i$,

$$[\![P_i]\!]^{TInv}(q, q') \stackrel{def}{=} [\![P_i]\!](q, q') \wedge \big([\![P_i]\!]^*(q_{0,i}, q)\big)(\mathcal{V}_i).$$

Intuitively, the precise transition invariant is obtained by restricting $[\![\mathcal{P}]\!]$ to the set of reachable configurations. Note that consequently, the set of (infinite) runs in P_i under $[\![\mathcal{P}]\!]$ is equal to the set of (infinite) runs in P_i under $[\![\mathcal{P}]\!]^{TInv}$. We say that $\mathcal{T} = (\mathcal{T}_1, \ldots, \mathcal{T}_n) \in \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$ is a *transition invariant* if and only if $[\![P_i]\!]^{TInv}(q, q') \Rightarrow \mathcal{T}_i(q, q')$ for each procedure P_i and for all $q, q' \in Q_i$. Thus, a transition invariant is an over-approximation of the precise transition invariant.

It has been shown in [PR04b] that a precise transition invariant of a procedure P without call transitions is disjunctively well-founded (meaning that each disjunct of the precise transition invariant is well-founded) if and only if P terminates from each initial configuration. The following lemma generalizes this statement to computation of (an over-approximation of) the weakest non-termination set.

Lemma 7.2 *Let $\mathcal{P}$ be a non-recursive program, let $\mathcal{T} = (\mathcal{T}_1, \ldots, \mathcal{T}_n) \in \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$ be a transition invariant of $\mathcal{P}$, $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n) \in \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$ be an over-approximation of $[\![\mathcal{P}]\!]^*$ and let $P_i = \langle \mathbf{x}_i, \overrightarrow{\mathbf{x}}_i^{in}, \overrightarrow{\mathbf{x}}_i^{out}, Q, q_{0,i}, q_{f,i}, q_{e,i}, \Delta_i \rangle$ be one of its procedures. Let $\mathcal{B}_i \subseteq \mathcal{V}_i$ be a set of valuations defined as $\nu \in B$ if and only if there exists an infinite run in P_i under $[\![P]\!]^{TInv}$ that starts in $\langle q_{0,i}, \nu \rangle$. Further, for each $q \in Q_i$, let there be relations $R_{q,k}$, $1 \leq k \leq p_q$, for some $p_q \geq 1$, such that*

$$\mathcal{T}_i(q, q) \Leftrightarrow \bigcup_{k=1}^{p_q} R_{q,k},$$

and define

$$\mathcal{N}_i \stackrel{def}{=} \bigcup_{q \in Q} \left(\big(\mathcal{A}_i(q_{0,i}, q)\big)^{-1} \left(\bigcup_{k=1}^{p_q} wnt(R_{q,k}) \right) \right).$$

Then, $\mathcal{B}_i \subseteq \mathcal{N}_i$. Moreover, if $\mathcal{T}_i = [\![P_i]\!]^{TInv}$ and $\mathcal{A}_i = [\![P_i]\!]^*$, then $\mathcal{B}_i = \mathcal{N}_i$.

Proof: First suppose that both $\mathcal{T}_i$ and $\mathcal{A}_i$ are precise, i.e. that $\mathcal{T}_i = [\![P_i]\!]^{TInv}$ and $\mathcal{A}_i = [\![P_i]\!]^*$.

" $\Rightarrow$ " Let $\rho_1 = \langle q_{0,i}, \nu_0\rangle\langle q_1, \nu_1\rangle\langle q_2, \nu_2\rangle \ldots$ be an infinite run of P_i under $[\![P]\!]^{TInv}$. Clearly, there exist infinitely many integers $1 \leq \ell_1 < \ell_2 < \ell_3 < \ldots$ such that $q = q_{\ell_1} = q_{\ell_2} = q_{\ell_3} = \ldots$ for some $q \in Q_i$. We construct a infinite meta-run $\rho_2 = \langle q_{0,i}, \nu_0\rangle\langle q, \nu_{\ell_1}\rangle\langle q, \nu_{\ell_2}\rangle \ldots$. It follows from definition of $[\![\mathcal{P}]\!]^{TInv}$ that

$$(\nu_{\ell_j}, \nu_{\ell_{j+1}}) \in [\![P_i]\!]^{TInv}(q,q) = [\![\mathcal{T}_i]\!]^{TInv}(q,q)$$

for all $j \geq 1$. We next rename valuations in ρ_2 to obtain $\rho_2 = \langle q_{0,i}, \mu_0\rangle\langle q, \mu_1\rangle\langle q, \mu_2\rangle \ldots$.

Let us assume that $[\![P]\!]^{TInv}(q,q) \Leftrightarrow \bigvee_{k=1}^{p} R_k$ for some $p \geq 1$. By definition of $[\![P]\!]^{TInv}$, it follows that for each $\ell > k \geq 1$, there exists $1 \leq j \leq p$ such that $(\mu_k, \mu_\ell) \in R_j$. Consequently, we can construct a function $f : \{(k,\ell) \mid \ell > k \geq 1\} \to \{R_1, \ldots, R_p\}$ such that $(\mu_k, \mu_\ell) \in f(k,\ell)$ for all $\ell > k \geq 1$.

Let $\sim$ be the kernel of f and thus, $(k,\ell) \sim (k',\ell')$ if and only if $f(k,\ell) = f(k',\ell')$. Clearly, $\sim$ is an equivalence relation with finite index, since the range of f is finite. Consequently, by Ramsey theorem [Ram30], there exists an infinite sequence of integers $1 \leq k_1 < k_2 < k_3 < \ldots$ and an equivalence class $[(m,n)]_\sim$ for some m, n such that $(k_i, k_{i+1}) \sim (m,n)$ for all $i \geq 1$. Thus, there exists $1 \leq j \leq p$ such that $f(k_i, k_{i+1}) = R_j$ for all $i \geq 1$. Consequently, $\mu_{k_1}\mu_{k_2} \ldots$ is an infinite run of R_j and hence, $\mu_{k_1} \in wnt(R_j)$. Since $(\mu_0, \mu_{k_1}) \in [\![P_i]\!]^*(q_{0,i}, q) = [\![\mathcal{A}_i]\!]^*(q_{0,i}, q)$, by definition of $[\![P_i]\!]^*$, if follows that

$$\nu_0 = \mu_0 \in \left([\![P_i]\!]^*(q_{0,i}, q)\right)^{-1}\left(wnt(R_j)\right) \subseteq \left([\![P_i]\!]^*(q_{0,i}, q)\right)^{-1}\left(\bigcup_{k=1}^{p} wnt(R_k)\right) \subseteq \mathcal{N}_i$$

Thus, $\mathcal{B}_i \subseteq \mathcal{N}_i$.

" $\Leftarrow$ " Clearly, if both $\mathcal{T}_i$ and $\mathcal{A}_i$ are precise, then $\bigcup_{k=1}^{p_q} wnt(R_{q,k})$ is the set of initial valuations of non-terminating runs that start in $q \in Q_i$, by definition of $[\![P_i]\!]^{TInv}$. Consequently, $\left(\mathcal{A}_i(q_{0,i}, q)\right)^{-1}\left(\bigcup_{k=1}^{p_q} wnt(R_{q,k})\right)$ is the set of valuations in $q_{0,i}$ from which a non-terminating run that loops infinitely at q exists. Consequently,

$$\mathcal{N}_i \stackrel{def}{=} \bigcup_{q \in Q}\left(\left(\mathcal{A}_i(q_{0,i}, q)\right)^{-1}\left(\bigcup_{k=1}^{p_q} wnt(R_{q,k})\right)\right)$$

is the set of valuations in $q_{0,i}$ from which a non-terminating run that loops in some $q \in Q_i$ and thus, $\mathcal{N}_i \subseteq \mathcal{B}_i$.

Next, observe that if $\mathcal{T}_i \supseteq [\![P_i]\!]^{TInv}$ and $\mathcal{A}_i \supseteq [\![P_i]\!]^*$, the " $\Rightarrow$ " direction of the above proof still holds. Consequently, the lemma holds. $\square$

7.2.2 Computing Termination Sets of Non-recursive Programs

In the rest of this section, we aim to compute over-approximations $\mathcal{N}_i$ of the weakest non-termination sets $[\![P_i]\!]^{nt}_{\exists}$, formally $\mathcal{N}_i \supseteq [\![P_i]\!]^{nt}_{\exists}$, and then use the complement of $\mathcal{N}_m$ as an under-approximation of the weakest termination set $[\![P_m]\!]^t$, formally $\neg\, \mathcal{N}_m \subseteq [\![P_m]\!]^t$. Recall from Section 7.1.2 that for each valuation $\nu \in \mathcal{V}_i$, $\nu \in [\![P_i]\!]^{nt}_{\mathcal{N}}$ if and only if either

1. there exists an infinite run in P_i under $[\![\mathcal{P}]\!]$ that starts in $\langle q_{0,i}, \nu\rangle$, or
2. there exists a finite run of length $k \geq 0$ in P_i under $[\![\mathcal{P}]\!]$ that starts in $\langle q_{0,i}, \nu\rangle$, ends in $\langle q', \nu'\rangle$ for some $q' \in Q_i, \nu' \in \mathcal{V}_i$, and moreover, there exists a transition $q' \xrightarrow{\overrightarrow{\mathbf{z}}=call_j(\overrightarrow{\mathbf{t}})} q'' \in \Delta_i$ and a valuation $\nu'' \in \mathcal{N}_j$ such that

$$\textsc{MatchCall}(q' \xrightarrow{\overrightarrow{\mathbf{z}}=call_j(\overrightarrow{\mathbf{t}})} q'', \nu', \nu'') \Leftrightarrow \top.$$

Note that if an over-approximation of $[\![\mathcal{P}]\!]$ is used instead of $[\![\mathcal{P}]\!]$ in the above definition, one defines an over-approximation of $[\![P_i]\!]^{nt}_{\mathcal{N}}$.

To address the point 1 for a given procedure P_i, we first compute (an over-approximation of) a transition invariant of P_i. To this end, we adapt the procedure summary algorithm from Section 7.1. Intuitively, a transition invariant $\mathcal{T}$ can be computed by

Algorithm 9 Over-approximating the Weakest Non-termination Set of a Procedure

input a procedure $P_i = \langle \mathbf{x}_i, \overrightarrow{\mathbf{x}}^{in}_i, \overrightarrow{\mathbf{x}}^{out}_i, Q_i, q_{0,i}, q_{f,i}, q_{e,i}, \Delta_i\rangle$, a transition invariant $\mathcal{T} = (\mathcal{T}_1, \ldots, \mathcal{T}_n)$ of $\mathcal{P}$ and an over-approximation $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ of $[\![\mathcal{P}]\!]^*$
output An over-approximation of $[\![P]\!]^{nt}_{\exists}$

1: **function** ProcedureNT($P_i = \langle \mathbf{x}_i, \overrightarrow{\mathbf{x}}^{in}_i, \overrightarrow{\mathbf{x}}^{out}_i, Q_i, q_{0,i}, q_{f,i}, q_{e,i}, \Delta_i\rangle$, $\mathcal{T}$, $\mathcal{A}$)
2: $W \leftarrow \emptyset$
3: **for each** $q \in Q_i$ **do**
4: **let** $\mathcal{T}_i(q, q) \Leftrightarrow \big(R_1 \vee \cdots \vee R_p\big)$ for some $R_1, \ldots, R_p$, $p \geq 1$
5: $V \leftarrow \bigvee_{j=1}^{p} \mathrm{WNT}(R_j)$
6: **for each** $q \xrightarrow{\overrightarrow{\mathbf{z}}'=call_j(\overrightarrow{\mathbf{t}})} q' \in \Delta_i$ **do**
7: $V \leftarrow V \vee PlugWNT(i, j, \overrightarrow{\mathbf{t}})$
8: $Reach \leftarrow \mathcal{A}_i(q_{0,i}, q)$
9: $W \leftarrow W \vee Reach^{-1}(V)$
10: **return** ProjectNT(W)

1: **function** PlugNT($i, j, \overrightarrow{\mathbf{t}}$)
2: **return** $\mathcal{N}_j\big[(\overrightarrow{\mathbf{t}})_k/(\overrightarrow{\mathbf{x}}^{in}_i)_k\big]_{k=1..|\overrightarrow{\mathbf{t}}|}$

1: **function** ProjectNT(R)
2: **return** $\exists(\mathbf{x}_i \setminus \overrightarrow{\mathbf{x}}^{in}_i)\ .\ R$

marking all control states as both initial and final. Then, we can apply Lemma 7.2 to compute the set of valuations for which the point 1 holds. Algorithm 9 achieves this by executing lines 4–5 and 8–9 for each control state $q \in Q$. Line 5 computes the set of valuation in a control state $q \in Q_i$ from which a run that loops infinitely at q exists and line 9 next propagates this set to the initial control state. Note that we can apply Algorithm 4 from Section 6.2.1 to compute the weakest non-termination set $\mathrm{WNT}(R_j)$. Finally, Algorithm 9 eliminates variables not included in the signature of P_i and returns (line 10).

To address the point 2 above for a procedure P_i, over-approximations of the weakest non-termination sets must be computed first for the called procedures. Since the programs we consider are not recursive, it follows from the discussion in Section 2.2.2 that we can apply an approach similar to the one in Section 7.1.2 and compute over-approximations $\mathcal{N}_i$ of $[\![P_i]\!]^{nt}_{\exists}$ following a topological ordering with respect to $CG(\mathcal{P})$. Thus, $\mathcal{N}_i$ is computed first for procedures that do not call other procedures and it is then propagated to the calling sites, as the point 2 above requires. Algorithm 9 achieves this propagation by executing lines 6-9 for each control state $q \in Q$. Line 7 propagates over-approximations of non-termination sets from the called procedure to the calling site and line 9 next propagates this set further to the initial control state.

Algorithm 10 computes an under-approximation of $[\![P_m]\!]^{t}_{\exists}$. It first calls the computation of over-approximations of $[\![P_i]\!]^{nt}_{\exists}$ for procedures P_i in the given topological order with respect to $CG(\mathcal{P})$(lines 3-4. Finally, it computes a termination precondition by negating the computed over-approximation of $[\![P_m]\!]^{nt}$ (line 5).

Algorithm 10 Termination Precondition for a Program

input input a program $\mathcal{P} = \langle \mathbf{x}_g, \{P_{i_1}, \ldots, P_{i_n}\}, P_m \rangle$ ordered w.r.t. $CG(\mathcal{P})$
output An under-approximation of the weakest termination set $[\![P_m]\!]^{t}_{\exists}$ of P_m

1: **function** TerminationPrecondition($\mathcal{P} = \langle \mathbf{x}_g, \{P_{i_1}, \ldots, P_{i_n}\}, P_m \rangle$)
2: $(\mathcal{N}_1, \ldots, \mathcal{N}_n) \leftarrow (\emptyset, \ldots, \emptyset)$
3: **for** $k = 1, \ldots, n$ **do**
4: $\mathcal{N}_{i_k} \leftarrow$ ProcedureNT(P_{i_k})
5: **return** $\neg\, \mathcal{N}_m$

7.2.3 Flat Integer Programs

In this section, we study *flat integer programs* and show that their weakest termination set is Presburger definable. This immediately entails decidability of the termination problem, by decidability of Presburger arithmetic.

A *flat integer program* is a single-procedure non-recursive program where

1. each control state belongs to at most one cycle in the control flow graph (CFG)
2. for each cycle $q_1 \xrightarrow{R_1} q_2 \xrightarrow{R_2} \ldots \xrightarrow{R_n} q_1$ in the CFG, the relation $(R_1 \circ \ldots \circ R_n)$ is either a difference bounds, octagonal, or finite monoid affine relation

Let P be a flat integer program. It is known that the reachability problem for flat integer programs is decidable [CJ98, LS05]. Consequently, $[\![P]\!]^*$ can be computed by Algorithm 6. Consider Algorithm 11 that uses an auxiliary procedure LoopLabel(P, q) which returns the composition of relations that label the unique cycle on which the control state q appears or empty relation if there is no such cycle. Let $q \in Q$ and let $L_q = \text{LoopLabel}(P, q)$. Then,

$$\text{WNT}(L_q) = \text{WNT}(L_q^+) = \text{WNT}([\![P]\!]^{TInv}(q, q)).$$

First equality obviously holds for arbitrary relation. The second equality holds for flat programs, since each control states belongs to at most one cycle in the control flow graph. Moreover, L_q is a periodic relation and therefore, the weakest non-termination set $\text{WNT}(L_q)$ is Presburger definable, as proved in Chapter 6. Thus, given a flat integer program P, line 4 in Algorithm 11 is equivalent to line 5 of Algorithm 9. Consequently, since flat programs have no call transitions, Algorithm 11 correctly returns a Presburger formula defining $[\![P]\!]^{nt}$. The following theorem summarizes these observations.

Theorem 7.3 *The weakest (non-)termination set of flat integer programs is effectively computable and Presburger definable. Consequently, the termination problem is decidable for flat integer programs.*

Algorithm 11 Weakest Non-termination Set for Flat Integer Programs

input a flat integer program $P = \langle \mathbf{x}, Q, q_0, q_f, q_e, \Delta \rangle$ and the program summary $[\![P]\!]^*$
output $[\![P]\!]^{nt}$, the weakest non-termination set of P

1: **function** ProgramSummary($P = \langle \mathbf{x}, Q, q_{init}, q_{final}, q_{err}, \Delta \rangle$, $[\![P]\!]^*$)
2: $V \leftarrow \emptyset$
3: **for each** $q \in Q$ **do**
4: $A \leftarrow \text{WNT}(\text{LoopLabel}(P, q))$
5: $B \leftarrow [\![P]\!]^*(q_i, q)$
6: $V \leftarrow V \cup B^{-1}(A)$
7: **return** V

7.3 Predicate Abstraction with Acceleration

This section presents Counterexample-Guided *Accelerated* Abstraction Refinement (CEGAAR), a new reachability analysis algorithm. CEGAAR combines *interpolation-based predicate discovery* in counterexample-guided predicate abstraction with *acceleration* technique for computing the transitive closure of loops. CEGAAR applies acceleration to dynamically discovered looping patterns in the unfolding of the transition system and combines overapproximation with underapproximation. It constructs inductive invariants that rule out an infinite family of spurious counterexamples, alleviating the problem of divergence in predicate abstraction without loosing its adaptive nature. We present

theoretical and experimental justification for the effectiveness of CEGAAR, showing that inductive interpolants can be computed from classical Craig interpolants and transitive closures of loops. Throughout this section, we assume a single-procedure integer program without call transitions $P = \langle \mathbf{x}, Q, q_{init}, q_{err}, \Delta \rangle$ which can be obtained from a non-recursive procedural program by procedure inlining.

Related Work. Predicate abstraction has proved to be a rich and fruitful direction in automated verification of detailed properties of infinite-state systems [GS97, HJMM04]. The pioneering work in [BL99] is, to the best of our knowledge, the first to propose a solution to the divergence problem in predicate abstraction. More recently, sufficient conditions to enforce convergence of refinement in predicate abstraction are given in [BPR02], but it remains difficult to enforce them in practice. A promising direction for ensuring completeness with respect to a language of invariants is parameterizing the syntactic complexity of predicates discovered by an interpolating *split prover* [JM06]. Because it has the flavor of invariant enumeration, the feasibility of this approach in practice remains to be further understood.

To alleviate relatively weak guarantees of refinement in predicate abstraction in practice, researchers introduced *path invariants* [BHMR07] that rule out a family of counterexamples at once using constraint-based analysis. Our CEGAAR approach is similar in the spirit, but uses acceleration [BIK10, FL02, Boi99] instead of constraint-based analysis, and therefore has complementary strengths. Acceleration naturally generates precise *disjunctive invariants*, needed in many practical examples, while constraint-based invariant generation [BHMR07] resorts to an ad-hoc unfolding of the path program to generate disjunctive invariants. Acceleration can also infer expressive predicates, in particular modulo constraints, which are relevant for purposes such as proving memory address alignment.

The idea of generalizing spurious error traces was introduced also in [HHP09] by extending an infeasible trace, labeled with interpolants, into a finite interpolant automaton. The method of [HHP09] exploits the fact that some interpolants obtained from the infeasibility proof happen to be inductive w.r.t. loops in the program. In our case, given a spurious trace that iterates through a program loop, we *compute* the needed inductive interpolants, combining interpolation with acceleration. The method that is probably closest to CEGAAR is proposed in [CFLZ08]. In that work, the authors define *inductive interpolants* and prove the existence of effectively computable inductive interpolants for a class of affine loops, called *poly-bounded*. The approach is, however, limited to programs with one poly-bounded affine loop, for which initial and error states are specified. We only consider loops that are more restricted than the poly-bounded ones, namely loops for which transitive closures are Presburger definable. On the other hand, our method is more general in that it does not restrict the number of loops occurring in the path program and benefits from regarding both interpolation and transitive closure computation as black boxes. The ability to compute closed forms of certain loops is also exploited in algebraic approaches [BHHK10]. These approaches can also naturally be generalized to perform useful over-approximation [AAGP11] and under-approximation.

Roadmap. In Section 7.3.1, we define the notion of an *abstract reachability tree (ART)* that represents an abstraction of an integer program and define the notion of a *spurious* trace in an ART. Next, in Section 7.3.2, we overview the technique of interpolation-based refinement which rules out a spurious trace and define the notion of a *trace scheme*, a generalization of a trace, and a notion of inductive interpolant that rules out a potentially infinite number of traces. Section 7.3.3 describes how to compute inductive interpolants, and Section 7.3.4 presents how this computation is incorporated in the CEGAAR algorithm.

7.3.1 Abstract Reachability Tree

Let $P = \langle \mathbf{x}, Q, q_{init}, q_{err}, \Delta \rangle$ be a single-procedure integer program without call transitions. A *path* in P is a sequence $\theta : q_1 \xrightarrow{R_1} q_2 \xrightarrow{R_2} q_3 \ldots q_{n-1} \xrightarrow{R_{n-1}} q_n$ where $q_1, q_2, \ldots, q_n \in Q$ and $q_i \xrightarrow{R_i} q_{i+1}$ is an edge in Δ for each $i = 1, \ldots, n-1$. We denote the relation $R_1 \circ R_2 \circ \ldots \circ R_{n-1}$ by $\rho(\theta)$ and assume that the set of free variables of $\rho(\theta)$ is $\mathbf{x} \cup \mathbf{x}'$. The path θ is said to be a *cycle* if $q_1 = q_n$, and a *trace* if $q_1 = q_{init}$. The path θ is said to be *feasible* if and only if there exist valuations $\nu_1, \ldots, \nu_n : \mathbf{x} \to \mathbb{Z}$ such that $\nu_i, \nu_{i+1} \models R_i$ for all $i = 1, \ldots, n-1$.

Given a set of predicates $\mathcal{P}$, an abstract reachability tree represents an abstraction of an integer program with respect to $\mathcal{P}$.

Definition 7.4 *Given a program $P = \langle \mathbf{x}, Q, q_{init}, q_{err}, \Delta \rangle$, and a (possibly infinite) set of predicates $\mathcal{P}$ such that $\top, \bot \in \mathcal{P}$, an* abstract reachability tree *(ART) for P is a tuple $T = \langle S, \pi, r, e \rangle$ where $S \subseteq Q \times 2^{\mathcal{P} \setminus \{\bot\}}$ is a set of nodes (notice that for no node $\langle q, \Phi \rangle$ in T we may have $\bot \in \Phi$), $\pi : Q \to 2^{\mathcal{P}}$ is a mapping associating control states with sets of predicates, $i = q_{init} \times \{\top\}$ is the root node, $e \subseteq S \times S$ is a tree-structured edge relation:*

- *all nodes in S are reachable from the root r,*
- *for all $n, m, p \in S$, $e(n, p) \wedge e(m, p) \Rightarrow n = m$,*
- $e(\langle q_1, \Phi_1 \rangle, \langle q_2, \Phi_2 \rangle) \;\Rightarrow\; q_1 \xrightarrow{R} q_2$ *and* $\Phi_2 = \{P \in \pi(q_2) \mid post(\bigwedge \Phi_1, R) \to P\}$.

We say that an ART node $\langle q_1, \Phi_1 \rangle$ is *subsumed* by another node $\langle q_2, \Phi_2 \rangle$ if and only if $q_1 = q_2$ and $\bigwedge \Phi_1 \to \bigwedge \Phi_2$. It is usually considered that no node in an ART is subsumed by another node, from the same ART.

It can be easily checked that each path $\sigma : r = \langle q_1, \Phi_1 \rangle, \langle q_2, \Phi_2 \rangle, \ldots, \langle q_k, \Phi_k \rangle$, starting from the root in T, can be mapped into a trace $\theta : q_1 \xrightarrow{R_1} q_2 \ldots q_{k-1} \xrightarrow{R_{k-1}} q_k$ of P such that $post(\top, \rho(\theta)) \to \bigwedge \Phi_k$. We say that θ is a *concretization* of σ or that σ concretizes to θ. A path in an ART is said to be *spurious* if none of its concretizations is feasible.

7.3.2 Interpolation-Based Abstraction Refinement

By *refinement*, we understand the process of enriching the predicate mapping π of an ART $T = \langle S, \pi, r, e\rangle$ with new predicates. The goal of refinement is to prevent spurious counterexamples (paths to an error state) from appearing in the ART. To this end, an effective technique used in many predicate abstraction tools is that of *interpolation*.

Given an unsatisfiable conjunction $A \wedge B$, an interpolant I is a formula using the common variables of A and B such that $A \rightarrow I$ is valid and $I \wedge B$ is unsatisfiable. Intuitively, I is the explanation behind the unsatisfiability of $A \wedge B$. Below, we introduce a slightly more general definition of a *trace interpolant*.

Definition 7.5 ([JM06]) *Let $P = \langle \mathbf{x}, Q, q_{init}, q_{err}, \Delta\rangle$ be a program and*

$$\theta : q_1 \xrightarrow{R_1} q_2 \xrightarrow{R_2} q_3 \ldots q_{n-1} \xrightarrow{R_{n-1}} q_n$$

be an infeasible trace of P. An interpolant *for θ is a sequence of predicates $\langle I_1, I_2, \ldots, I_n\rangle$ with free variables in $\mathbf{x}$ such that: $I_1 = \top$, $I_n = \bot$, and for all $i = 1, \ldots, n-1$, $post(I_i, R_i) \rightarrow I_{i+1}$.*

Interpolants exist for many theories, including all theories with quantifier elimination, and thus for Presburger arithmetic. Moreover, a trace is infeasible if and only if it has an interpolant [EKS08]. Including any interpolant of an infeasible trace into the predicate mapping of an ART suffices to eliminate any abstraction of the trace from the ART. We can thus refine the ART and exclude an infeasible trace by including the interpolant that proves the infeasibility of the trace, as proved by the following lemma.

Lemma 7.6 *Let $P = \langle \mathbf{x}, Q, q_{init}, q_{err}, \Delta\rangle$ be a program and $\theta : q_1 \xrightarrow{R_1} q_2 \ldots q_{n-1} \xrightarrow{R_{n-1}} q_n$ be an infeasible trace of P. If $T = \langle S, \pi, r, e\rangle$ is an ART for G such that there exists an interpolant $\langle I_i \in \pi(q_i)\rangle_{i=1}^{n}$ for θ, then no path in T concretizes to θ.*

Proof: By contradiction, suppose that there exists a path

$$\sigma \ : \ \langle q_1, \Phi_1\rangle, \langle q_2, \Phi_2\rangle, \ldots, \langle q_n, \Phi_n\rangle$$

in T that concretizes to θ. We show by induction on i that $I_i \in \Phi_i$ for all $i = 1, \ldots, n$. By the definition of T, $I_1 = \top \in \Phi_1$. For the induction step, assume that $I_{i-1} \in \Phi_{i-1}$, for some $i > 1$. By the definition of T, we have $\Phi_i = \{P \in \pi(q_i) \mid post(\bigwedge \Phi_{i-1}, R_i) \rightarrow P\}$. Since $post(I_{i-1}, R_i) \rightarrow I_i$, by Definition 7.5 and $I_{i-1} \in \Phi_{i-1}$, we have $\bigwedge \Phi_{i-1} \rightarrow I_{i-1}$, and by monotonicity of the *post* operator, $post(\bigwedge \Phi_{i-1}, R_i) \rightarrow I_i$. But $I_i \in \pi(q_i)$ which implies $I_i \in \Phi_i$, by the definition of T. Consequently, $I_n = \bot \in \Phi_n$, which is in contradiction with the fact that no node in T may contain $\bot$ in its second component. □

Note that the refinement technique using Definition 7.5 only guarantees that *one* spurious counterexample is eliminated from the ART with each refinement step. This

fact hinders the efficiency of predicate abstraction tools, which must rely on the ability of theorem provers to produce interpolants that are general enough to eliminate more than one spurious counterexample at the time. The following definition generalizes the notion of a trace.

Definition 7.7 ([CFLZ08], Def. 2.4) *Given a program P, a* trace scheme *in P is a sequence of the form:*

$$\xi : q_0 \xrightarrow{Q_1} \overset{L_1}{\curvearrowright} q_1 \xrightarrow{Q_2} \ldots \xrightarrow{Q_{n-1}} \overset{L_{n-1}}{\curvearrowright} q_{n-1} \xrightarrow{Q_n} \overset{L_n}{\curvearrowright} q_n \xrightarrow{Q_{n+1}} q_{n+1} \tag{7.1}$$

where $q_0 = q_{init}$ and:

- *$Q_i = \rho(\theta_i)$ for some paths θ_i of P, from q_{i-1} to q_i,*
- *$L_i = \bigvee_{j=1}^{k_i} \rho(\lambda_{ij})$ for some cycles λ_{ij} of P, from q_i to q_i.*

Intuitively, a trace scheme represents an infinite regular set of traces in P. The trace scheme is said to be *feasible* if and only if at least one trace of P of the form

$$\theta_1; \lambda_{1i_1} \ldots \lambda_{1i_{j_1}}; \theta_2; \ldots; \theta_n; \lambda_{ni_n} \ldots \lambda_{ni_{j_n}}; \theta_{n+1}$$

is feasible.

The trace scheme is said to be *bounded* if $k_i = 1$, for all $i = 1, 2, \ldots, n$. A bounded[1] trace scheme is a regular language of traces, of the form $\theta_1 \cdot \lambda_1^* \cdot \ldots \cdot \theta_n \cdot \lambda_n^* \cdot \theta_{n+1}$, where θ_i are paths, and λ_i are cycles of P. The following is a stronger notion of an interpolant, which ensures generality with respect to an infinite family of counterexamples.

Definition 7.8 ([CFLZ08], Def. 2.5) *Let $P = \langle \mathbf{x}, Q, q_{init}, q_{err}, \Delta \rangle$ be a program and ξ be an infeasible trace scheme of the form (7.1). An* interpolant *for ξ is a sequence of predicates $\langle I_0, I_1, I_2, \ldots, I_n, I_{n+1} \rangle$, with free variables in $\mathbf{x}$, such that:*

1. *$I_0 = \top$ and $I_{n+1} = \bot$,*
2. *$post(I_i, Q_{i+1}) \rightarrow I_{i+1}$, for all $i = 0, 1, \ldots, n$,*
3. *$post(I_i, L_i) \rightarrow I_i$, for all $i = 1, 2, \ldots, n$.*

The main difference with Definition 7.5 is the third requirement, namely that each interpolant predicate (except for the first and the last one) must be *inductive* with respect to the corresponding loop relation. It is easy to see that each of the two sequences

$$\langle \top,\ post(\top, Q_1 \circ L_1^*),\ \ldots,\ post(\top, Q_1 \circ L_1^* \circ Q_2 \circ \ldots Q_n \circ L_n^*) \rangle \tag{7.2}$$

$$\langle wpre(\bot, Q_1 \circ L_1^* \circ Q_2 \circ \ldots Q_n \circ L_n^*),\ \ldots,\ wpre(\bot, Q_n \circ L_n^*),\ \bot \rangle \tag{7.3}$$

are interpolants for ξ provided that ξ is infeasible (Lemma 2.6 in [CFLZ08]). Just as for finite trace interpolants, the existence of an inductive interpolant suffices to prove the infeasibility of the entire trace scheme.

[1] This term is used in analogy with the notion of bounded languages [GS64].

Lemma 7.9 *Let $P = \langle \mathbf{x}, Q, q_{init}, q_{err}, \Delta \rangle$ be a program and ξ be an infeasible trace scheme of P of the form (7.1). If $T = \langle S, \pi, r, e \rangle$ is an ART for P such that there exists an interpolant $\langle I_i \in \pi(q_i) \rangle_{i=0}^{n+1}$ for ξ, then no path in T concretizes to a trace in ξ.*

Proof: By contradiction, suppose that there exists a path σ:

$$\langle q_0, \Phi_0 \rangle,\ \langle q_{11}, \Phi_{11} \rangle, \ldots, \langle q_{1i_1}, \Phi_{1i_1} \rangle,\ \ldots,\ \langle q_{n1}, \Phi_{n1} \rangle, \ldots, \langle q_{ni_n}, \Phi_{ni_n} \rangle,\ \langle q_{n+1}, \Phi_{n+1} \rangle \rangle$$

in T which concretizes to a trace in ξ. In analogy with the proof of Lemma 7.6, one shows that:

- $I_0 \in \Phi_0$,
- $I_k \in \Phi_{kj}$, for all $k = 1, \ldots, n$ and $j = 1, \ldots, i_k$,
- $I_{n+1} \in \Phi_{n+1}$.

The third condition of Definition 7.8 is needed for the proof of the second point above. Since $I_{n+1} = \bot$, this contradicts the fact that no node in T may contain $\bot$ in its second component. □

7.3.3 Computing Accelerated Interpolants

This section describes a method of refining an ART by excluding an infinite family of infeasible traces at once. Our method combines interpolation with acceleration in a way which is oblivious of the particular method used to compute interpolants. For instance, it is possible to combine proof-based [McM05] or constraint-based [RSS07] interpolation with acceleration, whenever computing the precise transitive closure of a loop is possible. In cases when the precise computation fails, we may resort to both over- and under-approximation of the transitive closure. In both cases, the accelerated interpolants are at least as general (and many times more general) than the classical interpolants extracted from a finite counterexample trace.

Given an infeasible error trace θ, we *fold* it into a trace scheme ξ. In this section, we describe approaches to abstraction refinement based on type of the trace scheme. The distinguishing factor is whether ξ is bounded and whether cycles in ξ can be accelerated. In Section 7.3.4, we show how these refinement techniques are integrated in the CEGAAR algorithm.

Precise Acceleration of Bounded Trace Schemes

We consider first the case of bounded trace schemes of the form (7.1) where the control states $q_1, \ldots, q_n$ belong to some cycles labeled with relations $L_1, \ldots, L_n$ such that L_i^* is Presburger definable. As discussed in Chapter 3, this requires that each L_i is either a difference bounds, octagonal, or finite monoid affine relation. Then, one can use acceleration algorithms from Chapter 3 or from [Boi99, FL02]. Under this restriction,

any infeasible bounded trace scheme has an effectively computable interpolant of one of the forms (7.2), or (7.3).

However, there are two problems with applying definitions (7.2) and (7.3) in order to obtain interpolants of trace schemes. On one hand, relational composition typically requires expensive quantifier eliminations. The standard proof-based interpolation techniques (e.g. [McM05]) overcome this problem by extracting the interpolants directly from the proof of infeasibility of the trace. Alternatively, constraint-based interpolation [RSS07] reduces the interpolant computation to a linear programming problem, which can be solved by efficient algorithms [Sch86]. Both methods apply, however, only to finite traces and not to infinite sets of traces defined as trace schemes. Another, more important, problem is related to the sizes of the interpolant predicates from (7.2), (7.3) compared to the sizes of interpolant predicates obtained by proof-theoretic methods (e.g. [KLR10]), as the following example shows.

Example. Let $R(x, y, x', y') \ : \ x' = x + 1 \wedge y' = y + 1$ and $\phi(x, y, \ldots)$, $\psi(x, y, \ldots)$ be some complex Presburger arithmetic formulae. The trace scheme

$$q_0 \xrightarrow{z=0 \wedge z'=z \wedge \phi} \underset{\curvearrowright}{q_1}^{z' = z+2 \wedge R} \xrightarrow{z=5 \wedge \psi} q_2 \tag{7.4}$$

is infeasible because z remains even, so it cannot become equal 5. One simple interpolant for this trace scheme has, at program point, q_1 the formula $z\%2 = 0$. On the other hand, the strongest interpolant has $(z = 0 \wedge z' = x \wedge \phi) \circ (z' = z + 2 \wedge R)^*$ at q_1, which is typically a much larger formula because of the complex formula ϕ. Note, however, that ϕ and R do not mention z, so they are irrelevant. □

To construct useful interpolants instead of the strongest or the weakest ones, we therefore proceed as follows. Let ξ be a bounded trace scheme of the form (7.1). For each control loop $q_i \xrightarrow{R_i} q_i$ of ξ, we define the corresponding *meta-transition* $q_i' \xrightarrow{R_i^*} q_i''$ labeled with the reflexive and transitive closure of R_i. Intuitively, firing the meta-transition has the same effect as iterating the loop an arbitrary number of times. We first replace each loop of ξ by the corresponding meta-transition. The result is the *meta-trace*:

$$\overline{\xi} : q_0 \xrightarrow{Q_1} q_1' \xrightarrow{L_1^*} q_1'' \xrightarrow{Q_2} q_2' \ \ldots \ q_{n-1}'' \xrightarrow{Q_n} q_n' \xrightarrow{L_n^*} q_n'' \xrightarrow{Q_{n+1}} q_{n+1} \tag{7.5}$$

Since we supposed that ξ is an infeasible trace scheme, the (equivalent) finite meta-trace $\overline{\xi}$ is infeasible as well, and it has an interpolant $\mathcal{I}_{\overline{\xi}} = \langle \top, I_1', I_1'', I_2', I_2'', \ldots, I_n', I_n'', \bot \rangle$ in the sense of Definition 7.5. This interpolant is not an interpolant of the trace scheme ξ in the sense of Definition 7.8. In particular, none of I_i', I_i'' is guaranteed to be inductive with respect to the loop relations L_i. To define compact inductive interpolants based on $\mathcal{I}_{\overline{\xi}}$ and the transitive closures L_i^*, we consider the following sequences:

$$\begin{aligned} \mathcal{I}_\xi^{post} &= \langle \top, post(I_1', L_1^*), post(I_2', L_2^*), \ldots, post(I_n', L_n^*), \bot \rangle \text{ and} \\ \mathcal{I}_\xi^{wpre} &= \langle \top, wpre(I_1'', L_1^*), wpre(I_2'', L_2^*), \ldots, wpre(I_n'', L_n^*), \bot \rangle. \end{aligned}$$

The following lemma proves the correctness of this approach.

Lemma 7.10 *Let $P = \langle \mathbf{x}, Q, q_{init}, q_{err}, \Delta \rangle$ be a program and ξ be an infeasible trace scheme of the form (7.1). Then $\mathcal{I}^{post}_{\xi}$ and $\mathcal{I}^{wpre}_{\xi}$ are interpolants for ξ, and moreover $\mathcal{I}^{wpre}_{\xi_i} \rightarrow \mathcal{I}^{post}_{\xi_i}$ for all $i = 1, 2, \ldots, n$.*

Proof: To prove that $\mathcal{I}^{post}_{\xi}$ is an interpolant for ξ, we show the three points of Definition 7.8. The first point holds by the construction of $\mathcal{I}^{post}_{\xi}$. For the second point, we have:

$$\begin{array}{rcll} post(I'_i, L^*_i) & \rightarrow & I''_i & \text{since } \mathcal{I}_{\overline{\xi}} \text{ is an interpolant for } \overline{\xi}, \\ post(post(I'_i, L^*_i), Q_{i+1}) & \rightarrow & post(I''_i, Q_{i+1}) & \text{since } post \text{ is monotone}, \\ post(\mathcal{I}^{post}_{\xi_i}, Q_{i+1}) & \rightarrow & I'_{i+1} & \text{since } \mathcal{I}_{\overline{\xi}} \text{ is an interpolant for } \overline{\xi}. \end{array}$$

We must show next that $I'_{i+1} \rightarrow \mathcal{I}^{post}_{\xi_{i+1}}$. For this, we compute:

$$\begin{array}{rcl} post(I'_{i+1}, L^*_{i+1}) & = & \exists \mathbf{z} \,.\, I'_{i+1}(\mathbf{z}) \wedge L^*_{i+1}(\mathbf{z}, \mathbf{x}) \\ & = & \exists \mathbf{z} \,.\, I'_{i+1}(\mathbf{z}) \wedge \bigvee_{k=0}^{\infty} L^k_{i+1}(\mathbf{z}, \mathbf{x}) \\ & = & \bigvee_{k=0}^{\infty} \exists \mathbf{z} \,.\, I'_{i+1}(\mathbf{z}) \wedge L^k_{i+1}(\mathbf{z}, \mathbf{x}) \\ & = & \exists \mathbf{z} \,.\, I'_{i+1}(\mathbf{z}) \wedge \epsilon \ \vee \ \bigvee_{k=1}^{\infty} \exists \mathbf{z} \,.\, I'_{i+1}(\mathbf{z}) \wedge L^k_{i+1}(\mathbf{z}, \mathbf{x}) \end{array}$$

We have that $\exists \mathbf{z} \,.\, I'_{i+1}(\mathbf{z}) \wedge \epsilon$ is equivalent to I'_{i+1}, which concludes the second point. For the third point, we compute:

$$\begin{array}{rcl} post(\mathcal{I}^{post}_{\xi_i}, L_i) & = & \exists \mathbf{z} \,.\, post(I'_i, L^*_i)(\mathbf{z}) \wedge L_i(\mathbf{z}, \mathbf{x}) \\ & = & \exists \mathbf{z} \exists \mathbf{t} \,.\, I'_i(\mathbf{t}) \wedge L^*_i(\mathbf{t}, \mathbf{z}) \wedge L_i(\mathbf{z}, \mathbf{x}) \\ & = & \exists \mathbf{t} \,.\, I'_i(\mathbf{t}) \wedge L^+_i(\mathbf{t}, \mathbf{x}) \\ & \rightarrow & \exists \mathbf{t} \,.\, I'_i(\mathbf{t}) \wedge L^*_i(\mathbf{t}, \mathbf{x}) \\ & = & post(I'_i, L^*_i) = \mathcal{I}^{post}_{\xi_i} \end{array}$$

The proof for the $\mathcal{I}^{wpre}_{\xi}$ interpolant is symmetric, using the fact that *post* and *wpre* form a Galois connection. Finally, we have $wpre(I''_i, L^*_i) \rightarrow I'_i \rightarrow post(I'_i, L^*_i)$, which proves the last statement. □

Notice that computing $\mathcal{I}^{post}_{\xi}$ and $\mathcal{I}^{wpre}_{\xi}$ requires n relational compositions, which is, in principle, just as expensive as computing directly one of the extremal interpolants (7.2), (7.3). However, by re-using the meta-trace interpolants, one potentially avoids the worst-case combinatorial explosion in the size of the formulae, which occurs when using (7.2), (7.3) directly.

Example. Let us consider again the trace scheme (7.4). The corresponding unfeasible finite trace $\overline{\xi}$ is:

$$q_0 \xrightarrow{z=0 \wedge z'=z \wedge \phi} q'_1 \xrightarrow{\exists k \geq 0 \,.\, z'=z+2k \ \wedge \ x'=x+k \ \wedge \ y'=y+k} q''_1 \xrightarrow{z=5 \wedge \psi} q_2$$

A possible interpolant for this trace is $\langle \top, z = 0, \exists k \geq 0 \,.\, z = 2k, \bot \rangle$. An inductive interpolant for the trace scheme, derived from it, is $\mathcal{I}^{post}_{\xi} = \langle \top, post(z = 0, \exists k \geq 0 . z' = z + 2k \wedge x' = x + k \wedge y' = y + k), \bot \rangle = \langle \top, \ z\%2 = 0, \ \bot \rangle$. □

Bounded Overapproximations of Trace Schemes

Consider a trace scheme (7.1), not necessarily bounded, where the transitive closures of the relations L_i labeling the loops are not computable by any available acceleration method from Chapter 3 or from [Boi99, FL02]. One alternative is to find abstractions $L_i^\sharp$ of the loop relations, i.e. relations $L_i^\sharp \leftarrow L_i$, for which transitive closures are computable. If the new abstract trace remains infeasible, it is possible to compute an interpolant for it, which is an interpolant for the original trace scheme. However, replacing the relations L_i with their abstractions $L_i^\sharp$ may turn an infeasible trace scheme into a feasible one where the traces introduced by abstraction are spurious. In this case, we give up the overapproximation and turn to the underapproximation technique described in the next section.

The overapproximation method computes an interpolant for a trace scheme ξ of the form (7.1) under the assumption that the abstract trace scheme:

$$\xi^\sharp : q_0 \xrightarrow{Q_1} \overset{L_1^\sharp}{\underset{\frown}{q_1}} \xrightarrow{Q_2} \ldots \xrightarrow{Q_{n-1}} \overset{L_{n-1}^\sharp}{\underset{\frown}{q_{n-1}}} \xrightarrow{Q_n} \overset{L_n^\sharp}{\underset{\frown}{q_n}} \xrightarrow{Q_{n+1}} q_{n+1} \tag{7.6}$$

is infeasible. In this case, one can effectively compute the interpolants $\mathcal{I}^{post}_{\xi^\sharp}$ and $\mathcal{I}^{wpre}_{\xi^\sharp}$ since the transitive closures of the abstract relations labeling the loops are computable by acceleration. The following lemma proves that, under certain conditions, computing an interpolant for the abstraction of a trace scheme is sound.

Lemma 7.11 *Let P be a program and ξ be a trace scheme (7.1) such that the abstract trace scheme $\xi^\sharp$ (7.6) is infeasible. Then the interpolants $\mathcal{I}^{post}_{\xi^\sharp}$ and $\mathcal{I}^{wpre}_{\xi^\sharp}$ for $\xi^\sharp$ are also interpolants for ξ.*

Proof: We show that $\mathcal{I}^{post}_{\xi^\sharp}$ meets the three conditions of Definition 7.8. The first condition is trivially true, while the proof of the second condition is essentially the same as in the proof of Lemma 7.10. For the third point, since $L_i \rightarrow L_i^\sharp$, we have:

$$\begin{aligned} post(\mathcal{I}^{post}_{\xi^\sharp}, L_i) &= post(post(I_i', L_i^{\sharp *}), L_i) \\ &\rightarrow post(post(I_i', L_i^{\sharp *}), L_i^\sharp) \\ &= post(I_i', L_i^{\sharp +}) \rightarrow \mathcal{I}^{post}_{\xi^\sharp} \end{aligned}$$

The proof for $\mathcal{I}^{wpre}_{\xi^\sharp}$ is symmetrical. □

To compute abstractions of relations that are guaranteed to have Presburger-definable transitive closures, we can use octagonal relations (Definition 2.17) and compute the *integer octagonal hull* of a relation L. This is the strongest conjunction $L^\sharp = \bigwedge\{u \leq c \mid u \in U(\mathbf{x} \cup \mathbf{x}'),\ L \rightarrow u \leq c\}$ In practice, if for instance, L is a union of convex polyhedra, one can use integer linear programming [Sch86] to compute $L^\sharp$ efficiently.

Bounded Underapproximations of Trace Schemes

Let ξ be a trace scheme of the form (7.1), where each relation L_i labeling a loop is a disjunction $L_{i1} \vee \ldots \vee L_{ik_i}$ of relations for which the transitive closures are effectively computable and Presburger definable. A *bounded underapproximation scheme* of a trace scheme ξ is obtained by replacing each loop $q_i \xrightarrow{L_i} q_i$ in ξ by a bounded trace scheme $\xi^\flat$ of the form

$$\overset{L_{i1}}{\curvearrowright} \quad \overset{L_{i2}}{\curvearrowright} \qquad \overset{L_{ik_i}}{\curvearrowright}$$
$$q_i^1 \xrightarrow{\epsilon} q_i^2 \xrightarrow{\epsilon} \ldots \; q_i^{k_i}$$

where ϵ denotes the identity relation. Let us denote[2] the result of this replacement by $\xi^\flat$. It is manifest that the set of traces $\xi^\flat$ is included in ξ.

Since we assumed that the reflexive and transitive closures L_{ij}^* are effectively computable and Presburger definable, the feasibility of $\xi^\flat$ is a decidable problem. If $\xi^\flat$ is found to be feasible, this points to a real error trace in the system. On the other hand, if $\xi^\flat$ is found to be infeasible, let $\mathcal{I}_{\xi^\flat} = \langle \top, I_1^1, \ldots, I_1^{k_1}, \ldots, I_n^1, \ldots, I_n^{k_n}, \bot \rangle$ be an interpolant for $\xi^\flat$. A refinement scheme using this interpolant associates the predicates $\{I_i^1, \ldots, I_i^{k_i}\}$ with the control state q_i from the original program. As the following lemma shows, this guarantees that any trace that follows the pattern of $\xi^\flat$ is excluded from the ART, ensuring that a refinement of the ART using a suitable underapproximation (that includes a spurious counterexample) is guaranteed to make progress.

Lemma 7.12 *Let $P = \langle \mathbf{x}, Q, q_{init}, q_{err}, \Delta \rangle$ be a program, ξ be an infeasible trace scheme of P (7.1) and $\xi^\flat$ a bounded underapproximation of ξ. If $T = \langle S, \pi, r, e \rangle$ is an ART for P, such that $\{I_i^1, \ldots, I_i^{k_i}\} \subseteq \pi(q_i)$, then no path in T concretizes to a trace in $\xi^\flat$.*

Proof: By contradiction, suppose that there exists a path in T which concretizes to a trace in $\xi^\flat$, and let

$$\underbrace{\langle q_i, \Phi_{i1}^1 \rangle, \; \ldots, \; \langle q_i, \Phi_{i\ell_{i,1}}^1 \rangle}_{\overset{L_{i1}}{\curvearrowright} \; q_i}, \ldots, \underbrace{\langle q_i, \Phi_{i1}^{k_i} \rangle, \; \ldots, \; \langle q_i, \Phi_{i\ell_{i,k_i}}^{k_i} \rangle}_{\overset{L_{ik_i}}{\curvearrowright} \; q_i}$$

be the fragment of the path which corresponds to the unfolding of the sub-trace

$$\overset{L_{i1}}{\curvearrowright} \quad \overset{L_{i2}}{\curvearrowright} \qquad \overset{L_{ik_i}}{\curvearrowright}$$
$$q_i \xrightarrow{\epsilon} q_i \xrightarrow{\epsilon} \ldots \; q_i \; .$$

One can show, along the lines of the proof of Lemma 7.10, that $I_i^j \in \Phi_{i\ell}^j$ for all $j = 1, \ldots, k_i$ and $\ell = 1, \ldots, \ell_{i,j}$. In this way, we obtain that the last set Φ contains $\bot$, which contradicts the definition of the ART. $\square$

[2]The choice of the name depends on the ordering of particular paths $L_{i1}, L_{i2}, \ldots, L_{ik_i}$, however we shall denote any such choice in the same way, in order to keep the notation simple.

Notice that a refinement scheme based on underapproximation guarantees the exclusion of those traces from the chosen underapproximation trace scheme, and not of all traces from the original trace scheme. Since a trace scheme is typically obtained from a finite counterexample, an underapproximation-based refinement still guarantees that the particular counterexample is excluded from further searches. In other words, using underapproximation is still better than the classical refinement method since it can potentially exclude an entire family of counterexamples (including the one generating the underapproximation) at once.

7.3.4 Counterexample-Guided Accelerated Abstraction Refinement

Finally, we describe the core of the Counterexample-Guided Accelerated Abstraction Refinement (CEGAAR) algorithm. The method starts by initializing the set of predicates with $\mathcal{P} = \{\top, \bot\}$. Then, it iteratively applies the following steps. The ART is constructed on-the-fly, according to a certain exploration strategy. When an abstract error state is reached, the corresponding abstract trace θ that leads from the root of the ART to the abstract error state is folded into a trace scheme ξ. We distinguish several cases:

1. If ξ is bounded, we construct a corresponding meta-trace $\overline{\xi}$. If $\overline{\xi}$ is feasible, a real error has been found and the algorithm terminates. Otherwise, inductive interpolants are computed using Lemma 7.10 and refine the abstraction using Lemma 7.6. Then, CEGAAR continues with a next iteration.

2. If ξ is not bounded, we compute a bounded abstract trace scheme $\xi^\sharp$. If the meta-trace corresponding to $\xi^\sharp$ is not feasible, $\xi^\sharp$ is spurious by Lemma 7.11. Then, inductive interpolants are computed by applying Lemma 7.10. These interpolants are used to refine the abstraction using Lemma 7.6, and CEGAAR continues with a next iteration.

3. Else, if the overapproximation $\xi^\sharp$ from Case 2 was found to be feasible, it could be the case that the abstraction of the scheme introduced a spurious error trace. In this case, we compute a bounded underapproximation $\xi^\flat$ of the trace scheme ξ and apply the same reasoning as in Case 1.

8 Experiments

In this chapter, we report on experiments we have performed in order to evaluate the methods presented in the preceding chapters. First, in Section 8.1, we report on experiments that evaluate the transitive closure algorithm presented in Chapter 3 and Chapter 4 and show that in some cases, we can achieve a speed-up of four orders of magnitude when compared with older algorithms for difference bounds and octagonal relations.

In Section 8.2, we evaluate the reachability analysis techniques from Chapter 7 on a number of benchmarks we collected. We observe that the reachability method from Section 7.1 computing precise procedure summaries outperforms earlier approaches based on acceleration. Further, we evaluate the method from Section 7.3 that combines interpolation-based predicate abstraction with acceleration and observe that the combined approach successfully verifies many models that can be handled by neither interpolation-based predicate abstraction nor the summary and acceleration-based method.

Finally, in Section 8.3, we evaluate the termination analysis techniques from Chapter 6 and Section 7.2 and shows that they can be applied to verify (non-)termination of several benchmarks.

8.1 Transitive Closure Computation

We have implemented Algorithm 1 for difference bounds and octagonal relations within the FLATA toolset [HIG+12]. We compared the performance of this algorithm with existing transitive closure computation methods for difference bounds [BIL09] and octagonal relations [BGI09].

Table 8.1 shows the results of the comparison between the older algorithms described in [BIL09, BGI09] (denoted as *old*) and Algorithm 1 for difference bounds relations $d_{1,\ldots,6}$ and octagonal relations $o_{1,\ldots,6}$. The tests have been performed on both *compact* (minimum number of constraints) and *canonical* (i.e. closed, for difference bounds and tightly closed, for octagons) relations. The *speedup* column gives the ratio between the *old* and *new* execution times. The experiments were performed on a 2.53GHz machine with 4GB of memory.

As shown in Table 8.1, the maximum observed speedup is almost 10^5 for difference bounds (d_4 in canonical form) and of the order of four for octagons. For the relations d_5 (canonical form), d_6 and o_6 the computation using older methods took longer than 10^6 msec. It is also worth noticing that the highest execution time with the new method was of 2.5 msec.

Table 8.2 compares FLATA with the FAST tool [BLP06] on counter systems with one self loop labeled with a randomly generated deterministic difference bounds relation. We

Table 8.1: A comparison with older algorithms on difference bounds and octagons. Times are in milliseconds.

Relation		new	compact		canonical	
			old	speedup	old	speedup
d_0	$(x - x' = -1) \wedge (x = y')$	0.18	0.70	3.90	38.77	215.39
d_1	$(x - x' = -1) \wedge (x' = y')$	0.18	18.18	101.00	38.77	215.39
d_2	$(x - x' = -1) \wedge (x = y') \wedge (x - z' \leq 5) \wedge (z = z')$	1.20	26.50	22.10	33431.20	27859.30
d_3	$(x - x' = -1) \wedge (x = y') \wedge (x - z \leq 5) \wedge (z = z')$	0.60	32.70	54.50	33505.50	55841.70
d_4	$(x - x' = -1) \wedge (x = y) \wedge (x - z \leq 5) \wedge (z = z')$	0.50	702.30	1404.60	48913.80	97827.60
d_5	$(a = c) \wedge (b = a') \wedge (b = b') \wedge (c = c')$	1.80	5556.60	3087.00	$> 10^6$	∞
d_6	$(a - b' \leq -1) \wedge (a - e' \leq -2) \wedge (b - a' \leq -2)$ $\wedge (b - c' \leq -1) \wedge (c - b' \leq -2) \wedge (c - d' \leq -1)$ $\wedge (d - c' \leq -2) \wedge (d - e' \leq -1 \wedge e - a' \leq -1)$ $\wedge (e - d' \leq -2) \wedge (a' - b \leq 4) \wedge (a' - c \leq 3)$ $\wedge (b' - c \leq 4 \wedge b' - d \leq 3) \wedge (c' - d \leq 4) \wedge (c' - e \leq 3)$ $\wedge (d' - a \leq 3 \wedge d' - e \leq 4) \wedge (e' - a \leq 4) \wedge (e' - b \leq 3)$	5.6	$> 10^6$	∞	$> 10^6$	∞
o_1	$(x + x' = 1)$	0.21	0.91	4.33	0.91	4.33
o_2	$(x + y' \leq -1) \wedge (-y - x' \leq -2)$	0.29	0.85	2.93	0.84	2.90
o_3	$(x \leq x') \wedge (x + y' \leq -1) \wedge (-y - x' \leq -2)$	0.32	0.93	2.91	0.94	2.94
o_4	$(x + y \leq 5) \wedge (-x + x' \leq -2) \wedge (-y + y' \leq -3)$	0.21	3.67	17.48	13.52	64.38
o_5	$(x + y \leq 1) \wedge (-x \leq 0) \wedge (-y \leq 0)$	1.20	20050.90	16709.10	$> 10^6$	∞
o_6	$(x \geq 0) \wedge (y \geq 0) \wedge (x' \geq 0) \wedge (y' \geq 0)$ $\wedge (x + y \leq 1) \wedge (x' + y' \leq 1) \wedge (x - 1 \leq x')$ $\wedge (x' \leq x + 1) \wedge (y - 1 \leq y') \wedge (y' \leq y + 1)$	2.5	$> 10^6$	∞	$> 10^6$	∞

Table 8.2: Comparison with FAST (MONA plugin) on deterministic difference bounds. Times are in seconds. E_T: timeout 30 s, E_B: BDD too large, E_M: out of memory.

vars	FLATA			FAST				
	done	av.	E_T	done	av.	E_T	E_M	E_B
10	50	1.5	0	49	0.6	0	0	1
15	50	1.6	0	31	10.5	17	0	2
20	50	1.6	0	4	3.4	9	8	29
25	50	1.6	0	2	4.2	2	10	36
50	50	1.6	0	0	-	0	0	50
100	49	7.7	1	0	-	0	0	50

(a) – matrix density 3%

vars	FLATA			FAST				
	done	av.	E_T	done	av.	E_T	E_M	E_B
10	50	1.5	0	22	6.9	23	1	4
15	50	1.5	0	1	20.6	4	3	42
20	50	1.6	0	0	-	1	0	49
25	43	1.7	7	0	-	0	0	50
50	50	2.3	0	0	-	0	0	50
100	42	5.5	8	0	-	0	0	50

(b) – matrix density 10%

generated 50 such relations for each size $N = 10, 15, 20, 25, 50, 100$. Notice that FAST usually runs out of memory for more than 25 variables, whereas FLATA can handle 100 variables in reasonable time (less than 8 seconds on average).

8.2 Reachability Analysis

We have implemented the reachability analysis based on acceleration and procedure summaries, described in Section 7.1, in the FLATA verifier [HIG+12]. We use algorithms that are specific to subclasses of integer relations (e.g. difference bounds or octagonal relations) for operations such as composition, satisfiability, and transitive closure. We resort to an external SMT solver YICES [DdM] only for checking satisfiability of polyhedra and modulo relations.

The CEGAAR algorithm, described in Section 7.3, was implemented by building on

(1) the FLATA verifier[1] [HIG+12] that computes accelerations and (2) on the predicate abstraction engine ELDARICA[2] [HIG+12] which uses the Princess interpolating theorem prover [BKRW10, Rüm08] to generate interpolants.

Table 8.3 compares the performance of the FLATA, ELDARICA, ELDARICA with *static acceleration*, and ELDARICA with *dynamic acceleration* (CEGAAR) on a number of benchmarks (the platform used for experiments is Intel® Core™2 Duo CPU P8700, 2.53GHz with 4GB of RAM). *Static acceleration* [CFLZ08] is a lightweight acceleration technique generalizing large block encoding (LBE) [BCG+09] with transitive closures. It simplifies the control flow graph prior to predicate abstraction.

The benchmarks are all in the Numerical Transition Systems format[3] (NTS). We have considered six sets of examples, extracted automatically from different sources: (a) C programs with arrays provided as examples of divergence in predicate abstraction [JM06], (b) verification conditions for programs with arrays, expressed in the SIL logic of [BHI+09] and translated to NTS, (c) small C programs with challenging loops, (d) NTS extracted from programs with singly-linked lists by the L2CA tool [BBH+06], (e) C programs with asynchronous procedure calls translated into NTS using the approach of [GM12] (the examples with extension .optim are obtained via an optimized translation method [Gan12]), and (f) models extracted from VHDL models of circuits following the method of [SV07]. Table 8.3 also reports on the size of NTS models, some of which have multiple procedures: $\|\mathbf{x}\|$, $\|Q\|$, and $\|T\|$ denote the total number of variables, the total number of control states, and the total number of transitions of all procedures of the respective model.

Moreover, we have made a comparison with several other reachability analysis tools based on different verification methodologies. The FAST verifier [BLP06] is based on acceleration of loops labeled with finite monoid affine relations. We have run FAST with several available plugins for solving Presburger queries: MONA [KM] (finite automata), Prestaf [Cou] (shared automata), and Omega [PRK+] (quantifier elimination). Table 8.3 reports on PresTaf which outperformed other plugins. The ARMC tool [PR07] uses predicate abstraction and interpolation-based abstraction refinement. The ASPIC tool [Gon] uses widening-based abstract interpretation.

Next, we briefly describe some of the benchmarks we considered and then comment on the results of our experiments.

Benchmarks

One of the set of models we considered—denoted (f) in Table 8.3—is taken from [SV07] where an approach for verification of generic VHDL circuit designs based on translation to counter automata is presented. Traditional verification techniques for hardware systems usually assume that the state space of these systems is finite. The approach presented in [SV07] aims at verification of parameterized VHDL components with infinite state space. The translation to counter automata described in [SV07] maps bit

[1] `http://www-verimag.imag.fr/FLATA.html`
[2] `http://lara.epfl.ch/w/eldarica`
[3] `http://richmodels.epfl.ch/ntscomp_ntslib`

Table 8.3: A comparison of reachability analysis tools. The letter after the model name distinguishes **C**orrect models from models with a reachable **E**rror state. Items with "-", "d", and "x" signify timeout of 300s, "don't know" answer, and an unsupported class of models, respectively.

Model	Size			Time [s]				Time [s]		
	$\|\mathbf{x}\|$	$\|Q\|$	$\|T\|$	FLATA	ELDARICA	ELDARICA-ACCEL STATIC	ELDARICA-ACCEL DYNAMIC (CEGAAR)	FAST	ARMC	ASPIC
(a) Examples from [JM06]										
anubhav (C)	29	20	25	0.8	3.0	4.0	3.1	49.2	2.6	0.2
copy1 (E)	39	21	24	2.0	7.2	5.8	5.9	14.5	44.0	d
cousot (C)	29	31	34	0.6	-	6.2	5.9	35.1	4.0	0.2
loop1 (E)	34	21	24	1.7	7.1	5.2	5.4	11.6	36.1	d
loop (E)	34	21	24	1.8	5.9	4.8	5.4	17.3	36.1	d
scan (E)	32	25	29	3.3	-	5.1	5.0	9.0	-	d
string_concat1 (E)	40	43	56	5.3	-	10.1	7.3	-	-	d
string_concat (E)	34	39	52	4.9	-	7.0	7.5	-	-	d
string_copy (E)	37	30	36	4.6	-	6.3	5.7	35.6	-	d
substring1 (E)	45	49	61	0.6	9.4	18.2	8.3	-	0.8	d
substring (E)	33	33	41	2.1	3.3	6.3	3.5	-	0.4	d
(b) Verification conditions for array programs [BHI+09]										
rotation_vc.1 (C)	11	13	55	0.6	2.0	9.5	2.0	x	0.6	x
rotation_vc.2 (C)	11	20	93	1.6	2.2	18.5	2.2	x	0.7	x
rotation_vc.1 (E)	11	13	56	1.1	1.3	10.2	1.3	x	0.3	x
split_vc.1 (C)	14	32	183	3.9	3.7	91.1	3.6	x	3.8	x
split_vc.2 (C)	14	29	146	3.0	2.3	74.1	2.2	x	1.1	x
split_vc.1 (E)	14	38	276	28.5	2.3	185.6	2.4	x	1.7	x
(c) Examples from [Mon12]										
boustrophedon (C)	25	22	27	-	-	-	14.4	-	-	d
gopan (C)	25	26	28	0.4	-	-	6.4	0.6	-	d
halbwachs (C)	29	32	38	-	-	7.3	7.0	-	-	d
rate_limiter (C)	35	25	27	31.7	6.1	8.1	5.5	x	8.1	x
(d) Examples from L2CA [BBH+06]										
bubblesort (E)	12	674	791	14.9	9.9	9.5	9.3	-	0.9	d
insdel (E)	7	28	31	0.1	1.3	2.5	1.4	1.2	0.1	d
insertsort (E)	13	130	169	2.0	4.2	5.0	4.0	-	0.3	d
listcounter (C)	4	31	35	0.3	-	1.9	3.7	14.2	2.3	0.1
listcounter (E)	6	31	34	0.3	1.4	1.6	1.4	-	0.1	d
listreversal (C)	7	97	107	4.5	3.0	6.0	3.3	-	47.9	0.1
listreversal (E)	10	99	107	0.8	2.7	8.1	2.8	-	0.3	d
mergesort (E)	11	544	606	1.2	7.7	21.3	7.4	-	0.7	d
selectionsort (E)	15	401	459	1.5	8.1	13.7	7.7	-	0.5	d
(e) Examples from [GM12]										
h1 (E)	28	40	50	-	5.1	5.6	5.1	x	17.7	x
h1.optim (E)	19	38	39	0.8	2.9	5.5	2.9	x	0.7	x
h1h2 (E)	29	41	52	-	9.4	10.1	12.2	x	57.0	x
h1h2.optim (E)	20	39	41	1.1	3.3	4.4	3.4	x	3.4	x
simple (E)	28	40	50	-	6.4	7.0	8.4	x	17.2	x
simple.optim (E)	19	38	39	0.8	3.0	5.1	2.9	x	0.7	x
test0 (C)	28	41	52	-	23.0	23.4	29.2	x	58.9	x
test0.optim (C)	19	39	40	0.3	3.2	5.4	3.2	x	4.3	x
test0 (E)	27	39	48	-	5.4	5.9	5.7	x	17.4	x
test0.optim (E)	19	37	38	0.6	3.0	5.8	2.9	x	0.6	x
test1.optim (C)	24	58	62	0.9	4.7	5.9	7.8	x	23.1	x
test1.optim (E)	24	56	60	1.5	4.4	5.9	4.7	x	10.8	x
test2_1.optim (E)	22	50	55	1.6	5.2	5.5	5.6	x	6.0	x
test2_2.optim (E)	22	51	56	2.9	4.6	5.9	4.6	x	5.9	x
test2.optim (C)	37	55	78	6.4	27.2	30.1	30.0	x	93.5	x
wrpc.manual (C)	5	9	13	0.6	1.2	1.4	1.2	x	47.1	x
wrpc (E)	54	60	89	-	7.9	8.4	8.2	x	0.3	x
wrpc.optim (E)	34	49	55	-	5.1	8.5	5.2	x	1.4	x
(f) VHDL models from [SV07]										
counter (C)	2	6	13	0.1	1.6	1.6	1.6	0.8	0.2	0.1
register (C)	2	10	49	0.2	1.1	1.1	1.1	0.5	0.2	0.1
synlifo (C)	3	43	1006	16.6	22.1	21.4	22.0	171.8	52.8	2.6

variables to control locations and integer variables to counters. Various safety properties are encoded as bit variables whose values are equivalent to propositional logic formulae representing the bad (unsafe) states. For instance, the SynLifo is a synchronous LIFO component with push and pop operations, which implements signals empty and full. The property checks if these signals are set correctly for a LIFO container of arbitrary size.

Another set of examples—denoted (b) in Table 8.3—are counter automata generated from programs with singly-linked lists, using the approach described in [BBH+06]. The main idea is that the set of heaps generated by a program with a finite number of local variables can be represented by a finite number of shape graphs, and the (unbounded) lengths of various list segments can be tracked by counters. The result of the translation of a program with lists is a counter automaton whose transition semantics is in bisimulation with the original program. For all singly-linked list programs, we check that there are no null pointer dereferences. For instance, the ListReversal is a textbook program that returns a list containing the same elements as the input list in the reversed order. The reversal is done in place by changing the links between the cells instead of creating a copy of the input list. Here, we also check that the lengths of the input list equals the length of the output list.

A next set of counter automata models—denoted (d) in Table 8.3—are obtained from the decision procedure of the array logic SIL (Singly Indexed Logic), described in [HIV08a]. The decidability of the satisfiability problem for SIL encodes the set of models of a formula as the union of sets of traces of a set of flat counter automata with difference bounds constraints, whose emptiness is known to be decidable, e.g., [CJ98, FL02]. Since Flata is guaranteed to terminate on flat models with periodic relations on loops, we can use it as a solver for the SIL logic. We report on two SIL formulae which arise as verification conditions for loop invariants of array manipulating programs. The *array rotation* program rotates an array by one element to the left, and the *array split* program splits an array to negative and non-negative parts.

The (f) benchmarks in Table 8.3 were generated from C programs with asynchronous procedure calls. For instance, wrpc is a simplified asynchronous implementation of windowed RPC, in which a client makes n asynchronous procedure calls in all, of which at most $w \leq n$ are pending at any time.

The (a) models include several tricky numerical puzzles as well well as programs that manipulate C strings, e.g. programs creating copies or concatenations of strings. The translation scheme [GI12] generates models that detect out-of-bound errors.

Experimental Results

First, consider the tools Flata and Fast which are both based on precise reachability methods that use acceleration. Table 8.3 shows that Flata significantly outperforms Fast on a vast majority of benchmarks. Note that we could not make a comparison for (b) and (e) benchmarks since the Fast tool does not support transitions with non-deterministic updates like $x' \geq 2$. The Aspic tool manifests strengths and weaknesses of abstract interpretation: correctness of models can be usually verified quickly, however,

absence of abstraction refinement often leads to "don't know" answers for models which have an error trace.

ELDARICA and ARMC are tools based on interpolation-based predicate abstraction and it turns out that they successfully verify almost same models (the sole exception being COUSOT and LISTCOUNTER models). Comparing FLATA with ELDARICA (or with ARMC), one can observe that the tools behave in a complementary way. In some cases (examples (a)), the predicate abstraction method fails due to an unbounded number of loop unrollings required by refinement. In these cases, acceleration was capable to find the needed invariant rather quickly. On the other hand (examples (e)), the acceleration approach was unsuccessful in reducing loops with linear but non-octagonal relations. In these cases, the predicate abstraction found the needed Presburger invariants for proving correctness and error traces for the erroneous examples.

Last, we report on our reachability analysis technique that combines predicate abstraction and acceleration: static acceleration described earlier in this section and dynamic acceleration (CEGAAR) described in Section 7.3. The results on this set of benchmarks suggest that we have arrived at a fully automated verifier that is robust in verifying automatically generated integer programs with a variety of looping control structure patterns. An important question we explored is the importance of dynamic application of acceleration as well as of overapproximation and underapproximation. In some cases, such as mergesort from the (d) benchmarks and split_vc.1 from (b) benchmarks, the acceleration overhead does not pay off. The problem is that static acceleration tries to accelerate every loop in the CFG rather than accelerating the loops occurring on spurious paths leading to error. Acceleration of inessential loops generates large formulas as the result of combining loops and composition of paths during large block encoding. The CEGAAR algorithm is the only approach that could handle all of our benchmarks. There are cases in which the FLATA tool outperforms CEGAAR such as test2.optim from (e) benchmarks. We attribute this deficiency to the nature of predicate abstraction, which tries to discover the required predicates by several steps of refinement. In the verification of benchmarks using CEGAAR, acceleration was exact 11 times in total. In 30 cases, the over-approximation of the loops was successful. In 15 cases, over-approximation failed, and so the tool resorted to under-approximation. This suggests that all techniques that we presented are essential to obtain an effective verifier.

8.3 Termination Analysis

We first report on our method from Section 6.3. We have compared (Table 8.4) our method for termination of linear affine loops with polynomially bounded examples given in [CGLA$^+$08] and found the same termination preconditions as they do, with one exception, in which we can prove universal termination in integer input values (row 2 of Table 8.4).

We next report on our method from Section 7.2 that computes transition invariants and then applies Algorithm 4 from Section 6.2 to compute termination preconditions. We have considered those benchmarks from Section 8.2 that were obtained by trans-

Table 8.4: Termination preconditions for several program fragments from [CGLA+08]

Program	Cook et al. [CGLA+08]	Linear Affine Loops

if (lvar $\geq$ 0)
while (lvar $< 2^{30}$)
lvar = lvar $\ll$ 1;
	$lvar > 0 \vee lvar < 0 \vee lvar \geq 2^{30}$	$\neg(lvar{=}0) \vee lvar \geq 2^{30}$

while (x $\geq$ N) x = -2*x + 10;	$x > 5 \vee x + y \geq 0$	$x \neq \frac{10}{3} \Leftrightarrow \text{true}$

//@ requires $n > 200$
x = 0;
while (1)
if (x < n) { x=x+y;
if (x $\geq$ 200) break; }
	$y > 0$	$y{>}0$

lations that preserve termination properties, namely ANUBHAV and COUSOT from the (a) example set and then all (c), (d), and (f) examples. We have managed to compute termination sets shown in Table 8.5.

Table 8.5: Termination Sets for Integer Programs.

Model	Size $\|\mathbf{x}\|$	$\|Q\|$	$\|T\|$	Time [s]	Termination Set
(a) Examples from [JM06]					
anubhav (C)	29	20	25	3.2	$i \geq 0$
cousot (C)	29	31	34	4.0	$\bot$
(d) Examples from L2CA [BBH+06]					
listcounter (C)	4	31	35	1.2	$\top$
listreversal (C)	7	97	107	32.6	$\top$
(f) VHDL models from [SV07]					
counter (C)	2	6	13	0.8	$\bot$
register (C)	2	10	49	1.4	$\bot$
synlifo (C)	3	43	1006	1016.4	$\bot$

First, by computing octagonal abstractions of disjuncts of a transition invariant, we have verified universal termination of the ListCounter and ListReversal programs. Next, we have verified the Counter and SynLifo programs by computing the precise transition invariant and then the weakest non-termination set, which was empty in both cases. Thus, these models have infinite runs for any input values, which is to be expected as they encode the behavior of synchronous reactive circuits. Similarly, we have computed the preconditions for two numerical programs ANUBHAV and COUSOT.

9 Conclusions

9.1 Summary

We have presented several methods that solve various problems related to formal verification of programs that manipulate integer data. Most of the techniques are built upon a novel algorithm computing transitive closures of difference bounds, octagonal, and finite monoid affine relations, which are shown to be periodic. We have proved that this algorithm runs in EXPTIME in the size of the binary representation of the input relation. Moreover, the experimental evidence for difference bounds and octagonal relations shows that the algorithm scales well in the number of variables and is up to four orders of magnitude faster than previous algorithms.

Next, we have studied the conditional termination problem for the classes of periodic relations and showed that the weakest non-termination set is Presburger definable for relations from these classes. As a consequence, the weakest non-termination set is Presburger definable for flat counter automata. Moreover, we have proved that the weakest non-termination set of octagonal (difference bounds) relations is an octagon (difference bounds constraint) itself and, moreover, that it can be computed in polynomial time.

We have presented a semi-algorithmic method for reachability analysis of non-recursive integer programs that is based on computation of procedure summaries and is therefore modular. It uses the transitive closure algorithm as one of its main components. The algorithm computing the summary relation can be adapted to compute transition invariants which are known to be crucial in proving program termination. We show that transition invariants can be used, in combination with algorithms computing non-termination sets, to compute termination preconditions for integer programs.

Further, we have addressed the divergence problem of the predicate abstraction and showed that it can be alleviated by incorporating acceleration into the abstraction-refinement framework in order to generate interpolants that are inductive and rule out potentially infinite sets of spurious counterexamples. Last but not least, we have performed experiments with a number of benchmarks and showed that for many of them, our methods outperform other existing approaches to verification of integer programs.

9.2 Published Results

The transitive closure algorithm studied in Chapter 3 and Chapter 4 is an optimized version of an algorithm that we originally published in [BIK10]. Chapter 6 extends the results we published in [BIK12] with a PTIME algorithm for the weakest non-termination preconditions of octagonal relations. Section 7.3 presents a predicate abstraction-based

method that we published in [HIK+12]. In [HIG+12], we presented the FLATA tool which implements most of the techniques described in this thesis. Last but not least, a work on verification of programs manipulating integer arrays, not presented in this thesis, has been published in [BHI+09].

9.3 Future Work

The asymptotic bound on the running time of the acceleration algorithm presented in this thesis has been shown to be in EXPTIME. A natural question that arises is whether the acceleration problem can be proved to be NP or PSPACE complete. Another question is whether the search for the correct prefix and period of a relation, performed in the transitive closure algorithm, cannot be optimized more than how it is currently achieved by the MAXCONSISTENT and MAXPERIODIC procedures. For instance, if the size of the prefix and the period could be efficiently computed (or at least approximated) directly from the input relations, the search could be significantly improved for relations which have very high prefix or period. Furthermore, direct computability of the size of the prefix and period would render the procedures MAXCONSISTENT and MAXPERIODIC superfluous, which would further reduce the complexity of the algorithm.

On what concerns our study of the termination problem for difference bounds and octagonal relations, a direction that is worth pursuing is to study whether the linear ranking functions, for which we proved existence, can be efficiently computed. We have shown that a linear ranking function can be directly constructed from negative cycles in zigzag automata. The problem we encounter here is that the size of zigzag automata is exponential. The question therefore is whether the construction of the zigzag automaton can be bypassed.

More broadly, a possible future research could explore whether the methods presented in this thesis can be extended to larger classes of programs, for instance by allowing recursion and parallelism. A related line of study is whether the (accelerated) predicate abstraction can be extended to handle recursive programs, for instance by making it reason about trace summaries instead of the set of reachable states.

Bibliography

[AAB00] A. Annichini, E. Asarin, and A. Bouajjani. Symbolic techniques for parametric reasoning about counter and clock systems. In *Proc. of CAV*, volume 1855 of *LNCS*, pages 419–434, Berlin, Heidelberg, 2000. Springer Verlag.

[AAGP11] E. Albert, P. Arenas, S. Genaim, and G. Puebla. Closed-form upper bounds in static cost analysis. *Journal of Automated Reasoning*, 46(2), 2011.

[AD91] R. Alur and D. L. Dill. The theory of timed automata. In *proc. of REX Workshop*, volume 600 of *LNCS*, pages 45–73, Berlin, Heidelberg, 1991. Springer Verlag.

[AM04] R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proc. of STOC*, pages 202–211, New York, NY, USA, 2004. ACM.

[AM06] R. Alur and P. Madhusudan. Adding nesting structure to words. In *Proc. of DLT*, pages 1–13, Berlin, Heidelberg, 2006. Springer-Verlag.

[BBH+06] A. Bouajjani, M. Bozga, P. Habermehl, R. Iosif, P. Moro, and T. Vojnar. Programs with lists are counter automata. In *Proc. of CAV*, volume 4144 of *LNCS*, pages 517–531, Berlin, Heidelberg, 2006. Springer Verlag.

[BCG+09] D. Beyer, A. Cimatti, A. Griggio, M. E. Keremoglu, and R. Sebastiani. Software model checking via large-block encoding. In *Proc. of FMCAD*, pages 25–32. IEEE, 2009.

[BCM+92] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[BFLS05] S. Bardin, A. Finkel, J. Leroux, and P. Schnoebelen. Flat acceleration in symbolic model checking. In *Proc. of ATVA*, volume 3707 of *LNCS*, pages 474–488, Berlin, Heidelberg, 2005. Springer Verlag.

[BG99] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using qdds. *Formal Methods in System Design*, 14(3):237–255, 1999.

[BGI09] M. Bozga, C. Gîrlea, and R. Iosif. Iterating octagons. In *Proc. of TACAS*, volume 5505 of *LNCS*, pages 337–351, Berlin, Heidelberg, 2009. Springer Verlag.

[BH99] A. Bouajjani and P. Habermehl. Symbolic reachability analysis of fifo-channel systems with nonregular sets of configurations. *Theoretical Computer Science*, 221(1-2):211–250, 1999.

[BHHK10] R. Blanc, T. A. Henzinger, T. Hottelier, and L. Kovács. ABC: Algebraic bound computation for loops. In *Proc. of LPAR*, volume 6355 of *LNCS*, pages 103–118, Berlin, Heidelberg, 2010. Springer Verlag.

[BHI+09] M. Bozga, P. Habermehl, R. Iosif, F. Konečný, and T. Vojnar. Automatic verification of integer array programs. In *Proc. of CAV*, volume 5643 of *LNCS*, pages 157–172, Berlin, Heidelberg, 2009. Springer Verlag.

[BHMR07] D. Beyer, T. A. Henzinger, R. Majumdar, and A. Rybalchenko. Path invariants. In *Proc. of PLDI*, pages 300–309, New York, NY, USA, 2007. ACM.

[BHRV06] A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract regular tree model checking of complex dynamic data structures. In *Proc. of SAS*, volume 4134 of *LNCS*, pages 52–70, Berlin, Heidelberg, 2006. Springer Verlag.

[BHV03] A. Bouajjani, P. Habermehl, and T. Vojnar. Verification of parametric concurrent systems with prioritized fifo resource management. In *Proc. of CONCUR*, volume 2761 of *LNCS*, pages 174–190, Berlin, Heidelberg, 2003. Springer Verlag.

[BHV04] A. Bouajjani, P. Habermehl, and T. Vojnar. Abstract regular model checking. In *Proc. of CAV*, volume 3114 of *LNCS*, pages 372–386, Berlin, Heidelberg, 2004. Springer Verlag.

[BHZ08] R. Bagnara, P. M. Hill, and E. Zaffanella. An improved tight closure algorithm for integer octagonal constraints. In *Proc. of VMCAI*, volume 4905 of *LNCS*, pages 8–21, Berlin, Heidelberg, 2008. Springer Verlag.

[BI07] M. Bozga and R. Iosif. On flat programs with lists. In *Proc. of VMCAI*, volume 4349 of *LNCS*, pages 122–136, Berlin, Heidelberg, 2007. Springer Verlag.

[BIK10] M. Bozga, R. Iosif, and F. Konečný. Fast acceleration of ultimately periodic relations. In *Proc. of CAV*, volume 6174 of *LNCS*, pages 227–242, Berlin, Heidelberg, 2010. Springer Verlag.

[BIK12] M. Bozga, R. Iosif, and F. Konečný. Deciding conditional termination. In *Proc. of TACAS*, volume 7214 of *LNCS*, pages 252–266, Berlin, Heidelberg, 2012. Springer Verlag.

[BIL09] M. Bozga, R. Iosif, and Y. Lakhnech. Flat parametric counter automata. *Fundamenta Informaticae*, 91(2):275–303, 2009.

[BIP08] M. Bozga, R. Iosif, and S. Perarnau. Quantitative separation logic and programs with lists. In *Proc. of IJCAR*, volume 5195 of *LNCS*, pages 34–49, Berlin, Heidelberg, 2008. Springer Verlag.

[BK11] D. Beyer and M. E. Keremoglu. CPAchecker: A tool for configurable software verification. In *Proc. of CAV*, volume 6806 of *LNCS*, pages 184–190, Berlin, Heidelberg, 2011. Springer Verlag.

[BKRW10] A. Brillout, D. Kroening, P. Rümmer, and T. Wahl. An interpolating sequent calculus for quantifier-free Presburger arithmetic. In *Proc. of IJCAR*, volume 6173 of *LNCS*, pages 384–399, Berlin, Heidelberg, 2010. Springer Verlag.

[BL99] S. Bensalem and Y. Lakhnech. Automatic generation of invariants. *Formal Methods in System Design*, 15(1):75–92, July 1999.

[BLP06] S. Bardin, J. Leroux, and G. Point. Fast extended release. In *Proc. of CAV*, volume 4144 of *LNCS*, pages 63–66, Berlin, Heidelberg, 2006. Springer Verlag.

[BMMR01] T. Ball, R. Majumdar, T. Millstein, and Sriram K. Rajamani. Automatic predicate abstraction of C programs. In *Proc. of PLDI*, pages 203–213, New York, NY, USA, 2001. ACM.

[BMS05] A. R. Bradley, Z. Manna, and H. B. Sipma. Linear ranking with reachability. In *Proc. of CAV*, volume 3576 of *LNCS*, pages 491–504, Berlin, Heidelberg, 2005. Springer Verlag.

[Boi99] B. Boigelot. *Symbolic Methods for Exploring Infinite State Spaces*. PhD Thesis. Université de Liège, 1999.

[BPR02] T. Ball, A. Podelski, and S. K. Rajamani. Relative completeness of abstraction refinement for software model checking. In *Proc. of TACAS*, volume 2280 of *LNCS*, pages 158–172, Berlin, Heidelberg, 2002. Springer Verlag.

[Bra06] M. Braverman. Termination of integer linear programs. In *Proc. of CAV*, volume 4144 of *LNCS*, pages 372–385, Berlin, Heidelberg, 2006. Springer Verlag.

[Car87] H. Carstensen. Decidability questions for fairness in Petri nets. In *Proc. of STACS*, volume 247 of *LNCS*, pages 396–407, Berlin, Heidelberg, 1987. Springer Verlag.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. of POPL*, pages 238–252, New York, NY, USA, 1977. ACM.

[CE82] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proc. of Logic of Programs*, volume 131 of *LNCS*, pages 52–71, Berlin, Heidelberg, 1982. Springer Verlag.

[CFLZ08] N. Caniart, E. Fleury, J. Leroux, and M. Zeitoun. Accelerating interpolation-based model-checking. In *Proc. of TACAS*, volume 4963 of *LNCS*, pages 428–442, Berlin, Heidelberg, 2008. Springer Verlag.

[CGJ+03] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM*, 50(5):752–794, 2003.

[CGLA+08] B. Cook, S. Gulwani, T. Lev-Ami, A. Rybalchenko, and M. Sagiv. Proving conditional termination. In *Proc. of CAV*, volume 5123 of *LNCS*, pages 328–340, Berlin, Heidelberg, 2008. Springer Verlag.

[CH78] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conference Record of the Fifth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 84–97, Tucson, Arizona, 1978. ACM Press, New York, NY.

[CJ98] H. Comon and Y. Jurski. Multiple counters automata, safety analysis and presburger arithmetic. In *Proc. of CAV*, volume 1427 of *LNCS*, pages 268–279, Berlin, Heidelberg, 1998. Springer Verlag.

[Cou] J.M. Couvreur. PresTAF. `http://altarica.labri.fr/forge/projects/3/wiki/PresTAF`.

[Cra57] W. Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *The Journal of Symbolic Logic*, 22(3):250–268, September 1957.

[CSRL01] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[DdM] B. Dutertre and L. de Moura. The YICES SMT Solver. `http://yices.csl.sri.com/`.

[DIP01] Z. Dang, O. H. Ibarra, and P. S. Pietro. Liveness verification of reversal-bounded multicounter machines with a free counter. In *FSTTCS*, pages 132–143, 2001.

[EKS08] J. Esparza, S. Kiefer, and S. Schwoon. Abstraction refinement with Craig interpolation and symbolic pushdown systems. *JSAT*, 5(1-4):27–56, 2008.

[Eve03] G. Everest. *Recurrence sequences*. American Mathematical Soc., 2003.

[FL02] A. Finkel and J. Leroux. How to compose presburger-accelerations: Applications to broadcast protocols. In *Proc. of FST TCS*, volume 2556 of *LNCS*, pages 145–156, Berlin, Heidelberg, 2002. Springer Verlag.

[FLS07] A. Finkel, E. Lozes, and A. Sangnier. Towards model-checking programs with lists. In *Proc. of ILC*, volume 5489 of *LNCS*, pages 56–86, Berlin, Heidelberg, 2007. Springer Verlag.

[Gan12] P. Ganty. Personal communication, 2012.

[GHM+08] A. Gupta, T. A. Henzinger, R. Majumdar, A. Rybalchenko, and R. Xu. Proving non-termination. In *Proc. of POPL*, pages 147–158, New York, NY, USA, 2008. ACM.

[GI12] F. Garnier and R. Iosif. Personal communication, 2012.

[GM12] P. Ganty and R. Majumdar. Algorithmic verification of asynchronous programs. *ACM Trans. Program. Lang. Syst.*, 34(1):6:1–6:48, 2012.

[Gon] L. Gonnord. ASPIC. `http://laure.gonnord.org/pro/aspic/aspic.html`.

[GS64] S. Ginsburg and E. H. Spanier. Bounded Algol-like languages. *Trans. of the AMS*, 113(2):333–368, 1964.

[GS97] S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In *Proc. of CAV*, volume 1254 of *LNCS*, pages 72–83, Berlin, Heidelberg, 1997. Springer Verlag.

[HHHK05] V. Halava, T. Harju, M. Hirvensalo, and J. Karhumaki. Skolem's problem – on the border between decidability and undecidability, 2005.

[HHP09] M. Heizmann, J. Hoenicke, and A. Podelski. Refinement of trace abstraction. In *Proc. of SAS*, volume 5673 of *LNCS*, pages 69–85, Berlin, Heidelberg, 2009. Springer Verlag.

[HHP10] M. Heizmann, J. Hoenicke, and A. Podelski. Nested interpolants. In *Proc. of POPL*, pages 471–482, New York, NY, USA, 2010. ACM.

[HIG+12] H. Hojjat, R. Iosif, F. Garnier, F. Konečný, V. Kuncak, and P. Rümmer. A verification toolkit for numerical transition systems. In *Proc. of FM*, 2012. To appear.

[HIK+12] H. Hojjat, R. Iosif, F. Konečný, V. Kuncak, and P. Rümmer. Accelerating interpolants. In *Proc. of ATVA*, 2012. To appear.

[HIRV07] P. Habermehl, R. Iosif, A. Rogalewicz, and T. Vojnar. Proving termination of tree manipulating programs. In *Proc. of ATVA*, volume 4762 of *LNCS*, pages 145–161, Berlin, Heidelberg, 2007. Springer Verlag.

[HIV08a] P. Habermehl, Radu I., and T. Vojnar. A logic of singly indexed arrays. In *Proc. of LPAR*, volume 5330 of *LNCS*, pages 558–573, Berlin, Heidelberg, 2008. Springer Verlag.

[HIV08b] P. Habermehl, R. Iosif, and T. Vojnar. What else is decidable about integer arrays? In *Proc. of FoSSaCS*, volume 4962 of *LNCS*, pages 474–489, Berlin, Heidelberg, 2008. Springer Verlag.

[HJMM04] T. A. Henzinger, R. Jhala, R. Majumdar, and K. L. McMillan. Abstractions from proofs. In *Proc. of POPL*, pages 232–244, New York, NY, USA, 2004. ACM.

[HJMS03] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. Software verification with BLAST. In *Proc. of SPIN*, volume 2648 of *LNCS*, pages 235–239, Berlin, Heidelberg, 2003. Springer Verlag.

[Iba78] O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25(1):116–133, January 1978.

[Jan90] P. Jančar. Decidability of a temporal logic problem for Petri nets. *Theoretical Computer Science*, 74(1):71–93, 1990.

[JM06] R. Jhala and K. L. McMillan. A practical and complete approach to predicate refinement. In *Proc. of TACAS*, volume 3920 of *LNCS*, pages 459–473, Berlin, Heidelberg, 2006. Springer Verlag.

[KLR10] D. Kroening, J. Leroux, and P. Rümmer. Interpolating quantifier-free Presburger arithmetic. In *Proc. of LPAR*, volume 6397 of *LNCS*, pages 489–503, Berlin, Heidelberg, 2010. Springer Verlag.

[KM] Nils Klarlund and Anders Møller. MONA. `http://www.brics.dk/mona/`.

[LS05] J. Leroux and G. Sutre. Flat counter automata almost everywhere! In *Proc. of ATVA*, volume 3707 of *LNCS*, pages 489–503, Berlin, Heidelberg, 2005. Springer Verlag.

[McM05] K. L. McMillan. An interpolating theorem prover. *Theoretical Computer Science*, 345(1):101–121, 2005.

[Min67] M. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

[Min06] A. Miné. The octagon abstract domain. *Higher-Order and Symbolic Computation*, 19(1):31–100, 2006.

[Mon12] D. Monniaux. Personal Communication, 2012.

[MS77] A. Mandel and I. Simon. On finite semigroups of matrices. *Theoretical Computer Science*, 5(2):101–111, 1977.

[PR04a] A. Podelski and A. Rybalchenko. A complete method for the synthesis of linear ranking functions. In *Proc. of VMCAI*, volume 2937 of *LNCS*, pages 465–486, Berlin, Heidelberg, 2004. Springer Verlag.

[PR04b] A. Podelski and A. Rybalchenko. Transition invariants. In *LICS'04*, pages 32–41, 2004.

[PR05] A. Podelski and A. Rybalchenko. Transition predicate abstraction and fair termination. In *Proc. of POPL*, pages 132–144, New York, NY, USA, 2005. ACM.

[PR07] A. Podelski and A. Rybalchenko. ARMC: The logical choice for software model checking with abstraction refinement. In *Proc. of PADL*, volume 4354 of *LNCS*, pages 245–259. Springer Verlag, Berlin, Heidelberg, 2007.

[Pre29] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. *Comptes rendus du I Congrés des Pays Slaves*, page 92–101, 1929.

[PRK+] W. Pugh, E. Rosser, W. Kelly, D. Wonnacott, and T. Shpeisman. Omega. `http://www.cs.umd.edu/projects/omega/`.

[QS82] J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in cesar. In *Proc. of the 5th Colloquium on International Symposium on Programming*, volume 137 of *LNCS*, pages 337–351, Berlin, Heidelberg, 1982. Springer Verlag.

[Ram30] F. P. Ramsey. On a problem of formal logic. *Proc. of the London Mathematical Society*, 30:264–285, 1930.

[Reu90] C. Reutenauer. *The mathematics of Petri nets.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[RSS07] A. Rybalchenko and V. Sofronie-Stokkermans. Constraint solving for interpolation. In *Proc. of VMCAI*, volume 4349 of *LNCS*, pages 346–362, Berlin, Heidelberg, 2007. Springer Verlag.

[Rüm08] P. Rümmer. A constraint sequent calculus for first-order logic with linear integer arithmetic. In *Proc. of LPAR*, volume 5330 of *LNCS*, pages 274–289, Berlin, Heidelberg, 2008. Springer Verlag.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1986.

[Sch00] B. De Schutter. On the ultimate behavior of the sequence of consecutive powers of a matrix in the max-plus algebra. *Linear Algebra and its Applications*, 307:103–117, 2000.

[SV07] A. Smrcka and T. Vojnar. Verifying parametrised hardware designs via counter automata. In *Proc. of HVC*, volume 4899 of *LNCS*, pages 51–68, Berlin, Heidelberg, 2007. Springer Verlag.

[SVG91] K. Sohn and A. Van Gelder. Termination detection in logic programs using argument sizes. In *Proc. of PODS*, pages 216–226, New York, NY, USA, 1991. ACM.

[Tiw04] A. Tiwari. Termination of linear programs. In *Proc. of CAV*, volume 3114 of *LNCS*, pages 70–82, Berlin, Heidelberg, 2004. Springer Verlag.