Diploma Thesis

Widening for Automata

Vijay D'silva

Institut Für Informatik
Universität Zürich

June 01, 2006

“*Never promise to do the possible.
Anyone could do the possible. You should promise to do the impossible,
because sometimes the impossible* `was` *possible, if you could find the right
way, and at least you could often extend the limits of the possible.*”

— Terry Pratchett, Going Postal

To my parents, Tony and Viji D'Silva and my sister Vanitha.

Abstract

Computing fixpoints of increasing sequences of sets is an important problem in many areas of computer science including algorithmic verification, program analysis, inductive inference and systems biology. For most problems, the fixpoint computation does not terminate, so an approximate solution has to be found. Widening is a technique to compute an over-approximation of an infinite, increasing sequence of sets. In this thesis, we present a framework for constructing widening operators for fixpoint computations over sets represented as automata. Many widening operators for automata that appear in the literature are instances of our framework. Moreover, two inductive inference algorithms in the literature naturally fall out as instances of this framework. We identify general criteria that characterise the effect of widening and use these criteria to study various properties of widening operators. We also provide several new results and generalise existing results about widening operators and inductive inference algorithms. Finally, we show how a widening operator defined in our framework can be combined with algorithms for automated verification of infinite state systems and provide a heuristic for generating counterexamples if verification fails.

Zusammenfassung

Die Bestimmng des kleinsten Fixpunktes einer aufsteigender Folge von Mengen ist ein zentrales Problem in vielen Teilgebieten der Informatik, wie zum Beispiel in der algorithmischen Verifikation, der Programm-Analyse, induktiver Inferenz und der Bioinformatik. Oft ist eine genaue Bestimmung des Fixpunkts nicht möglich; eine Überapproximation ist jedoch meist ausreichend. *Widening* ist eine Methode die zur Berechnung solcher Überapproximationen eingesetzt wird. Diese Arbeit präsentiert einen Framework zur Konstruktion von Widening Operatoren zur Berechnung solcher Überapproximationen, die als Automaten dargestellt werden können. Unser Framework stellt eine Verallgemeinerung vieler in der einschlägigen Literatur verwendeten Widening-Operatoren dar. Insbesondere deckt unsere Kategorisierung auch zwei auf induktiver Inferenz basierende Algorithmen ab. Wir definieren allgemeine Kriterien die den Effekt von Widening erfassen und wenden diese auf Widening Operatoren an. Wir präsentieren sowohl neue als auch Generalisierungen bereits vorhanderer Ergebnisse für Widening Operatoren und Algorithmen basierend auf induktiver Inferenz. Wir zeigen wie mit Hilfe unseres Frameworks erstellte Operatoren mit Algorithmen zur automatischen Verifikation von Systemen mit unendlichen Zustandsräumen kombiniert werden können und präsentieren Heuristiken zur Berechnung von Gegenbeispielen für den Fall dass die Verifikation fehlschlägt.

Acknowledgements

No man is an Iland, intire of itselfe;
Every man is a peece of the Continent, a part of the maine;
...
And therefore never send to know for whom the bell tolls;
It tolls for thee.
— John Donne, Meditation XVII, 1624

Today, I will complete one more trip around that distant ball of fire. As I write the last words of this thesis, it is in the humble yet comforting knowledge that I could not have done it alone. Much credit is due to:

Felix Klaedtke, my supervisor, for an extended intellectual high that a stimulating topic and a receptive collaborator made possible. Every incomplete and sloppy draft I wrote came back bedecked in red. Every harebrained idea I had was heard in full and followed by a *suggestion* of why it might be impractical. Every discussion we had ended with: "*If you have anything, just drop by.*" Such sincerity, unimposing guidance and availability is easy to take for granted and difficult to overstate. My hope is that I have imbibed some of these qualities along the way.

Prof. David Basin my supervising professor at ETH.

Prof. Abraham Bernstein, my supervising professor from the University, for his philanthropic support, pragmatic suggestions and for the small extension that has made all the difference.

Prof. Daniel Kröning, my part-time employer, for listening to me rattle about greatest fixpoint computations, for giving me time off during the last month and for reading and commenting on sections of my thesis on such short notice. I could not have asked for more.

Georg Weissenbacher from the next office. The Zusammenfassung owes its existence to him.

Yury Chebiryak from the next to next office. His liberal use of the word "*aaghlee*" on perusing my drafts led me to simplify my notation and make my proofs more readable.

Ádám Darvas. His questioning the legitimacy of my genealogy convinced me to jettison my reservations about time and write this page.

This thesis is the penultimate requirement of many, including several non-academic obstacles, that I had to meet on the way to my degree. I am indebted to many people who have helped me along the way, often for no reason and without even knowing me.

I was employed by David Basin as an academic guest in 2005 and by Daniel Kröning in 2006. Their support literally helped me survive and gave me access to resources and privileges no full-time student has.

My interaction with the University of Zurich began in a meeting with Prof. Klaus Dittrich, who was and remains ever affable and approachable. His colleague, Prof. Marting Glinz, forbore with my ambitious plans and systematically identified the optimal choices for me to complete my degree. Special thanks are due to Samira Ludi from the Lehrbereichssekretariat for answering several questions and helping me with much paperwork.

The summer semester in 2005 was my first and most intense at the University. Of the helpful lecturers, assistants and students, Patrick Reolon stands out. He befriended me, assisted me with several assignments and provided me with valuable course material. I shall forever be mystified by and regret how I managed to get our team both full points and disqualification in the cookie monster tournament.

The winter of 2005 was a tumultuous period in my studies because of several visa issues. Frau. E. Stucki from the Migrationsamt extended my residence permit till June 2006 and gave me exactly the time I needed. Many people both at the University and ETH provided suggestions, advice and support during this period. Ms. Monia Cursoli from the ETH Rektoratdienste made several phone calls for me and bore with my vacillation. Her confidence in my potential is overwhelming and for her efforts at that time, I am deeply grateful. Ms. Ruth Hidalgo's casual "*Wie gehts?*" was answered with 'Die Leiden des jungen Vijays'. She listened to it all and made some constructive suggestions, which have their part in my being here today.

An even earlier but significant event leading to this thesis is my arrival in Switzerland, which was on Prof. Bertrand Meyer's invitation. Ruth Bürkli's curious combination of blasé yet concerned pragmatism entered my world with a discussion about pots and pans and retains its presence in my health insurance. On my second day in this country, Till Bay epitomised hospitality by offering me a place to stay and helping me find an apartment.

Marco Vrankic made that apartment a home and we have had interesting times with several flatmates. Philipp Herrmann introduced me to his all-embracing social circle and Jonathan Landau is my frequent and welcome break from the ordinary. I have spent many a morning, afternoon or evening that shall be filed under nostalgia in the company of the software engineers, infsecies, formal verifiers, statisticians, biologists and several others who defy classification.

Though he will never read this thesis, Terry Pratchett's Moist von Lipwig has been my model and mantra for the last month. Finally, my parents' two visits were a welcome break and the silent support of my family over the last month has been exactly what I needed.

Contents

Chapter 1

Introduction

Fixpoint computations are ubiquitous in computer science. Almost any non-trivial task involves repeating a sequence of steps and can be viewed as a fixpoint computation. In this thesis, we develop techniques to over-approximate the fixpoint of an infinite, increasing sequence of sets represented as automata. Such sequences arise naturally in debugging and analysis of programs, computational learning theory, systems biology and algorithmic verification.

Such a bombastic claim cannot be allowed to pass unjustified. Let us provide a few examples, starting with program analysis. Points-to analysis is an important program analysis problem. The first step in this analysis is to identify the set of variables, called the points-to set, pointing to a memory location of interest when the program is initialised. In each subsequent analysis step, new variables that may point to the target memory location are added to the points-to set. The solution is computed if the points-to set is not further changed by any execution of the program.

What about computational learning theory? Gold [1967], states in his seminal paper:"*I wish to construct a precise model for the intuitive notion* `'able to speak a language'` *in order to be able to investigate theoretically how it can be achieved artificially*". The model proposed consists of a learning algorithm used by a learner who is provided with an infinite sequence of examples from a formal language. The learner uses the algorithm to infer the rules of the language from the examples. The learning process ends if the learner infers no new rules after a certain point and is successful if the learner has inferred the rules of the language.

Systems biology, very broadly defined, is the study of biological models. A biological model is typically a set of equations. Two questions can be asked of such a model. First, if the model is started from a biologically plausible configuration, does it evolve to exhibit experimentally observed behaviour? Second, is the model sufficient to characterise all possible initial configurations from which a biological system might evolve to exhibit a certain behaviour? The first question is usually answered by performing computer simulations of the model. An approach to answer the second question is to construct and analyse a discrete and continuous abstraction of the mathematical model. The analysis begins with a target set representing the target behaviour. In each step, configurations from which the abstract model may evolve to a configuration in the target set are added to the target set. The analysis terminates if no such configurations exist. If the set of possible initial configurations is contained in the final target set, we obtain an affirmative answer to the second question.

Algorithmic verification involves providing a guarantee that a mathematically specified property is satisfied by a model of a computer system. A typical system is an electronic controller, which receives input from the environment and produces control signals. A property might assert that a certain sequence of control signals is never produced. The system is analysed by computing the set of initial states of the controller and the sequence of states reached in each time step for all possible inputs. The system satisfies the property if the set of reachable states that has been computed does not change and the forbidden sequence of signals is never produced.

Each problem we described requires computing the fixpoint of an increasing sequence of sets. If the sequence is infinite, the analysis procedures described never terminate. A possible solution is to devise a procedure to compute the fixpoint by examining a finite sequence of sets. In general, the fixpoint may not be computable, so an approximation of the fixpoint has to be found. Our goal is to compute such approximate solutions for algorithmic verification problems. As a guarantee of correctness must be provided and a malfunction in the system may have dire consequences (both financial and social), it is acceptable to say that a system does not satisfy a property when it does. However, it not acceptable to say that the system satisfies a property if it does not. In short, an over-approximation of the fixpoint suffices but an under-approximation does not. In this thesis, we devise techniques for computing such over-approximations.

1.1 Widening and Automata

The problems with precisely computing the fixpoint of an infinite sequence are computability and representation. That is, the fixpoint may not be computable and may not have an efficient or finite representation. We address the computability problem by finding an over-approximation of the fixpoint and the representation problem by using automata to encode sets in a fixpoint computation.

Abstract interpretation [Cousot and Cousot 1977; Cousot 1978] is a theory for approximating set operations. If a fixpoint computation with sets requires infinite steps, Cousot and Cousot [1977] define an extrapolation technique called *widening* to accelerate convergence to the fixpoint and if possible enforce termination of the computation. A widening operator is used to detect and generalise an increment between sets in a fixpoint computation. The extrapolation introduced by a widening step is usually larger than the difference between two sets in the fixpoint computation, so convergence to the fixpoint is accelerated. If the set obtained after a widening step is an over-approximation of the fixpoint, the computation also terminates. As the over-approximation may be too imprecise, a technique called *narrowing* is used to improve the precision of the solution.

Defining widening operators for fixpoint computations with arbitrary sets is insufficient for practical purposes as complicated data structures may be required to represent the sets and set theoretic operations on these data structures may be computationally expensive. Our solution is to choose automata as a representation. As data structures, automata are simple but can encode infinite sets. They are closed under Boolean operations such as union, intersection and complement as well as concatenation, homomorphisms and many language-theoretic operations. Automata admit canonical representations and there exist libraries implementing many standard operations on automata [Klarlund et al. 2002; LASH].

Automata also have a theoretical appeal. Many interesting mathematical structures can be encoded as automata [Khoussainov and Nerode 1995; Blumensath and Grädel 2000]. Examples include, Presburger arithmetic [Büchi 1960; Cobham 1969; Semenov 1977; Bruyère et al. 1994], linear arithmetic over the integers and reals [Boigelot et al. 2005], temporal logics [Vardi and Wolper 1994], and the second order monadic theory of one successor [Büchi 1960;1962]. Algorithmic verification of finite state systems using temporal or modal logic specifications has been formalised in terms of automata [Vardi and Wolper 1986; Kupferman et al. 2000] and implemented in several tools [Bardin et al. 2003; Yavuz-Kahveci et al. 2005; Holzmann 2004].

Applying widening techniques to accelerate automata based verification is not new. Lesens et al. [1997] develop a model checking procedure for networks of automata. Widening is used to accelerate the computation of network invariants. Bouajjani et al. [2000] suggest a simple heuristic for widening sets represented by regular expressions for acceleration of their model checking algorithms. Touili [2001] generalises this heuristic to a larger class of regular expressions. Boigelot et al. [2003;2004] use simulation relations to detect increments between automata and generalise the increment. They apply this method to extrapolate the language of automata encoding arithmetic but remark that such a technique is applicable to any automata based representation. Bartzis and Bultan [2004] present a widening operator for automata encoding Presburger arithmetic. All these widening techniques use some criteria to compare states in a sequence of automata and detect an increment. The extrapolation step involves merging states in an automaton, which has the effect of increasing the number of loops and final states in the automaton. As a result, the language accepted increases. In this thesis, we show that barring the widening operator of [Touili 2001], all other widening operators mentioned above are instances of the widening framework we develop.

Cousot and Cousot [1992c] observe that in comparison to other techniques defined in abstract interpretation, "*the design of widenings and narrowings is often thought of to be more difficult since it appears as a heuristic to cope with induction.*" Over a decade later, Halbwachs [2006] remarked that "*widening is still often considered as a kind of dirty heuristic in the model-checking community.*" We see two possible reasons for this status quo. Either widening does not provide significant benefits over existing techniques, or designing widening operators is a cumbersome and intimidating task that is best avoided. Indeed, the widening operators presented in [Boigelot et al. 2003] and [Bartzis and Bultan 2004] incorporate many seemingly complicated steps. Though the authors provide formal results about and demonstrate the practical utility of their widening operators, a rigorous analysis of the effect of widening is difficult.

1.2 Contents of this Thesis

The aim of this thesis is ambitious. We wish to dispel some misconceptions about widening, at least in the context of automata, by showing that it is possible to construct and rigorously analyse simple yet useful widening operators.

We construct widening operators from relations between states of automata using only simple set theoretic operations. A relation is used to detect an increment between or a pattern in the languages accepted by two automata in a fixpoint computation. The states of the larger automaton are partitioned using this relation and merged using the standard quotient operation for automata.

There are three questions we study for each widening operator. (1) Does it extrapolate the language of the automaton? (2) Does the fixpoint computation with widening terminate? (3) Under what conditions can the precise fixpoint be computed? We provide general conditions that aid answering questions about extrapolation and precision and provide examples if a computation with widening does not terminate. We analyse eight different widening operators and show that five of them correspond to widening operators or inductive inference algorithms existing in the literature. Following the analysis of widening operators, we show how our widening framework can be combined with model checking algorithms.

This thesis is organised as follows: In Chapter 2 we present the relevant background material. We believe this thesis is self contained and only assumes familiarity with set theory. In Chapter 3, we present and study a framework for widening with automata. In Sections 3.1 and 3.2 we provide a technical introduction to widening and formalise the connection to computational learning theory. In Section 3.3, we provide an algorithm for constructing widening operators and in Section 3.4, we study the solutions that can be computed using widening. In Chapter 4 we introduce and analyse different widening operators. In particular, the widening operators of [Bartzis and Bultan 2004] and [Boigelot et al. 2003] are studied in Sections 4.4 and 4.5 respectively. We present the use of widening with model checking in Chapter 5 and conclude with a discussion of open research problems in Chapter 6.

Chapter 2

Background

In this chapter, we fix our notation and review the relevant concepts of formal language theory and abstract interpretation. For introductory expositions to these topics, see [Hopcroft and Ullman 1979] for formal languages and automata, [Thomas 1997] for automata over infinite words, [Davey and Priestley 1990] for lattice theory and [Cousot 2005] for abstract interpretation.

We use standard set theoretic notation. The difference between two sets P and Q, denoted $P \backslash Q$, is the set of elements in P that are not in Q. The powerset of Q is denoted $\wp(Q)$. The set of natural numbers is denoted $\mathbb{N}$. Given a relation $\mathcal{R}$, $\mathcal{R}^{-1}$ denotes the inverse relation, $\mathcal{R} \circ \mathcal{R}$ denotes relational composition, $\mathcal{R}^{=}$ the reflexive closure and $\mathcal{R}^{*}$ the reflexive, transitive closure of $\mathcal{R}$. Given an equivalence relation $\sim$ on a set S, $[s]_{\sim}$ denotes the equivalence class of $s \in S$. Partitions and equivalence relations are equivalent notions. The block of a partition π of S containing $s \in S$ is denoted $[s]_{\pi}$. A partition π' is said to *refine* π, denoted $\pi' \preceq \pi$ iff every block of π is a union of blocks of π'. In this case, we also say that π is *coarser* than π' or that π' is *finer* than π. The *trivial partition* of a set S is the set of blocks $\{s\}$ where $s \in S$.

2.1 Regular Languages and Automata on Finite Words

An alphabet Σ is a nonempty finite set of symbols. A word w is a sequence of symbols. The length of a finite word w is denoted $|w|$. The set Σ^{*} denotes all words of finite length constructed from symbols in Σ. The empty word λ is the unique word of length 0 in Σ^{*}. The concatenation of two words u and v is denoted $u \cdot v$ or just uv. A prefix of a word w is a possibly empty sequence of leading symbols in w and a suffix, a possibly empty sequence of trailing symbols.

A language $L \subseteq \Sigma^{*}$ is a set of words. Given a language, we define the prefixes of L as $Pre(L) = \{u \in \Sigma^* |$ for some $v, uv \in L\}$ and suffixes of L as $Suff(L) = \{v \in \Sigma^* |$ for some $u, uv \in L\}$. The concatenation of two languages L and L', is the language $LL' = \{uv | u \in L, v \in L'\}$. If $L = \{u\}$ or $L' = \{v\}$, we write uL' or Lv. for the concatenation of the two languages. The syntactic right congruence for languages is called the *Nerode equivalence.*

Definition 1 (Nerode Equivalence). The Nerode equivalence, $\sim_L \subseteq \Sigma^* \times \Sigma^*$, is a relation between words such that $u \sim_L v$ iff for all $w \in \Sigma^*$, $uw \in L \Leftrightarrow vw \in L$.

A language L is *regular* iff $\sim_L$ is of finite index. By the Myhill-Nerode theorem, the Nerode equivalence is the coarsest right congruent equivalence relation of finite index for a regular language [Myhill 1957; Nerode 1958]. The class of regular languages is closed under the Boolean operators union, intersection and complement, the concatenation and Kleene star operators, homomorphisms and inverse homomorphisms and left and right-quotient. Another characterisation of the regular languages provided by the Myhill-Nerode theorem is that they are accepted by finite automata.

Definition 2 (Finite Automaton). A finite automaton A is a tuple $(Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is a finite input alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states and $\delta : Q \times \Sigma \to \wp(Q)$ is the transition function.

A finite automaton as defined above is also called a *nondeterministic finite automaton* (NFA). An automaton A is a *deterministic finite automaton* (DFA) if for all states $q \in Q$ and symbols $a \in \Sigma$, $|\delta(q, a)| \leq 1$. Note that every DFA is an NFA. An automaton is *strictly nondeterministic* if it is not deterministic.

A *transition* is a tuple (q, a, q') such that $q' \in \delta(q, a)$. The function $\delta^* : Q \times \Sigma^* \to \wp(Q)$ is the extension of the transition relation to words such that $\delta^*(q, \lambda) = \{q\}$ and for all words $w \in \Sigma$ and symbols $a \in \Sigma$, $\delta^*(q, wa) = \bigcup_{q' \in \delta^*(q,w)} \delta(q', a)$. If it is clear that an automaton is a DFA, by abuse of notation, δ is written as a function from $Q \times \Sigma$ to Q and δ^* as a function from $Q \times \Sigma^*$ to Q.

A word $w \in \Sigma^*$ is accepted by an automaton iff $\delta^*(q_0, w) \cap F \neq \emptyset$. The language accepted by a finite automaton A, denoted $L(A)$, is the set of all words accepted by A. The set of prefixes of a state q, $Pre(q) = \{w \in \Sigma^* | q \in \delta^*(w, q_0)\}$, is the set of words by which q is reachable from the initial state. The set of suffixes of a state q, $Suff(q) = \{w \in \Sigma^* | \delta^*(w, q) \cap F \neq \emptyset\}$, is the set of words by which a final state is reachable from q. By definition, $L(A) = Suff(q_0)$. The functions Pre and $Suff$ are naturally extended to sets of states as $Pre(Q) = \bigcup_{q \in Q} Pre(q)$ and $Suff(Q) = \bigcup_{q \in Q} Suff(q)$.

An automaton is *minimal* iff no two states accept the same language. That is, for all any two states q and q', $Suff(q) \neq Suff(q')$. An automaton is *complete* if for all states q and symbols $a \in \Sigma$, $\delta(q, a) \neq \emptyset$. An automaton is *trim* iff for every state q, there exists $w \in \Sigma^*$ such that $\delta^*(q, w) \cap F \neq \emptyset$.

Given a partition of the states of an automaton, a *quotient automaton* can be defined.

Definition 3 (Quotient Automaton). Let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and π be a partition of Q. The quotient automaton $A/\pi = (Q', \Sigma, \delta', [q_0]_\pi, F')$ is defined as follows: $Q' = \{[q]_\pi | q \in Q\}$, $F' = \{[q]_\pi | q \in F\}$, $\delta' : Q' \times \Sigma \to \wp(Q')$ is the function $\delta'([q]_\pi, a) = \bigcup_{p \in [q]} \{[q']_\pi | q' \in \delta(p, a)\}$.

The quotient automaton is constructed by merging states of A, preserving transitions between states and marking partitions containing a final state as final. Note that the quotient of a DFA is not necessarily a DFA. Given two partitions π and π' of the states of an automaton A, if $\pi \preceq \pi'$ then $L(A/\pi) \subseteq L(A/\pi')$. Given an equivalence relation $\equiv$ on the states of A, the quotient automaton $A/\equiv$ is defined in terms of the partition induced by $\equiv$.

Let π be the partition of a regular language L induced by the Nerode equivalence. The *canonical automaton* for L, $A_L = (Q, \Sigma, \delta, q_0, F)$ is defined as follows: $Q = \pi$, $F = \{[w]_\pi | w \in L\}$, $q_0 = [\lambda]_\pi$ and $\delta([w]_\pi, a) = \{[wa]_\pi\}$. The canonical automaton for a regular language L is the unique minimal and deterministic automaton accepting L.

2.2 Automata on Infinite Words

An infinite word or ω-word α over an alphabet Σ is a mapping $\alpha : \mathbb{N} \to \Sigma$. We denote the set of ω-words over the alphabet Σ by Σ^ω. The concatenation of a word $u \in \Sigma^*$ and an ω-word $\beta \in \Sigma^\omega$ is denoted $u\beta$. Concatenation is not defined between ω-words. Every $\alpha \in \Sigma^\omega$ has infinitely many factorisations of the form $u\beta$, where $u \in \Sigma^*$ is a finite prefix and $\beta \in \Sigma^\omega$ is an infinite suffix. An ω-word, which admits a factorisation of the form uv^ω is *ultimately periodic*, where u is the prefix and v is the *period* of the factorisation. Unlike words of finite length, being a suffix does not define an order relation over Σ^ω. For example, the ω-words $(ab)^\omega$ and $(ba)^\omega$ are suffixes of each other.

An ω-language $\mathcal{L} \subseteq \Sigma^\omega$ is a set of ω-words. Given an ω-language $\mathcal{L}$, the prefixes of $\mathcal{L}$ are defined as the set $Pre(\mathcal{L}) = \{u \in \Sigma^* |$ for some $\beta \in \Sigma^\omega, u\beta \in \mathcal{L}\}$ and the infinite suffixes of $\mathcal{L}$ are the set $Suff_\omega(\mathcal{L}) = \{\beta \in \Sigma^\omega |$ for some $u \in \Sigma^*, u\beta \in \mathcal{L}\}$. There exists a syntactic right congruence for ω-languages, analogous to the Nerode equivalence [Maler and Staiger 1997].

Definition 4. The syntactic right congruence for ω-words $\approx_\mathcal{L} \subseteq \Sigma^* \times \Sigma^*$ is a relation such that for $u, v \in \Sigma^*$ and $\mathcal{L} \subseteq \Sigma^\omega$, $u \approx_\mathcal{L} u$ iff for all ω-words $\beta \in \Sigma^\omega$, $u\beta \in \mathcal{L} \Leftrightarrow v\beta \in \mathcal{L}$.

An ω-language $\mathcal{L}$ with $\approx_\mathcal{L}$ of finite index [Maler and Staiger 1997] is called *finite state* [Staiger 1983]. In contrast to the regular languages, every finite state ω-language may not be accepted by an automaton isomorphic to its syntactic right congruence. We restrict our attention to ω-languages accepted by minimal automata isomorphic to their syntactic right congruence, called *weak ω-languages.* The automata accepting these languages are *weak deterministic Büchi automata* (WDBA).

A Büchi automaton is a finite automaton as in Definition 2. The notion of determinism, nondeterminism and the notation for Büchi automata are identical to those for finite automata. Consider a Büchi automaton $B = (Q, \Sigma, \delta, q_0, F)$. A state $q \in Q$ is *recurrent* iff there exists $w \in \Sigma^*$ such that $q \in \delta^*(q, w)$. A state that is not recurrent is *transient.* A *run* ρ of B on an ω-word α is a mapping $\rho : \mathbb{N} \to Q$ such that $\rho(0) = q_0$ and for all $i \geq 0$, $\rho(i+1) \in \delta(\rho(i), \alpha(i))$. By definition, a run starts in the initial state and respects the transition function of the automaton.

Let $\mathit{Inf}(\rho)$ be the set of states that occur infinitely often in a run ρ of A. By abuse of notation, let $\mathit{Inf}(\alpha, B)$ be the states that occur infinitely often in a run of B on α. A run ρ is accepting iff $\mathit{Inf}(\rho) \cap F \neq \emptyset$. An ω-word α is accepted by a Büchi automaton iff the automaton has an accepting run on α. The language $L_\omega(B)$ of a Büchi automaton B is the set of ω-words accepted by B. The ω-regular languages are the set of languages accepted by nondeterministic Büchi automata.

A *co-Büchi automaton* is defined as a finite automaton. A run ρ of a co-Büchi automaton A is accepting iff $\mathit{Inf}(\rho) \cap F = \emptyset$. Words and the language accepted by a co-Büchi automaton are similarly defined.

Definition 5 (Weak Büchi automaton). A Büchi automaton $B = (Q, \Sigma, \delta, q_0, F)$ is weak iff there exists a partition of Q into disjoint subsets $Q_1, \ldots, Q_n$ such that:

- for each Q_i, either $Q_i \subseteq F$ or $Q_i \cap F = \emptyset$.
- there exists a partial order $\leq$ on the sets $Q_1, \ldots, Q_n$ such that if $q \in Q_i$, $q' \in Q_j$ and $q' \in \delta(q, a)$ for some $a \in \Sigma$, then $Q_j \leq Q_i$.

A *strongly connected component* of an automaton with states Q is a set $S \subseteq Q$ satisfying that for all $q, q' \in S$, there exists $w \in \Sigma^*$ such that $q' \in \delta^*(q, w)$ and there exists no $S' \supset S$ satisfying this property. Note that each Q_i in Definition 5 has to be a union of strongly connected components. Thus, the strongly connected components of a weak Büchi automaton either contain only final states or non-final states.

A *weak deterministic Büchi automaton* (WDBA) is a Büchi automaton that is both weak and deterministic. We also say weak deterministic automaton for a weak deterministic Büchi automaton. As we do not refer to finite automata as weak, this nomenclature is unambiguous. Weak deterministic automata have many appealing properties. They can be determinised using the breakpoint construction, which can be implemented in a manner similar to the powerset construction for finite automata [Miyano and Hayashi 1984; Kupferman and Vardi 2001]. Further, a WDBA B admits a minimal normal form, which is isomorphic to the syntactic right congruence for $L_\omega(B)$ [Maler and Staiger 1997]. Using a preprocessing step suggested by Löding [2001], WDBA can be minimized using the classical algorithm for finite automata [Hopcroft 1971].

The quotient of a Büchi automaton with respect to a partition π is defined as before. It also holds that for any Büchi automaton B and partition π, $L_\omega(B) \subseteq L_\omega(B/\pi)$ [Etessami et al. 2005].

2.3 Lattice Theory and Abstract Interpretation

A *partial order* $\sqsubseteq$ on a set S is a binary relation that is reflexive, antisymmetric and transitive. A *preorder* $\preceq$ is a binary relation that is reflexive and transitive. A *poset* $\langle S, \sqsubseteq\rangle$ is a set S equipped with a partial order. We write $x \sqsupseteq y$ for $y \sqsubseteq x$ and $x \sqsubset y$ for $x \sqsubseteq y \wedge x \neq y$. A *chain* of a poset $\langle S, \sqsubseteq\rangle$ is a subset X of S such that for all $x, y \in X$, $x \sqsubseteq y$ or $y \sqsubseteq x$. For example, the set $\mathbb{N}$ is a chain of the poset $\langle \mathbb{N}, \leq\rangle$. A poset $\langle S, \sqsubseteq\rangle$ satisfies the *ascending chain condition* iff any infinite sequence $x_0 \leq x_1 \leq \ldots$ of elements of S is not strictly increasing. The *duality principle* is that if a statement is true of all posets, the dual of the statement is also true of for all posets.

An *upper bound* of a set $X \subseteq S$ is an element $u \in S$ such that for all $x \in X$, $x \sqsubseteq u$. The *least upper bound* of X, denoted $lub(X)$ is an element of S such that $lub(X) \sqsubseteq u$ for all upper bounds u of X. The element $lub(X)$ may not exist, but if it does, is unique though it need not be an element of X. A lower bound of a set $X \subseteq S$ is an element $l \in S$ such that $l \sqsubseteq x$ for all $x \in X$. The *greatest lower bound*, $glb(X)$ satisfies that $l \sqsubseteq glb(X)$ for all lower bounds l of X. For the poset $\langle \mathbb{N}, \leq\rangle$, $glb(\mathbb{N}) = 0$ and $lub(\mathbb{N})$ does not exist.

Definition 6 (Lattice). A lattice $\langle S, \sqsubseteq, lub, glb\rangle$ is a poset $\langle S, \sqsubseteq\rangle$ such that any two elements $x, y \in S$ have a least upper bound $lub(\{x, y\})$ and a greatest lower bound $glb(\{x, y\})$.

A lattice is *complete* if any subset X of S has a least upper bound in S. A complete lattice has a least element called *bottom*, denoted $\bot$, and a greatest element, called *top*, denoted $\top$. All finite lattices are complete. If a lattice has a bottom and satisfies the ascending chain condition, it is a complete lattice.

A *fixpoint* of an operator $f : S \to S$ on a poset $\langle S, \sqsubseteq\rangle$ is an element $x \in S$ such that $f(x) = x$. The set of fixpoints of f is the set *Fixpoints*$(f) = \{x \in S | f(x) = x\}$. The set of pre-fixpoints is *pre-Fixpoints*$(f) = \{x \in S | x \sqsubseteq f(x)\}$ and the dual set of post-fixpoints is *post-Fixpoints*$(f) = \{x \in S | x \sqsupseteq$

$f(x)\}$. The set $\mathit{Fixpoints}(f) = \mathit{pre\text{-}Fixpoints}(f) \cap \mathit{post\text{-}Fixpoints}(f)$. The *least fixpoint* of f, denoted $\mathit{lfp}(f)$ is the least element of $\mathit{Fixpoints}(f)$ and the greatest fixpoint of f, denoted $\mathit{gfp}(f)$ is the greatest element of $\mathit{Fixpoints}(f)$.

Abstract interpretation [Cousot and Cousot 1977; Cousot 1978] provides a lattice theoretic framework for over-approximating set theoretic operations and was originally conceived for computing approximate solutions to program analysis problems. Let *Val* be the set of all possible values of a program variable and $\wp(\mathit{Val})$ the powerset of *Val*. Most program analysis problems involve computing the elements of $\wp(\mathit{Val})$ associated with some program locations. In particular, the solution is usually the fixpoint of an equation involving the program's variables. However, the fixpoint computation may not terminate, so an approximate solution must be constructed.

The *concrete domain* for the analysis is the set $\wp(\mathit{Val})$ of possible values, which forms a lattice with set inclusion, $\subseteq$, as its partial order. For any two sets $A, B \in \wp(\mathit{Val})$, $A \cup B$ is the least upper bound and $A \cap B$ is the greatest lower bound. Approximations are expressed using an *abstract domain Abs*, which also forms a lattice. The abstract domain is connected to $\wp(\mathit{Val})$ by an abstraction function $\mathit{abs} : \wp(\mathit{Val}) \rightarrow \mathit{Abs}$ and its dual, a concretisation function $\mathit{conc} : \mathit{Abs} \rightarrow \wp(\mathit{Val})$. The abstraction is *sound* if the set of values computed by an analysis in the abstract domain includes the set of values computed by an analysis in the concrete domain. This correspondence is formalised using a Galois connection.

Definition 7 (Galois Connection). Given two posets $\langle C, \sqsubseteq_C \rangle$ and $\langle A, \sqsubseteq_A \rangle$, a Galois connection is given by an abstraction function $\mathit{abs} : C \rightarrow A$ and concretisation function $\mathit{conc} : A \rightarrow C$ such that: $\forall c \in C, \forall a \in A : \mathit{abs}(c) \sqsubseteq_A a \Leftrightarrow c \sqsubseteq_C \mathit{conc}(a)$.

An element $c \in C$ of the concrete domain has a *minimal over-approximation* if there exists an element $a \in A$ of the abstract domain such that $c \sqsubseteq_C \mathit{conc}(a)$ and there exists no $a' \in A$ such that $a' \sqsubset_A a$ and $c \sqsubseteq_C \mathit{conc}(a')$. Minimal over-approximations may not always exist, but should be preferred to other over-approximations when they do. A *best over-approximation* is a minimal over-approximation that is unique. The best over-approximation of $c \in C$ is the greatest lower bound of the set of abstract over-approximations of c.

A choice of two methods exists to accelerate a fixpoint computation in the abstract domain. If a finite abstract domain is used, the abstract computation converges in finite time to an abstract fixpoint, which is guaranteed to exist. The alternative is to use an infinite abstract domain and define *widening* operators to accelerate convergence to an over-approximation of the fixpoint and enforce termination of a fixpoint computation. As successive applications of a widening operator result in imprecision, a *narrowing* operator is used to improve precision. For any given program, a finite abstract domain that provides the same precision as an infinite abstract domain with a widening operator can be found. However, for a family of programs, there may be no single finite domain, which provides the same precision as an infinite domain with a widening operator [Cousot and Cousot 1992c]. We use the definition of widening in [Cousot and Cousot 1977], which is sufficient for accelerating convergence and enforcing termination of a fixpoint computation. This definition is weaker than the widely used one of [Cousot and Cousot 1977;1992a], which is required when widening is additionally used to over-approximate least upper bounds.

Definition 8 (Widening). A widening operator on an abstract domain $\langle A, \sqsubseteq \rangle$ is a partial function $\triangledown : \wp(A) \nrightarrow A$ satisfying two properties:

Convergence Let S be an element of $\wp(A)$. If $\triangledown(S)$ is defined, then for any $s \in S, s \sqsubseteq \triangledown(S)$.

Termination For every increasing chain $y_0 \sqsubseteq y_1 \sqsubseteq \ldots$, the increasing chain defined as $y'_0 = y_0$, $y'_{i+1} = \triangledown(\{y_j | 0 \leq j \leq i\})$ stabilises after a finite number of terms.

The first property ensures that a computation with widening converges in the limit to an over-approximation of the least fixpoint of the computation. The second property ensures that successive application of a widening operator enforces termination. The imprecision introduced by widening is reduced by using narrowing. Once again, we use the weaker definition of narrowing in [Cousot and Cousot 1992b] than the more popular one of [Cousot and Cousot 1977], which is required when narrowing is used to refine an arbitrary over-approximation.

Definition 9 (Narrowing). A narrowing operator on an abstract domain $\langle A, \sqsubseteq \rangle$ is a partial function $\triangle : \wp(A) \nrightarrow A$ satisfying two properties:

Convergence Let S be an element of $\wp(A)$. If $\triangle(S)$ is defined, then $glb(S)$ exists and there exists $s \in S$ such that $glb(S) \sqsubseteq \triangle(S) \sqsubseteq s$.

Termination For every decreasing chain $y_0 \sqsupseteq y_1 \ldots$, the decreasing chain defined as $y'_0 = y_0$, $y'_{i+1} = \triangle(\{y'_j | 0 \leq j \leq i\})$ stabilises after a finite number of terms.

A narrowing operator is applied to a post-fixpoint and by the first property, results in a more precise post-fixpoint. In the limit, a computation with narrowing converges to a post-fixpoint. If the second property is satisfied, the computation with narrowing is terminates.

We emphasise that widening and narrowing are not dual concepts [Cousot and Cousot 1992b]. A widening operator as defined above is used to over-approximate the fixpoint of an increasing sequence. A narrowing operator is used to over-approximate the limit of a decreasing sequence thereby ensuring that an unknown fixpoint is not overshot. The dual of a widening operator ∇, is a *dual widening* operator $\overline{\nabla}$ that is used to accelerate convergence of a decreasing fixpoint computation to an under-approximation of the greatest fixpoint [Cousot 1978].

Definition 10 (Dual Widening). A dual widening operator on an abstract domain $\langle A, \sqsubseteq \rangle$ is a partial function $\overline{\nabla} : \wp(A) \nrightarrow A$ satisfying two properties:

Convergence Let S be an element of $\wp(A)$. If $\overline{\nabla}(S)$ is defined, then for any $s \in S, \overline{\nabla}(S) \sqsubseteq s$.

Termination For every decreasing chain $y_0 \sqsupseteq y_1 \sqsupseteq \ldots$, the decreasing chain defined as $y'_0 = y_0$, $y'_{i+1} = \overline{\nabla}(\{y'_j | 0 \leq j \leq i\})$ stabilises after a finite number of terms.

Dual widening, like widening introduces imprecision. A *dual narrowing* operator is used to improve the precision of the under-approximation. A dual narrowing widening operator is used to compute an under-approximation of the limit of an increasing sequence.

Definition 11 (Dual Narrowing). A dual narrowing operator on an abstract domain $\langle A, \sqsubseteq \rangle$ is a partial function $\underline{\triangle} : \wp(A) \nrightarrow A$ satisfying two properties:

Convergence Let S be an element of $\wp(A)$. If $\underline{\triangle}(S)$ is defined, then $lub(S)$ exists and there exists $s \in S$ such that $s \sqsupseteq \underline{\triangle}(S) \sqsupseteq lub(S)$.

Termination For every increasing chain $y_0 \sqsupseteq y_1 \ldots$, the increasing chain defined as $y'_0 = y_0$, $y'_{i+1} = \triangle(\{y'_j | 0 \leq j \leq i\})$ stabilises after a finite number of terms.

As we use the term fixpoint computation frequently, we say computation for fixpoint computation and computation with widening for a fixpoint computation with widening.

Chapter 3

A Framework for Widening with Automata

Our concern is over-approximating fixpoint computations involving sets encoded as regular or weak ω-regular languages and represented by finite automata or weak deterministic Büchi automata. In this chapter, we introduce and analyse a framework for widening such fixpoint computations. In Section 3.1 we describe in detail the class of fixpoint computations that we consider. The language accepted by a finite automaton or WDBA is extrapolated by merging states in the automaton. Our use and subsequent analysis of state merging techniques is influenced by algorithms for learning regular languages. We formalise the connection between fixpoint computations and computational learning theory in Section 3.2. We present a widening framework in Section 3.3 and analyse the solutions that can be computed within this framework in Section 3.4.

3.1 Fixpoint Computations and Automata

Consider a system that manipulates a set of variables. The domain of interest D is the set of all possible valuations of these variables. The initial values of the variables are described by a set $S_0 \subseteq D$ and the evolution of the system by a transition function $\mathcal{T} : \wp(D) \to \wp(D)$. The possible states of the system after the first transition are $\mathcal{T}(S_0)$. The set of states reached after $k > 0$ steps is $S_k = S_{k-1} \cup \mathcal{T}(S_{k-1})$. The sequence $S_0 \subseteq S_1 \subseteq \ldots$ is an *increasing sequence*, which converges to the set S_l such that $S_l \subseteq S_{l-1}$, where $l \in \mathbb{N} \cup \{\infty\}$. S_l is the least fixpoint of $\mathcal{T}$ starting from S_0 and represents the set of *reachable states* of the system. An alternative approach is to begin with the set $P_0 = D$ of all possible values. The set of states reached after $k > 0$ steps is $P_k = P_{k-1} \cap \mathcal{T}(P_{k-1})$. The sequence $P_0 \supseteq P_1 \supseteq \ldots$ is a *decreasing sequence*, which converges to the set P_g such that $P_g \supseteq P_{g-1}$, where $g \in \mathbb{N} \cup \{\infty\}$. P_g is the greatest fixpoint of $\mathcal{T}$ and is an over-approximation of S_l.

Computing a fixpoint of a transition function is an important step for many verification and static analysis problems. However, two main problems arise: (1) The sets of states S_i or P_i and in particular S_l and P_g may not have an efficient or finite representation. (2) Neither the least fixpoint nor the greatest fixpoint computation may terminate in finite time. We address the first problem by using automata to represent over-approximations of sets of states and the second problem by using widening.

Figure 3.1 illustrates the effect of widening. A least fixpoint computation begins from set S_0. The increasing sequence is computed by repeated application of the transition function. Convergence to a fixpoint is accelerated using widening by introducing an increment greater than what would be achieved by a single application of the transition function. Termination of the least fixpoint computation is enforced by extrapolating an element of the sequence to a post-fixpoint S_p such that $S_p \supseteq P_g$ where P_g is the greatest fixpoint of $\mathcal{T}$. The precision of the over-approximation is improved using narrowing to compute a value between the least fixpoint of the computation S_l and the over-approximation S_p.

Similarly, a greatest fixpoint computation begins from a set P_0 and proceeds as indicated. A dual widening operator (not shown) is used to accelerate this computation by introducing a large decrement. Termination is enforced by extrapolating an element of the sequence to a pre-fixpoint P_p such that $P_p \subseteq S_l$. A dual narrowing operator is used to compute a value between P_p and the greatest fixpoint P_g.

Let us return to the issue of representation. We say a set has a regular representation if it can be encoded as a regular language. We focus on fixpoint computations over structures that can be represented as finite automata or weak deterministic Büchi automata. Consider a transition function $\mathcal{T}$, and a set of initial states S_0. If finite automata are used as a representation, we make the following assumptions:

1. The set of initial states, S_0, is represented by a regular language L_0.
2. Given a set S_i represented by a regular language L_i, the set $\mathcal{T}^{=}(S_i)$ is also representable by a regular language L_{i+1}.

If weak deterministic Büchi automata are used as a representation, we assume that the initial states and the states reached after each transition have weak ω-regular representations. Note that the two assumptions imply that the set of states reached after finitely many steps also have a regular representation. However, we do not assume that the fixpoints of $\mathcal{T}$ are regular or weak ω-regular. In fact, there exist transition functions that satisfy these assumptions and have non-regular fixpoints.

Example 1. Consider a system with one integer valued variable x. The initial value of x is 1. The value of x after k steps is k^2. We encode the

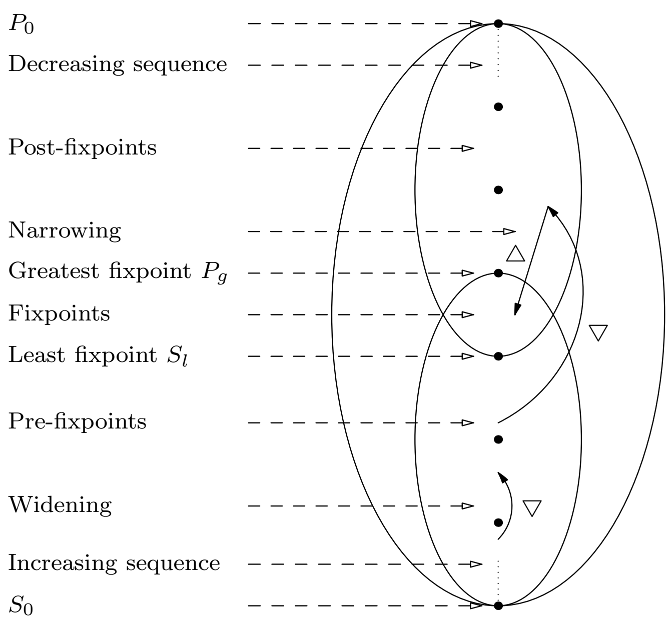


Figure 3.1: Over-approximating fixpoint computations with widening

value of x as a word over the alphabet $\Sigma = \{a\}$. A positive integer n is encoded as a^n and 0 is encoded as λ. The initial state of this system is represented by the language $\{a\}$, which is regular. The set of states of the system after k steps is represented by the language $\{a, a^4, \ldots, a^{k^2}\}$. The set of states reached after finitely many states is of finite cardinality and hence, has a regular representation. However, the fixpoint of this sequence is the language $\{a^{k^2} | k \geq 1\}$, which is not regular [Hopcroft and Ullman 1979, Example 3.2].

Widening is an extrapolation technique. A regular language is extrapolated by adding words to the language such that the language remains regular. We define widening operators that are used to partition the states of a finite automaton and merge states within the same partition. Merging states may increase the number of words accepted by an automaton and hence the language as well. This approach is directly applicable to weak deterministic Büchi automata as well, though we may have to perform a check to ensure that the automaton obtained by merging states is weak.

Our use and analysis of state merging techniques is influenced by algorithms for learning regular languages. We briefly review the learning theory models related to our problem.

3.2 Widening in Computational Learning Theory

Language identification in the limit or inductive inference, introduced by Gold [1967], is the problem of inferring a formal language from a sequence of examples, also called data. The examples may consist of words which are in the language, called *positive examples*, and words which are not, called *negative examples*. The data is *complete* if every word is eventually classified as being a positive or negative example. An inductive inference procedure is supplied with a sequence of examples from a language L and has to make a series of guesses or hypotheses $H_0, H_1, \dots$ where $H_j \subseteq \Sigma^*$ such that there exists some $k \geq 0$ with $H_{i+1} = H_i$ for all $i \geq k$. The procedure correctly infers a language L if the limit of the sequence of hypotheses is L. Note that the inference procedure only receives examples as input and cannot determine the correctness of a hypothesis as the examples may or may not conflict with the current conjecture. Inference of regular languages or finite automata is called *regular inference*.

Inductive inference is a passive process. In contrast, formal language learning, is an active process. A learner is provided with access to an oracle or a teacher and has to learn a formal language $L \subseteq \Sigma^*$ called the *target language*. An oracle is a machine, which answers restricted queries about the target language. If an oracle is provided, the learner updates the current hypothesis using answers to queries. Let L be the target language. We mention a few types of queries the learner can make and the answers that the oracle may provide. See [Angluin 1987b] for a complete survey.

- *Membership.* The input is a word $w \in \Sigma^*$. The answer is *yes* if $w \in L$ and *no* otherwise.
- *Equivalence.* The input is a language $H_i \subseteq \Sigma^*$. The answer is *yes* if $H_i = L$ and *no* if $H_i \neq L$. If the answer is *no*, either a positive example $w \in L \backslash H_i$ or a negative example $w' \in H_i \backslash L$ is returned.
- *Subset.* The input is a language $H_i \subseteq \Sigma^*$. The answer is *yes* if $H_i \subseteq L$ and *no* if $H_i \nsubseteq L$. If the answer is *no*, a word $w \in H_i \backslash L$ is returned.
- *Superset.* The input is a language $H_i \subseteq \Sigma^*$. The answer is *yes* if $H_i \supseteq L$ and *no* if $H_i \nsupseteq L$. If the answer is *no*, a word $w \in L \backslash H_i$ is returned.

Over-approximating the least-fixpoint of an increasing sequence is a variation of learning with superset queries using only positive examples. The target language L is a regular over-approximation of the least fixpoint. The interaction between the learner and the oracle is illustrated in Figure 3.2. The learner's initial guess is the language H_0 encoding the initial states of the system. At step i, the learner makes a query to the oracle with the current guess H_i. If some predefined criteria, *Criteria*$(\{H_0, \dots, H_i\})$, on the

```
LEARNER
    $H_0 \leftarrow$ initial state
    $i \leftarrow 0$
    repeat
        if $Criteria(\{H_0, \ldots, H_i\})$ then
            $H_{guess} \leftarrow \nabla(H_i)$
        else
            $H_{guess} \leftarrow H_i$
        end
        $\langle ans, H_{i+1} \rangle \leftarrow$ ORACLE($H_{guess}$)
        $i \leftarrow i + 1$
    until $ans =$ Yes
```

```
ORACLE($query$)
    $F \leftarrow query \cup \mathcal{T}(query)$
    if $F \subseteq query$ then
        $ans \leftarrow$ Yes
    else
        $ans \leftarrow$ No
    end
    return $\langle ans, F \rangle$
```

Figure 3.2: Widening as learning with superset queries

history of guesses are met, the learner applies a learning algorithm to generalise the current guess. We use a widening operator instead of a learning algorithm. The current guess is updated with the result of the query. This process continues until the oracle returns `Yes`. The oracle accepts a query as input and returns an answer, either `Yes` or `No`, and feedback F. The oracle contains a transition relation $\mathcal{T}$ and returns `No` if the query is a pre-fixpoint of $\mathcal{T}$. The possibly infinite sequence of sets in a fixpoint computation is the sequence of guesses that the learner makes. Similarly, computing an under-approximation of the greatest-fixpoint can be viewed as learning with subset queries and only negative examples.

The learning process we sketched is also related to inductive inference. As the oracle only returns positive examples, there is no way to determine if words have incorrectly been added to the language. Hence, an inference procedure that correctly identifies a class of regular languages from positive data can be used to design a widening operator that precisely computes a class of fixpoints. Similarly, any algorithm for learning a class of regular languages using superset queries can also be used to design widening operators that precisely compute a fixpoint. In addition, results about the conditions that must be satisfied for a type of inference procedure or learning algorithm to produce a correct result provide us with similar conditions that must be satisfied for the application of a widening operator.

We highlight three important differences between computing a regular over-approximation of the least fixpoint and regular inference or learning with superset queries: (1) In an inference or learning process, a single word is provided as input at each step. In a fixpoint computation, a possibly infinite set of words may be provided as input. As a result, it may be more difficult to identify a pattern in the given input sequence. (2) In regular inference and learning, the target language is regular and must be precisely computed. In a fixpoint computation, the target language may not be regular, but a non-trivial over-approximation suffices. We say non-trivial, because the language Σ^* is an over-approximation of every fixpoint. (3) Regular inference and learning algorithms may assume some properties about the data. For example, an algorithm might include the assumption that all words of length n from the target language appear before words of length $n + 1$ in the learning process. The data in our case is produced by a combination of a transition function and initial states, both of which are arbitrary. Therefore, we may not make any such assumptions.

Let us examine some useful results in regular inference and learning with queries. Gold [1967] proved that any set of languages containing all the finite languages and at least one infinite language cannot be learnt from positive data alone. This result applies to many interesting classes of languages, in particular the regular languages. However, it is not the case that no interesting languages can be identified from positive data. Angluin [1980b] characterised the set of languages that can be inferred from positive data alone and provided inference algorithms for subclasses of the regular languages called pattern languages [Angluin 1980a] and *reversible languages* [Angluin 1982]. A pattern is a concatenation of variable and constant symbols such as $1xx0$. The language of a pattern is the set of words that can be obtained by substituting variables in the pattern by a sequence of constant symbols. The words in the language of the pattern $1xx0$ include 110100 and 100000 but not 000 or 100. We now describe the class of k-reversible languages. Let k be a non-negative integer and L a regular language. L is k-reversible iff whenever $u_1vw \in L$ and $u_2vw \in L$ for some $u_1, u_2, v, w \in \Sigma^*$ with $|v| = k$, it holds that for every $z \in \Sigma^*$, $u_1vz \in L \Leftrightarrow u_2vz \in L$. An example of a zero-reversible language is the set of words over the alphabet $\{0, 1\}$ that contain an even number of 0's and 1's. Angluin's algorithm for inference of k-reversible languages only uses comparisons followed by state merging operations and provides us with a widening operator for this class of languages. Other regular inference algorithms for subclasses of regular languages that use only state merging operations also provide starting points for designing widening operators [Bierman and Feldman 1972; Miclet 1980].

Angluin [1978] and Gold [1978] independently showed that the problem of finding a deterministic automaton of minimum size that is compatible with two sets S^+ and S^- of positive and negative examples is NP-hard. Despite being intractable, the problem has a useful structure. If A is a prefix tree automaton accepting S^+, every automaton compatible with the sets S^+ and S^- can be obtained by merging states in A. In addition, if π_1 and π_2 are two partitions of the states of A such that $\pi_1 \preceq \pi_2$, then A/π_2 can be obtained by merging states in the automaton A/π_1. Thus, if A/π_1 accepts a negative example S^-, so will A/π_2. These results provide insights for our widening framework. A widening operator can be used to precisely compute a fixpoint A_∞ from an automaton A only if A_∞ is isomorphic to A/π for some partition π. If there are two different widening operators ∇_1

and $\bigtriangledown_2$ that induce the partitions π_1 and π_2 of the states of an automaton A such that $\pi_1 \preceq \pi_2$, any extrapolation introduced by π_1 is also introduced by π_2. Conversely, if π_1 introduces too much imprecision, so does π_2. Angluin and Smith [1983] discuss other techniques and applications in their survey and Murphy [1996] provides a survey of regular inference algorithms.

In contrast to regular inference, algorithms for learning with superset queries have received less attention. Angluin [1987b] provided an algorithm for learning pattern languages with superset queries. Angluin [1987a] showed that it is possible to learn any regular language using membership and equivalence queries if both positive and negative examples are available. We keep this in mind as we can obtain both positive and negative examples by combining least and greatest fixpoint computations.

The ω-regular languages have also received little attention. Maler and Pnueli [1995] extended Angluin's [1987a] algorithm for learning regular languages for learning weak deterministic Büchi automata using membership and equivalence queries. Saoudi and Yokomori [1994] provide an algorithm for learning two subclasses of ω-regular languages and Higuera and Janodet [2001] consider the problem of learning ω-regular languages using only finite prefixes. In Section 3.4 we characterise the set of weak deterministic Büchi automata computable within our widening framework using positive ω-examples as defined in [Maler and Pnueli 1995].

3.3 Widening for Fixpoint Computations with Automata

Let us examine fixpoint computations using automata. Let $\mathcal{A}$ be the set of finite automata, $A_0 \in \mathcal{A}$ be an automaton accepting words encoding the set of initial states and $\mathcal{T} : \mathcal{A} \to \mathcal{A}$ be the transition function. The sequence $A_0, A_1, \ldots, A_k$ with $L(A_{i+1}) = L(A_i) \cup L(\mathcal{T}(A_i))$ for $i \geq 0$ is an increasing sequence in a fixpoint computation. We use widening to accelerate the convergence of a fixpoint computation, to enforce termination, and to find regular approximations of non-regular fixpoints.

We present a framework for the design and application of widening operators. Let $\mathcal{U}$ be a universe of binary relations between two sets of states Q and Q' such that for all relations $\mathcal{R} \in \mathcal{U}$, $\mathcal{R} \subseteq Q \times Q'$. Let A_i and A_j be two automata with $j > i$ and let Q_i and Q_j denote the states of these automata. We use a set $\mathcal{S} \subseteq \mathcal{U}$ of binary relations to detect repeated patterns in and increments between A_i and A_j. The relations in $\mathcal{S}$ and the automaton A_i are used to construct an equivalence relation $\equiv_{\mathcal{S}}^{A_i} \subseteq Q_j \times Q_j$ that partitions the states of A_j. The automaton A_j is called the *widening candidate* and the *widened automaton* is the quotient $A_j / \equiv_{\mathcal{S}}^{A_i}$. The set $\mathcal{S}$ is called a *widening seed* and the equivalence relation $\equiv_{\mathcal{S}}^{A_i}$, a *widening equivalence.* As the relation $\equiv_{\mathcal{S}}^{A_i}$ may differ based on the choice of A_i, this automaton is called

```
1  Fixpoint Computation With Widening($\mathcal{T}, A_0$)
   Input: Transition function $\mathcal{T}$, Initial automaton $A_0$
   Data: Universe of binary relations $\mathcal{U}$

2  begin
3      $i \leftarrow 1$
4      $A_\triangledown \leftarrow A_0$
5      repeat
6          $A_i \leftarrow A_\triangledown \cup \mathcal{T}(A_\triangledown)$
7          $\mathcal{S} \leftarrow$ Select Seed($\{A_0, \ldots, A_i\}$)
8          $M \leftarrow$ Select Parameter($\{A_0, \ldots, A_i\}, \mathcal{S}$)
9          $\equiv_i \leftarrow$ Construct Equivalence($M, A_i, \mathcal{S}$)
10         $A_\triangledown \leftarrow A_i / \equiv_i$
11         $i \leftarrow i + 1$
12     until $L(A_i) \subseteq L(A_{i-1})$
13 end
```
Algorithm 1: Fixpoint computation with widening

the *widening parameter*. We denote the widening parameter as M to reduce visual clutter. To summarise, a widening seed and a widening parameter are used to construct a widening equivalence that partitions the states of an automaton called the widening candidate. A widened automaton is the quotient of an automaton with respect to a widening equivalence.

A scheme for computing least fixpoints using widening is codified in Algorithm 1. The algorithm takes the transition function and an automaton accepting an encoding of the set of initial states as input. The algorithm contains the universe of relations $\mathcal{U}$. Let $A_\triangledown$ denote the widened automaton. At each step $i \in \mathbb{N}$, a heuristic Select Seed is used to examine the history of the computation, $\{A_0, \ldots, A_i\}$, and select a widening seed $\mathcal{S}$. A heuristic Select Parameter is used to select a widening parameter M. The widening seed and parameter are used to construct an equivalence relation $\equiv_i$ that is used for widening. This process continues until an over-approximation of the least-fixpoint is reached.

Our task is now clear. We have to specify all details that are left abstract in this algorithm. We need to identify relations in the universe $\mathcal{U}$ and possible widening seeds. We say *identify* as opposed to *define* because a vast number of relations between the states of automata exist and can be used directly. Given a set of widening seeds, we need to define heuristics to select a widening seed and parameter. The guiding principle behind widening is to extrapolate at the cost of precision. A widening seed is chosen by examining the history of the computation and deciding how much to extrapolate. Recall that we use widening seeds to detect patterns between a

```
1  CONSTRUCT EQUIVALENCE ($M$, $A_i$, $\mathcal{S}$)
   Input  : A widening seed $\mathcal{S} \subseteq \mathcal{U}$
   Output: A widening equivalence $\equiv_{\mathcal{S}}$
2  begin
3      forall $\mathcal{R}_i \in \mathcal{S}$ do
4          Compute $\mathcal{R}_i$ between $M$ and $A_i$
5      end
6      $\mathcal{R}_\triangledown \leftarrow \emptyset$
7      forall $\mathcal{R}_i, \mathcal{R}_j \in \mathcal{S}$ do
8          $\mathcal{R}_\triangledown \leftarrow \mathcal{R}_\triangledown \cup \mathcal{R}_i \circ \mathcal{R}_j^{-1}$
9      end
10     $\equiv_{\mathcal{S}} \leftarrow \mathcal{R}_\triangledown^*$
11     return $\equiv_{\mathcal{S}}^M$
12 end
```

Algorithm 2: Construct a widening equivalence from a widening seed

pair of automata. The pattern detected varies with the automata chosen, hence we require a heuristic, SELECT PARAMETER, to select an appropriate widening parameter. Further, we require procedures to construct different equivalence relations and compute the quotient automaton.

We define Algorithm 2 for constructing an equivalence relation from a widening seed. The input to the algorithm is a widening seed $\mathcal{S} \subseteq \mathcal{U}$. In the loop between lines 4 and 6, a symmetric relation of the form $\mathcal{R}_\triangledown \subseteq Q' \times Q'$ is constructed using relations $\mathcal{R}_i, \mathcal{R}_j \subseteq Q \times Q'$ in $\mathcal{S}$. Note that $\mathcal{R}_\triangledown$ ranges over a different pair of states from the relations $\mathcal{R} \in \mathcal{S}$. The equivalence relation $\equiv_{\mathcal{S}}$ is the reflexive, transitive closure of $\mathcal{R}_\triangledown$. We use the following lemma to prove the correctness of the algorithm.

Lemma 1. *Given two relations $\mathcal{R}_1, \mathcal{R}_2 \subseteq Q_1 \times Q_2$, the relation $\mathcal{R} = (\mathcal{R}_2 \circ \mathcal{R}_1^{-1}) \cup (\mathcal{R}_1 \circ \mathcal{R}_2^{-1})$ is symmetric.*

Proof. Consider $\langle t, t' \rangle \in \mathcal{R}$. If $t = t'$, we are done. If $t \neq t'$, there are two cases.

$$\begin{aligned}
\langle t, t' \rangle \in \mathcal{R}_2 \circ \mathcal{R}_1^{-1} &\Rightarrow \exists r : \langle r, t \rangle \in \mathcal{R}_1 \wedge \langle r, t' \rangle \in \mathcal{R}_2 && [\text{definition of } \mathcal{R}_2 \circ \mathcal{R}_1^{-1}] \\
&\Rightarrow \langle t', t \rangle \in \mathcal{R}_1 \circ \mathcal{R}_2^{-1} && [\text{definition of } \mathcal{R}_1 \circ \mathcal{R}_2^{-1}] \\
&\Rightarrow \langle t', t \rangle \in \mathcal{R} && [\text{definition of } \mathcal{R}]
\end{aligned}$$

The case $\langle t, t' \rangle \in \mathcal{R}_1 \circ \mathcal{R}_2^{-1}$ is similar. □

Theorem 1. CONSTRUCT EQUIVALENCE$(M, A_i, \mathcal{S})$ *returns an equivalence relation.*

Proof. We need to show that $\mathcal{R}^*_\triangledown$ is an equivalence relation. $\mathcal{R}^*_\triangledown$ is reflexive and transitive by definition. We show that $\mathcal{R}^{k+1}_\triangledown$ is symmetric for any $k \geq 0$ by induction over k. $\mathcal{R}^0_\triangledown$ is the identity relation and is symmetric. Consider $\langle t, t' \rangle \in \mathcal{R}_\triangledown \circ \mathcal{R}^k_\triangledown$. For the induction hypothesis, assume that $\mathcal{R}^k_\triangledown$ is symmetric. Let $\langle t, t_k \rangle \in \mathcal{R}^k_\triangledown$ and $\langle t_k, t' \rangle \in \mathcal{R}_\triangledown$. By the induction hypothesis, $\langle t_k, t \rangle \in \mathcal{R}^k_\triangledown$. Observe that $\mathcal{R}_\triangledown = \bigcup_{\mathcal{R}_i, \mathcal{R}_j \in \mathcal{S}} (\mathcal{R}_j \circ \mathcal{R}_i^{-1}) \cup (\mathcal{R}_i \circ \mathcal{R}_j^{-1})$. By Lemma 1, $(\mathcal{R}_j \circ \mathcal{R}_i^{-1}) \cup (\mathcal{R}_i \circ \mathcal{R}_j^{-1})$ is symmetric, whereby, $\langle t', t_k \rangle$ and $\langle t_k, t' \rangle \in (\mathcal{R}_j \circ \mathcal{R}_i^{-1}) \cup (\mathcal{R}_i \circ \mathcal{R}_j^{-1})$ for some $\mathcal{R}_i, \mathcal{R}_j \in \mathcal{S}$. Hence, $\langle t_k, t' \rangle \in \mathcal{R}_\triangledown$ and $\langle t', t \rangle \in \mathcal{R}^k_\triangledown \circ \mathcal{R}_\triangledown$. By the associativity of relational composition, $\mathcal{R}^k_\triangledown \circ \mathcal{R}_\triangledown = \mathcal{R}_\triangledown \circ \mathcal{R}^k_\triangledown = \mathcal{R}^{k+1}_\triangledown$. □

A typical least fixpoint computation without widening is an instance of Algorithm 1. To see this, let $\mathcal{S}_\bot = \emptyset$ be a widening seed that is always returned by the heuristic SELECT SEED. At step i, the heuristic SELECT PARAMETER returns A_{i-1}. The algorithm CONSTRUCT EQUIVALENCE computes the reflexive, transitive closure of $\mathcal{S}_\bot$ and returns the identity relation. Given these specific heuristics, lines 5-12 of Algorithm 1 reduce to:

```
begin
    $i \leftarrow 1$
    repeat
        $A_i \leftarrow A_{i-1} \cup \mathcal{T}(A_{i-1})$
        $i \leftarrow i + 1$
    until $L(A_i) \subseteq L(A_{i-1})$
end
```

The widening seed $\mathcal{S}_\bot$ introduces no over-approximation. A complementary widening seed is $\mathcal{S}_\top = \{Q \times Q'\}$. CONSTRUCT EQUIVALENCE$(M, A_i, \mathcal{S}_\top)$ returns an equivalence relation, which at each step i in Algorithm 1, relates all states of A_i provided the state set of the widening parameter M is non-empty. The widening equivalence has index 1 and the widened automaton has only one state.

3.4 The Space of Solutions for Widening

The extrapolations introduced by different widening seeds form a spectrum between the two extremes defined by $\mathcal{S}_\bot$ and $\mathcal{S}_\top$. In this section, we characterise the solutions that can be computed within our widening framework and show that state based widening operators are extremely sensitive to the structure of an automaton.

3.4.1 Regular Languages

Let $\mathcal{Reg}$ denote the set of regular languages over the alphabet Σ. When equipped with the partial order $\subseteq$, we have the lattice of regular languages

$\langle \mathcal{Reg}, \subseteq, \cup, \cap \rangle$. For any two languages $L_1, L_2 \in \mathcal{Reg}$, the least upper bound is $L_1 \cup L_2$ and the greatest lower bound is $L_1 \cap L_2$. The bottom is the empty language $\emptyset$ and the top is the language Σ^*. We make the following observations about this lattice:

Remark 1. Every language $L \subseteq \Sigma^*$ has an over-approximation in $\mathcal{Reg}$.

Remark 2. Every regular language has a best approximation in $\mathcal{Reg}$.

Remark 3. The lattice $\langle \mathcal{Reg}, \subseteq, \cup, \cap \rangle$ is not complete.

Remark 4. Every non-regular language has no minimal over-approximation in $\mathcal{Reg}$.

For Remarks 1 and 2, observe that every language is over-approximated by Σ^* and that every regular language is best over-approximated by itself. Recall that every subset of elements of a complete lattice must have a least upper bound in the lattice. We show Remark 3 by identifying a set $S \subseteq \mathcal{Reg}$, which has no least upper bound in $\mathcal{Reg}$. Consider the infinite sequence of regular languages $L_0, L_1, \ldots$ such that L_i contains all words of length less than $i+1$ generated by a context free grammar. The set of languages $\{L_i\}_{i \geq 0}$ has no least upper bound in $\mathcal{Reg}$. To see that Remark 4 is true, consider any regular over-approximation L' of a non-regular language L. As regular languages are closed under set difference with finite sets, L' must contain infinitely many words not in L. A more precise, regular over-approximation of L can be obtained by removing any finite set of words in the set $L' \setminus L$ from L'. As the precision of *any* regular over-approximation of a non-regular language can be improved in this manner, non-regular languages have no minimal over-approximation in $\mathcal{Reg}$.

Remark 3 must be kept in mind because many existing results about fixpoints and abstract interpretation apply only to complete lattices. If a minimal, regular over-approximation does not exist, either another abstract representation must be used or a choice between various over-approximations must be made. Some possible courses of action are discussed in [Cousot and Cousot 1992b]. The choice can be guided by various factors such as knowledge of the system being verified and practical considerations.

Let us assume that a we have identified a unique, regular over-approximation of a fixpoint. Under what conditions can it be computed using widening? Dupont, Miclet and Vidal [1994] formulate regular inference as a search problem in a lattice of automata. We use a similar approach to study widening operators. Given an automaton in a fixpoint computation, the set of automaton that can be obtained by merging states form a lattice. A fixpoint can be computed precisely using widening only if the automaton representing the fixpoint is contained in this lattice.

Let π_1, π_2 denote partitions of the states of an automaton A. Define the preorder $\preceq$ between quotients of automata as $A/\pi_1 \preceq A/\pi_2$ iff $\pi_1 \preceq \pi_2$. Let *Quotients*(A) be the set of automata that can be obtained by merging states in A and $\pi_\top$ be the partition of states of A that has unit index. The set *Quotients*(A), with the preorder $\preceq$ defines a lattice with A as the bottom and $A/\pi_\top$ as the top elements. Let *Lattice*(A) denote this lattice.

Remark 5. For any automaton A, *Lattice*(A) is complete.

Remark 6. If $\pi_1 \preceq \pi_2$ are partitions of the states of A, *Lattice*$(A/\pi_2) \subseteq$ *Lattice*(A/π_1).

Remark 5 follows from *Lattice*(A) being finite. If $\pi_1 \preceq \pi_2$, A/π_2 belongs to the lattice *Lattice*(A/π_1), hence, Remark 6 follows.

In the sequel, let $A_\infty = (Q_\infty, \Sigma, \delta_\infty, t_0, F_\infty)$ be an over-approximation of a fixpoint that we would like to compute and $A = (Q, \Sigma, \delta, q_0, F)$ be an automaton in the fixpoint computation. We provide a sufficient condition for obtaining A_∞ by merging states of A. Dupont et al. [1994] provide a sufficient condition in the case that $L(A)$ is finite, which is a special case of our result.

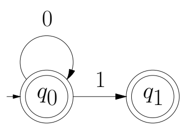
Given a language L, a set of words $S \subseteq L$ is called a *sample.* Our characterisation relies on *structurally complete* samples.

Definition 12 (Structural Completeness). A sample $S \subseteq \Sigma^*$ is structurally complete with respect to a trim automaton $A = (Q, \Sigma, \delta, q_0, F)$ iff

1. $S \subseteq L(A)$
2. For each $q \in F$, there exists $w \in S$ such that $q \in \delta^*(q_0, w)$.
3. For each $q, q' \in Q$ and $a \in \Sigma$ such that $q' \in \delta(q, a)$, there exists $w = uav \in S$ such that $q \in \delta^*(q_0, u)$.

A sample is S structurally complete with respect to an automaton A if the runs of A on the words in S exercise every transition in A and use every final state in A as an accepting state. Note that the second condition is not subsumed by the third because the run of A on a word may exercise every transition in A but only one state is used as the accepting state. A language L is trivially structurally complete with respect to every automaton accepting exactly L. Our definition differs slightly from that in [Dupont et al. 1994], where a structurally complete sample has to be finite. For every structurally complete sample $S \subseteq \Sigma^*$, there is a finite sample $S' \subseteq S$ that is also structurally complete.

Example 2. Consider the automaton A below:



A

The sample $\{001\}$ satisfies condition 3 but not condition 2 of Definition 12. The sample $\{000, 01\}$ is structurally complete for this automaton.

A sample S is *prefix complete* with respect to an automaton A iff $S = (Pre(S) \cap L(A))$. That is, every prefix of a word in S that is accepted by A is also in S. For a word $w = a_1a_2 \dots a_n$, let $w[i]$ denote the symbol a_i. Dupont et al. [1994] consider two representations for finite, structurally complete samples: maximal canonical automaton (MCA) and tries.

Definition 13 (Maximal Canonical Automaton). The Maximal Canonical Automaton (MCA) for a finite sample S is $MCA(S) = (Q, \Sigma, \delta, q_0, F)$ where $Q = \{q_0\} \cup \{q_{w[i]} | w \in S, 1 \leq i \leq |w|\}$, $F = \{q_{w[n]} | w \in S, n = |w|\}$ and $q_0 \in F$ if $\lambda \in S$ and $\delta(q_{w[i]}, a) = \{q_{w[i+1]} | w[i+1] = a\}$ is the transition relation.

$MCA(S)$ contains the words in S but does not additionally structure them. It is the largest, trim automaton accepting exactly S and is generally nondeterministic. The largest, deterministic automaton accepting exactly S is a *trie* or *prefix tree acceptor*, denoted $Trie(S)$.

Definition 14 (Trie). The trie for a finite, sample S is an automaton $Trie(S) = (Q, \Sigma, \delta, q_0, F)$, where $Q = Pre(S)$, $q_0 = \lambda$, $F = S$ and $\delta(w, a) = wa$ whenever $w, wa \in Q$.

Tries have been used to index and re`trie`ve strings [Knuth 1998], hence the name. Angluin [1982] used tries to develop a learning algorithm for the k-reversible languages. Let $CA(S)$ denote the canonical automaton representing the set S. For a finite sample S, $MCA(S)$, $Trie(S)$ and $CA(S)$ are structures of decreasing size accepting S. An MCA represents each word independently, a trie collects words with common prefixes and the canonical automaton, being minimal and deterministic additionally collects words with common suffixes.

We begin with a simple lower bound for obtaining A_∞ and then present the results in [Dupont et al. 1994].

Remark 7. If an automaton A has fewer states, final states or transitions than A_∞, then A_∞ cannot be obtained by merging states of A.

Theorem 2 (Dupont et al. 1994, Theorems 1 and 3). *A finite sample S is structurally complete with respect to an automaton A if and only if A belongs to $Lattice(MCA(S))$.*

Theorem 3 (Dupont et al. 1994, Theorem 2). *If S is a structurally complete, finite sample with respect to a minimal, deterministic automaton A, then A belongs to $Lattice(Trie(S))$.*

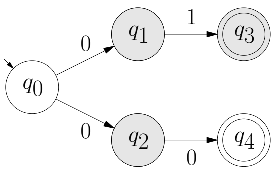
From Theorem 2 we know that, given an automaton A_∞, there exists an automaton accepting a finite language from which we can obtain A_∞ by merging states. From Theorem 3, we know that there even exists such an automaton, which is deterministic. We also see from Theorem 2 that *all* automata corresponding to a structurally complete sample S are contained in *Lattice*$(MCA(S))$. A trie for a sample S can be obtained by merging states in $MCA(S)$ that are reachable by the identical prefixes. Therefore, $Trie(S)$ and consequently, by Remark 6, *Lattice*$(Trie(S))$ are contained in *Lattice*$(MCA(S))$. Note that for an arbitrary sample S, neither $MCA(S)$ nor $Trie(S)$ may be minimal. In fact, it may be possible to merge states and obtain A_∞ from $MCA(S)$ and $Trie(S)$, but not from $CA(S)$.

Remark 8. There exist finite samples that are structurally complete with respect to an automaton A_∞ such that A_∞ belongs to *Lattice*$(MCA(S))$ and *Lattice*$(Trie(S))$ but not to *Lattice*$(CA(S))$.

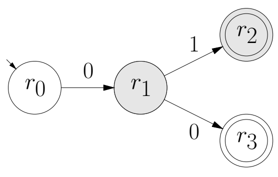
Remark 9. There exist automata A_1, A_2 and A_∞ such that A_∞ belongs to *Lattice*(A_1) and A_2 belongs to *Lattice*(A_1) but A_∞ does not belong to *Lattice*(A_2).

The following example is an illustration of Remark 8.

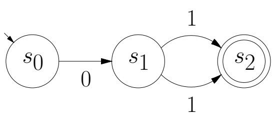
Example 3. Consider the automaton A_∞ below and a sample following automata accepting a sample $S = \{00, 01\}$, that is structurally complete with respect to A_∞.



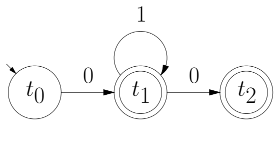
$MCA(S)$



$Trie(S)$



$CA(S)$



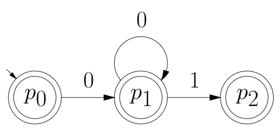
A_∞

Consider the partitions $\pi_1 = \{\{q_0\}, \{q_1, q_2, q_3\}, \{q_4\}\}$ of states of $MCA(S)$ and $\pi_2 = \{\{r_0\}, \{r_1, r_2\}, \{r_3\}\}$ of states of $Trie(S)$. The states in the same partitions are shaded in the illustration. The automata $MCA(S)/\pi_1$ and $Trie(S)/\pi_2$ are isomorphic to A_∞. However, there exists no such partition of the states of $CA(S)$.

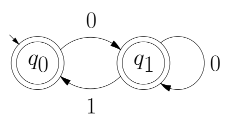
In Example 3, we can also see that $Trie(S)$ belongs to *Lattice*$(MCA(S))$ and that $CA(S)$ belongs to both *Lattice*$(MCA(S))$ and *Lattice*$(Trie(S))$ but as we have seen, A_∞ does not belong to *Lattice*$(CA(S))$. Hence, care must be taken when applying widening because the computation might move from a state in which a fixpoint be computed precisely to one in which this is no longer possible.

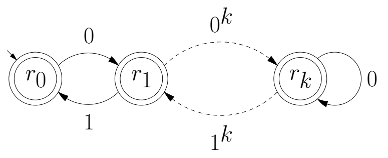
The ability to obtain a certain automaton by merging states changes dramatically if the structurally complete sample has infinitely many words. Given such a sample S for an automaton A_∞, we can obtain a trie or an MCA by unwinding transitions in $CA(S)$ a finite number of times to obtain an automaton A such that $L(A)$ is structurally complete with respect to A_∞. A problem arises as we do not know how many times the transitions in $CA(S)$ should be unwound. Formally, let A_1 be an automaton obtained by unwinding the transitions in the automaton $CA(S)$ accepting an infinite sample S. For any such A_1, we can pick A_2 obtained by another unwinding of $CA(S)$ such that A_∞ belongs to *Lattice*(A_2) but not to *Lattice*(A_1). We make this argument clear with an example.

Example 4. Consider a sample S accepted by $CA(S)$ below.



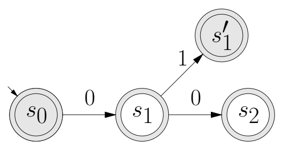
$CA(S)$

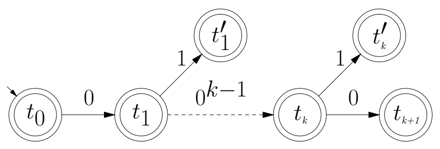




A_1 A_k

The sample S is structurally complete with respect to A_1. Further, there exists an infinite family of automata A_k, with $k \geq 1$, such that S is structurally complete with respect to each A_k. The transition $(p_1, 0, p_1)$ in $CA(S)$ can be unwound once to obtain the trie T_1 and k times to obtain the trie T_k. Both tries are shown below.





T_1 T_k

A_1 can be obtained from T_1 by merging s_1' with s_0 and s_2 with s_1. A_k can be obtained from T_k by merging t_i' with t_{i-1} for $0 \leq i \leq k$ and by merging t_{k+1} with t_k.

The automaton A_k in Example 4 belongs to *Lattice*(T_k). However, in order to obtain an automaton by merging states, we do not need to restrict ourselves to considering an MCA or a trie.

Theorem 4. *Let $A = (Q, \Sigma, \delta, r_0, F)$ be trim and deterministic and $A_\infty = (Q_\infty, \Sigma, \delta_\infty, t_0, F_\infty)$ be trim and deterministic. If $L(A)$ is structurally complete with respect to A_∞ and if for all $w \in L(A)$, $Pre(\delta^*(r_0, w)) \subseteq Pre(\delta_\infty(t_0, w))$, then A_∞ belongs to $Lattice(A)$.*

Proof. We construct a partition of the states of A and show that the quotient of A by this partition is isomorphic to A_∞. As $L(A)$ is structurally complete, for each $t \in Q_\infty$, there exists $w = uv \in L(A)$ such that $\delta_\infty(t_0, u) = t$. As A_∞ is deterministic, $Pre(t) \cap Pre(t') = \emptyset$ for each $t, t' \in Q_\infty$. By the condition of the theorem, $Pre(\delta^*(r_0, u)) \subseteq Pre(t)$, hence, for all $t' \neq t \in Q_\infty$, $Pre(\delta^*(r_0, u)) \cap Pre(t') = \emptyset$. As each state in A is reachable by a prefix of some state in A_∞, $|Q| \geq |Q_\infty|$. By structural completeness of $L(A)$ and a similar argument, we also have that $|F| \geq |F_\infty|$ and that A has at least as many transitions as A_∞. We have that A satisfies the lower bound of Remark 7.

Define a partition π as follows: For all $r, r' \in Q$, $[r]_\pi = [r']_\pi$ iff for some $t \in Q_\infty$, $Pre(r) \subseteq Pre(t)$ and $Pre(r') \subseteq Pre(t)$. The index of the partition is $|Q_\infty|$. Consider the quotient automaton A/π. A state $[r]_\pi$ in A/π is final iff there exists a state $r' \in [r]_\pi$ that is final, which in turn is possible iff the state $t \in Q_\infty$ corresponding to $[r]_\pi$ is final. So, we have that the number of final states in A/π is $|F_\infty|$. A similar argument applies for the number of non-final states in A/π. If there exists a transition between $t, t' \in Q_\infty$, by structural completeness, there exists a transition between the corresponding partitions $[r]_\pi$ and $[r']_\pi$. As both A_∞ and A are deterministic, for each $r_i, r_j \in [r]_\pi$, $[\delta(r_i, a)]_\pi = [\delta(r_j, a)]_\pi$. Therefore, if there exists a transition $([r]_\pi, a, [r']_\pi)$ in A/π, there exists a corresponding transition (t, a, t') in δ_∞. As A/π and A_∞ have the same number of states, final states and have the same transition relation, they are isomorphic. □

Observe that in an MCA and a trie, each state has exactly one prefix, so these automata always satisfy the conditions of the theorem. To prove Theorem 4, we constructed a partition of A using A_∞. In general A_∞ is not known and widening is used to choose and element of $Lattice(A)$. This lattice may be quite large and many different widening operators may be applicable. A merge between two states is *compatible* iff both states are final or both are non-final. If $L(A)$ is prefix complete, A_∞ can be obtained from A by compatible merges.

Corollary 1. *If A and A_∞ are automata satisfying the conditions of Theorem 4, and $L(A)$ is prefix complete, then A_∞ can be obtained from A using only compatible merges.*

Proof. Construct a partition of the states of A as in the proof of Theorem 4. All states in a partition have the same prefixes as a given state in A_∞. If this state is accepting, all states in the partition must be accepting and vice versa. □

Given an automaton A, the number of automata in *Lattice*(A) is the number of partitions of states of A. The number of partitions of a set with n elements is called the Bell number, denoted ϖ_n. The number $\varpi_n = \Theta(n/\log n)^n$ [Knuth 2005, Section 7.2.1.5]. The number ϖ_{100} has a 116 digits! Clearly, trying all the possibilities, even for small automata is not an option.

3.4.2 Weak ω-Regular Languages

We present similar results for weak ω-regular languages. We state a corollary of Theorem 4 that provides us with a condition for constructing a WDBA from a structurally complete sample. As structurally complete samples are defined in terms of the regular language accepted by an automaton, we introduce ω-samples and provide a sufficient condition, analogous to Theorem 4 in terms of the ω-language of a WDBA.

Let ω-$\mathcal{Reg}$ denote the set of weak ω-regular languages over the alphabet Σ. Equipped with the partial order $\subseteq$, we have the lattice $\langle \omega\text{-}\mathcal{Reg}, \subseteq, \cup, \cap \rangle$. The bottom is the empty language $\emptyset$ and the top is the language Σ^ω. As with regular languages, every ω-language has an over-approximation and every weak ω-regular language has a best over-approximation in ω-$\mathcal{Reg}$.

Remark 10. The lattice $\langle \omega\text{-}\mathcal{Reg}, \subseteq, \cup, \cap \rangle$ is not complete.

We show Remark 10 by identifying a set $S \subseteq \omega$-$\mathcal{Reg}$ that has no least upper bound in the lattice. Consider the infinite sequence of weak ω-languages $(0^2 1^2)^\omega$, $(0^2 1^2 + 0^3 1^3)^\omega, \ldots$. The limit of this sequence is an ω-context free language and is not contained in ω-$\mathcal{Reg}$.

Consider a sequence of weak deterministic Büchi automata $B_0, B_1, \ldots$ in a fixpoint computation with an over-approximation B_∞ that we wish to compute using widening. As Theorem 4 only requires that the two automata be trim and deterministic, we directly obtain a condition for merging states in B_i to obtain B_∞ in terms of the regular languages the two automata accept.

Corollary 2. *Let $B = (Q, \Sigma, \delta, r_0, F)$ and $B_\infty = (Q_\infty, \Sigma, \delta_\infty, t_0, F_\infty)$ be trim, weak and deterministic Büchi automata. If $L(B)$ is structurally complete with respect to $L(B_\infty)$ and if for all $w \in L(B)$, $Pre(\delta^*(r_0, w)) \subseteq Pre(\delta^*(t_0, w))$, B_∞ belongs to Lattice(B).*

Maler and Pnueli [1995] define an ω-observation table for weak ω-regular languages. We use a similar idea to define ω-samples. An *ω-sample* is a set of pairs $S \subseteq \Sigma^* \times \Sigma^\omega$ where for $(u, \beta) \in S$, β is ultimately periodic. For $u \in \Sigma^*$ and $\beta \in \Sigma^\omega$, we write $u \in S$ if there exists $\alpha \in \Sigma^\omega$ such that $(u, \alpha) \in S$ and we write $\beta \in S$ if there exists $v \in \Sigma^*$ such that $(v, \beta) \in S$. The set of ω-words in an ω-sample S, *words*$(S) = \{u\beta | (u, \beta) \in S\}$. We provide conditions for constructing a WDBA B_∞ from another WDBA B in terms of structurally complete ω-samples.

Definition 15 (ω-Structural Completeness). An ω-sample S is structurally complete with respect to a weak deterministic Büchi automaton $B = (Q, \Sigma, \delta, r_0, F)$ iff the following conditions hold:

1. The set $words(S) \subseteq L_\omega(B)$.
2. For each $q, q' \in Q$ and $a \in \Sigma$ such that $q' \in \delta(q, a)$, there exists $(ua, \beta) \in S$ such that $q \in \delta^*(q_0, u)$.
3. For each recurrent state $q \in F$, there exists $(u, \beta) \in S$ such that for the run ρ of B on $u\beta$, $q \in Inf(\rho)$.

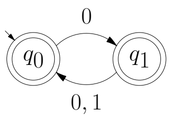
A structurally complete ω-sample for WDBA B contains words accepted by a B factored into a finite prefix and an ultimately periodic suffix. If condition 2 is satisfied, the prefixes in the ω-sample exercise every transition in B. By condition 3, the runs of B on the words constructed from an ω-sample visit every recurrent final state infinitely often. Recall that finite structurally complete samples are used to construct maximal canonical automata, tries and minimal automata, which are acyclic. Likewise, we construct a WDBA accepting an ω-sample.

Definition 16. A transition graph for an ω-sample S, $G(S) = (Q, \Sigma, \delta, q_0, \emptyset)$, is an automaton with $Q = \cup_{u \in S} Pre(u)$ where $u \in \Sigma^*$, $\delta(u, a) = ua$, if there exists $u \in \Sigma^*, \beta, \beta' \in \Sigma^\omega$ such that$(u, \beta), (ua, \beta) \in S$ and $\delta(u, a) = v$ if for all $\beta \in \Sigma^\omega$, $ua\beta \in words(S) \Leftrightarrow v\beta \in words(S)$ and $q_0 = \lambda$.

The WDBA accepting an ω-sample S, $B(S) = (Q, \Sigma, \delta, q_0, F)$ is a transition graph with $F = \{q \in Inf(\alpha, G(S)) | \alpha \in words(S)\}$.

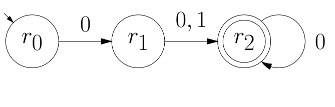
As with finite automata, ω-structural completeness of a WDBA with respect to a representation of the fixpoint is insufficient to guarantee that the fixpoint can be computed using widening.

Example 5. The WDBA B_∞ below accepts infinite words such that the symbol 1 only appears in even positions.

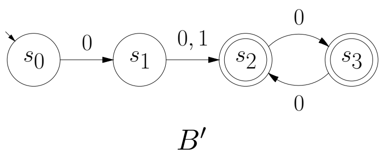


B_∞

The ω-sample $S = \{(01, 0^\omega), (000, 0^\omega)\}$ is structurally complete with respect to B_∞. Consider the two WDBAs $B(S)$ and B', which accept $words(S)$. $B(S)$ is constructed as indicated in Definition 16.



$B(S)$



B'

Observe that B_∞ belongs to *Lattice*(B') but not to *Lattice*$(B(S))$.

The problem we see is similar to that in the case of finite automata. An ultimately periodic ω-word accepted by B_∞ may be accepted by B but may traverse a cycle that has fewer states. As we only modify B by merging states, there is not way to increase the number of states in a cycle. Notice that the automaton $B(S)$ is minimal but B' is not. We see once again that minimising a deterministic automaton *before* widening, dramatically alters the precision of the solution that can be obtained. As states in a WDBA can be uniquely identified by their prefixes, we obtain a similar characterisation to the finite automata case.

Theorem 5. *Let $B = (Q, \Sigma, \delta, r_0, F)$ and $B_\infty = (Q_\infty, \Sigma, \delta_\infty, t_0, F_\infty)$ be trim, weak and deterministic Büchi automata. If $L(B)$ can be factored into a structurally complete ω-sample for to B_∞ and if for all $w \in Pre(L_\omega(B))$, $Pre(\delta^*(r_0, w)) \subseteq Pre(\delta_\infty(t_0, w))$, then, there exists $B_\pi \in Lattice(B)$ such that $L_\omega(B_\infty) = L_\omega(B_\pi)$.*

Proof. Let S be the sample obtained by factoring words $\alpha \in L_\omega(B)$ into finite prefixes and ultimately periodic suffixes. Define a partition π of Q as follows: For all $r, r' \in Q$, $[r] = [r']$ iff for some $t \in Q_\infty$, $Pre(r) \subseteq Pre(t)$ and $Pre(r') \subseteq Pre(t)$. Denote $B_\pi = B/\pi = (Q_\pi, \Sigma, \delta_\pi, [r_0], F_\pi)$. Following a similar reasoning as in the proof of Theorem 4 we can show that $|Q_\infty| = |Q_\pi|$ and that δ_π is isomorphic δ_∞. It remains to identify F_π. If $t \in F_\infty$ is recurrent, there exists $\alpha \in L_\omega(B)$ such that $t \in Inf(\alpha, B_\infty)$. By the definition of structural completeness and the restriction on prefixes, there exists $r \in Inf(\alpha, B)$ such that $Pre(r) \subseteq Pre(t)$. As B is a WDBA, this state is final, so $[r]$ is also final. Hence, for every recurrent, final state in F_∞, there exists a recurrent, final state in B_π. As transient states can be visited at most once, they do not affect the ω-language accepted by an automaton. As B_∞ and B_π have the same number of states, the same transition relation and the same recurrent final states, $L(B_\infty) = L(B_\pi)$. □

There is a subtle difference between the results in Theorems 4 and 5. In Theorem 4, we provided a sufficient condition for A_∞ to be in *Lattice*(A) for some A. In Theorem 5, we only identify $B_\pi \in Lattice(B)$ such that $L(B_\pi) = L(B_\infty)$. The automata B_π and B_∞ are not necessarily isomorphic.

Chapter 4

Widening Seeds

The most important step in a fixpoint computation with widening is selecting the widening operator. In our framework, a widening operator is defined by a combination of a widening seed and a widening parameter. The widening seed defines the nature of extrapolation introduced and the widening parameter is used to tune the extrapolation to determine the structure of the final automaton. In this chapter, we introduce and study different widening seeds.

4.1 A Universe of Relations

Widening seeds are subsets of a universe $\mathcal{U}$ of binary relations between states of automata. Various state-based preorders exist in the literature and are used to check language inclusion between automata, construct abstractions of transition systems and for minimisation. Such work is devoted to studying how states in an automaton can be merged *without* changing the accepted language. The study of pre-orders and equivalences for various kinds of transition systems such as process algebras [Baeten and Weijland 1990], timed automata [Dembiński et al. 2002] and hybrid automata [Henzinger et al. 2005] with respect to the logical properties that they preserve has also received much attention. In contrast, the use of preorders to merge states to increase the language accepted by an automaton has been studied only recently [Dams et al. 2001; Boigelot et al. 2003; Bartzis and Bultan 2004]. We use existing preorder relations to define widening seeds and study the extrapolation the resulting widening operators introduce.

We begin by identifying useful elements of the universe $\mathcal{U}$. Common preorders for automata use language based or transition based criteria. In Table 4.1, we list a few relations that use language based criteria to compare states. States with the same suffixes are related by $=_s$ and states with the same prefixes are related by $=_p$. Given a state r, define $Pre_k(r) = \{w \in \Sigma^* | w \in Pre(r) \wedge |w| \leq k\}$ as the prefixes of r of length at most k and

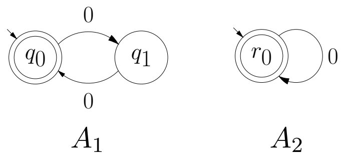
Relation	Definition		
$=_s$	$r{=_s}t$	$\Leftrightarrow$	$\mathit{Suff}(r) = \mathit{Suff}(t)$
$=_p$	$r{=_p}t$	$\Leftrightarrow$	$\mathit{Pre}(r) = \mathit{Pre}(t)$
$=_p^k$	$r{=_p^k}t$	$\Leftrightarrow$	$\mathit{Pre}_k(r) = \mathit{Pre}_k(t)$
$=_s^k$	$r{=_s^k}t$	$\Leftrightarrow$	$\mathit{Suff}_k(r) = \mathit{Suff}_k(t)$
$\subseteq_p$	$r{\subseteq_p}t$	$\Leftrightarrow$	$\mathit{Pre}(r) \subseteq \mathit{Pre}(t)$
$\subseteq_s$	$r{\subseteq_s}t$	$\Leftrightarrow$	$\mathit{Suff}(r) \subseteq \mathit{Suff}(t)$
$\cap_p$	$r{\cap_p}t$	$\Leftrightarrow$	$\mathit{Pre}(r) \cap \mathit{Pre}(t) \neq \emptyset$
$\cap_s$	$r{\cap_s}t$	$\Leftrightarrow$	$\mathit{Suff}(r) \cap \mathit{Suff}(t) \neq \emptyset$

Table 4.1: Relations for comparing states

$\mathit{Suff}_k(r) = \{w \in \Sigma^* | w \in \mathit{Suff}(r) \wedge |w| \leq k\}$ as the suffixes of r of length at most k. The relations $=_s^k$ and $=_p^k$ relate states with the same suffixes or prefixes of length at most k. The first four relations in Table 4.1 are equivalence relations. If every prefix of a state r is a prefix of a state t, they are in the relation $\subseteq_p$. Similarly, if the suffixes of r are suffixes of t, the states are related by $\subseteq_s$. These two relations are preorders. The symmetric relations $\cap_p$ and $\cap_s$ relate states that have a common prefix or suffix. We use the infix notation, as in Table 4.1, to denote that two states are in a certain relation.

Preorders using language containment or equivalence are easy to define but may be expensive to compute. In particular, determining language inclusion between nondeterministic automata is PSPACE hard. In contrast, simulation relations are more restrictive but can be computed efficiently. Simulation relations have been studied extensively in concurrency theory [Milner 1995] to understand the branching behaviour of concurrent processes and have been used in verification tools for state space reduction [Dill et al. 1992]. The classic definition of simulation proposed by Milner [1971] is referred to as *ordinary simulation*. There exists an ordinary simulation between a state r and a state t (t o-simulates r), denoted $r \sqsubseteq^o t$, iff for every transition from r to a state r' with a symbol a, there is a transition from t to a state t' with a symbol a such that t' o-simulates r'. If there exists an ordinary-simulation between r and t, for every run from the state r on a word w, there exists a run from t' on w. Acceptance criteria are not considered for ordinary simulation. Dill et al. [1992] introduced *direct simulation* (originally called safety simulation), an extension of ordinary simulation for check language inclusion between finite automata and Büchi automata. There exists a direct simulation between a state r and a state t (t di-simulates r), denoted $r \sqsubseteq^{di} t$, iff for every transition from r to a state r' with a symbol a, there is a transition from t to a state t' with a symbol a such that t' di-simulates r' *and* if r is a final state, then t should be a final state. Direct simulation between the initial states of two automata implies language inclusion [Dill et al. 1992]. We illustrate the difference between the two relations with an example.

Example 6 *(Ordinary and Direct Simulation).* Consider the automata A_1 and A_2 below.



Every run from the states q_0 and q_1 is defined from the state r_0 and vice versa. Therefore, we have that $q_0 \sqsubseteq^o r_0$ and $r_0 \sqsubseteq^o q_0$. In addition, $q_0 \sqsubseteq^{di} r_0$ and $q_1 \sqsubseteq^{di} r_0$ but $r_0 \not\sqsubseteq^{di} q_0$. As $L(A_1) \subseteq L(A_2)$ but $L(A_2) \not\subseteq L(A_1)$, only direct simulation is a sound condition with respect to language inclusion.

Direct simulation is sufficient for checking language inclusion between finite automata but may be too strong for checking language inclusion between Büchi automata. In Example 6, the ω-language accepted by A_1 and A_2 is the same but $r_0 \not\sqsubseteq^{di} q_0$. Dill et al. [1992] make this observation and propose a *live cycles Büchi simulation relation* (BSR-lc) for a subclass of Büchi automata. We refer to the class of simulation relations defined for Büchi automata as ω-simulations. Note that direct simulation is also an ω-simulation. Henzinger et al. [2002] use a game theoretic framework to study the simulation relations for various ω-automata and introduce *fair simulation*, which uses acceptance criteria specific to Büchi automata. There exists a fair simulation between states t and r in a Büchi automaton (t f-simulates r), denoted $r \sqsubseteq^f t$, iff t o-simulates r and for every infinite accepting, run from r, the corresponding infinite run from t is also accepting. In the automata in Example 6, every infinite run from r_0 and q_0 is accepting, hence, $q_0 \sqsubseteq^f r_0$ and $r_0 \sqsubseteq^f q_0$. It also holds that $q_1 \sqsubseteq^f r_0$ and vice versa. In the example, merging the states q_0 and q_1 does not change the ω-language of the automaton.

Etessami et al. [2005] highlight that in general, merging states that fair simulate each other does not preserve the ω-language of the automaton and propose *delay simulation.* Recall that if ρ is an infinite run, $\rho(i)$ is the i^{th} state visited in the run. There exists a delay simulation between states t and r in a Büchi automaton (t de-simulates r), denoted $r \sqsubseteq^{de} t$, iff t o-simulates r and for every infinite run ρ from r, if $\rho(i)$ is a final state, there exists $j \geq i$ such that, in the corresponding infinite run τ from t, $\tau(j)$ is a final state. By definition, for every accepting run from r, there exists a corresponding accepting run from t. A formal definition of the simulation relations discussed follows.

Definition 17 (Simulation Relations). Consider two automata $A_1 = (Q_1, \Sigma, \delta_1, r_0, F_1)$ and $A_2 = (Q_2, \Sigma, \delta_2, t_0, F_2)$. Ordinary, direct, fair and delay simulations are binary relations $\sqsubseteq^o$, $\sqsubseteq^{di}$, $\sqsubseteq^f$ and $\sqsubseteq^{de} \subseteq Q_1 \times Q_2$, defined as:

1. *Ordinary*: $r \sqsubseteq^o t \Leftrightarrow \forall r' \in Q_1, \forall a \in \Sigma : r' \in \delta(r, a) \Rightarrow \exists t' \in Q_2 : t' \in \delta(t, a) \wedge r' \sqsubseteq^o t'$.
2. *Direct*: $r \sqsubseteq^{di} t \Leftrightarrow (r \in F_1 \Rightarrow t \in F_2) \wedge (\forall r' \in Q_1, \forall a \in \Sigma : r' \in \delta(r, a) \Rightarrow \exists t' \in Q_2 : t' \in \delta(t, a) \wedge r' \sqsubseteq^{di} t')$.
3. *Fair*: $r \sqsubseteq^f t \Leftrightarrow r \sqsubseteq^o t$ and for all words $\alpha \in \Sigma^\omega$, and an infinite run ρ in A_1 on α such that $\rho(0) = r \wedge \mathit{Inf}(\rho) \cap F_1 \neq \emptyset$, there exists a run τ of A_2 in α such that $\tau(0) = t$ and $\mathit{Inf}(\tau) \cap F_2 \neq \emptyset$.
4. *Delay*: $r \sqsubseteq^{de} t \Leftrightarrow r \sqsubseteq^o t$ and for all words $\alpha \in \Sigma^\omega$, and an infinite run ρ in A_1 on α such that $\rho(0) = r$, there exists an infinite run τ in A_2 on α such that $\tau(0) = t$ and for all $i \in \mathbb{N}$, $\rho(i) \in F_1 \Rightarrow \exists j \geq i : \tau(j) \in F_2$.

Delay simulation is more restrictive than fair simulation as non-accepting runs must also satisfy a condition involving final states. Etessami et al. [2005] prove that delay simulation implies language containment and order different ω-simulations by inclusion.

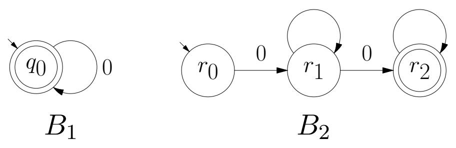
Theorem 6 (Etessami et al. 2005, Proposition 3). *Consider ordinary, direct, fair and delay simulations for Büchi automata.*

1. *For sim $\in \{o, di, f, de\}$, the simulation relation $\sqsubseteq^{sim}$ is a preorder.*
2. *The simulation relations are ordered by containment. That is, $r \sqsubseteq^{di} t \Rightarrow r \sqsubseteq^{de} t \Rightarrow r \sqsubseteq^f t \Rightarrow r \sqsubseteq^o t$.*
3. *For sim $\in \{di, f, de\}$, if $r \sqsubseteq^{sim} t$, then $\mathit{Suff}_\omega(r) \subseteq \mathit{Suff}_\omega(t)$.*

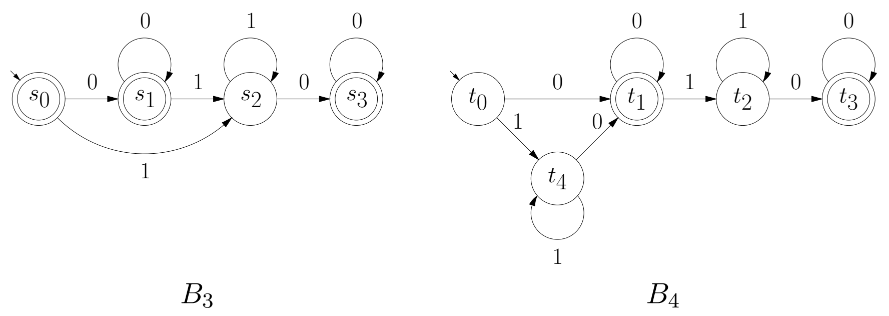
How do these results interest us? Just as we use preorders and equivalence relations to relate states in a finite automaton, we can use ω-simulations to design widening operators for WDBAs. If preorders that differ in the number of state they relate are defined, we can construct widening operators with different precision. Results like those in Theorem 6 provide insights into which states are merged and allow us to compare the effect of different widening operators.

Much work on ω-simulation relations is aimed at efficiently minimising Büchi automata. Etessami et al. [2005] prove that unlike fair simulation, merging two states that delay simulate each other does not change the ω-language of the Büchi automaton and provide examples where delay simulation differs from fair and direct simulation. Gurumurthy et al. [2002] provide an example in which no two states delay simulate each other but states that fair simulate each other can be merged without changing the language of the automaton. These examples show that it is worthwhile to study different ω-simulation relations and that no ω-simulation is in general better than others. The examples in existing studies only feature Büchi automata that are not weak, which begs the question: Which results about ω-simulation relations hold for WDBAs? If the different relations behave identically with weak automata, it is sufficient to consider only one. Do the differences between ω-simulation relations exist even if only weak automata are considered? We provide a positive answer to this question.

Example 7 *(ω-simulations and WDBAs).* We compare delay simulation with direct and fair simulation.



For the WDBAs B_1 and B_2, $q_0 \sqsubseteq^{de} r_0$ but $q_0 \not\sqsubseteq^{di} r_0$. In terms of the ω-language accepted by B_1 and B_2, we have that $L_\omega(A_1) \subseteq L_\omega(A_2)$. Thus, direct simulation is not a necessary condition for language inclusion between weak Büchi automata. Next, we compare fair and delay simulation.



For the weak deterministic automata B_3 and B_4, $L_\omega(B_3) \subseteq L_\omega(B_4)$ and $s_0 \sqsubseteq^{f} t_0$. Consider the runs ρ and τ of B_3 and B_4 on the word 1^ω. As s_0 is final and t_0 is not and for all $i > 0$, $\rho(i)$ and $\tau(i)$ are not final, the condition for delay simulation is not satisfied and $s_0 \not\sqsubseteq^{de} t_0$.

We conclude that fair, delay and direct simulation behave differently with WDBAs and may each be useful for designing widening operators. Lynch and Vaandrager [1995] survey other simulation relations for labelled transition systems and Bustan and Grumberg [2004] compare other ω-simulations.

We call the relations we have presented so far *elementary*. A widening seed is a set of relations. Every elementary relation can be used as a widening seed. We provide a brief overview and example of the analysis we undertake for each widening seed.

Let us recall some terminology and notation. A widening candidate A is an automaton in a fixpoint computation. A widening parameter M is an automaton such that $M = A$ or M occurs before A in the computation. A widening seed, $\mathcal{S} \subseteq \mathcal{U}$, is a set of binary relations. A widening equivalence is a relation $\equiv_{\mathcal{S}}^{M}$ on the states of A constructed from M and $\mathcal{S}$ using Algorithm 2, Construct Equivalence. The widened automaton A_{∇} is $\left(A/\equiv_{\mathcal{S}}^{M}\right)$. If the fixpoint of the computation is regular, we denote it as A_{∞}.

We analyse three properties of a widening seed.

1. Extrapolation: Do there exist A and M such that $L(A) \subset L(A_{\nabla})$?
2. Termination: For all computations, do there exist A and M such that the computation with widening terminates?
3. Precision: If the fixpoint is regular, under what conditions does the computation with widening converge to the exact fixpoint.

We say a widening seed is extrapolating if there exist A and M such that $L(A) \subset L(A_{\nabla})$. Results about precision should be interpreted carefully as a computation with widening that converges to the precise fixpoint may not do so in a finite number of steps. We demonstrate the style of our analysis and elaborate on the previous statement with a simple example. Let $\mathcal{S}_{\perp}$ be the widening seed.

Lemma 2 (Properties of $\mathcal{S}_{\perp}$). *$\mathcal{S}_{\perp}$ has the following properties:*

1. *It is not extrapolating.*
2. *It does not enforce termination.*
3. *If the fixpoint is regular, the computation with widening converges to the precise fixpoint.*

Proof. (1) For any choice of M and A, the widening equivalence $\equiv_{\mathcal{S}}^{M}$ is the identity relation, hence, $A_{\nabla} = A$.
(2) Any computation with widening is identical to the computation without and does not terminate.
(3) The computation trivially converges to the fixpoint. The computation with widening, being identical to the one without, also converges to the precise fixpoint. □

In the sequel, we repeat a similar analysis for each widening seed we consider.

4.2 Extrapolation and Convergence Criteria

In this section, we identify general conditions that are useful for proving extrapolation and convergence properties of widening seeds. We use the notation introduced in the previous section. Let A be the automaton $(Q, \Sigma, \delta, r_0, F)$ and A_∞ be an automaton representing the fixpoint. To compare the language of A and A_∇, we need to examine the quotient of A with respect to the widening equivalence. We show that it is sufficient to study the effect of merging two states in A and provide criteria that must be satisfied for the language accepted by the quotient of a finite automaton or weak deterministic Büchi automaton to increase.

Define the language between two states $L(r, r') = \{w \in \Sigma^* | r' \in \delta^*(r, w)\}$ to be the set of words by which r' is reachable from r. Let S be a set of states. The set of words $L(S) = \bigcup_{t,t' \in S} L(t, t')$. For the remainder of this section, let π denote a partition of index $|Q| - 1$ obtained by merging exactly two states in A and let $A/\pi = (Q_\pi, \Sigma, \delta_\pi, [r_0], F_\pi)$ be the quotient of A with respect to π. For $r \in Q$, $[r]$ denotes both the partition of r in π and the state corresponding to r in Q_π. Observe that any partition π' of Q of index $|Q| - n$ with $0 \leq n \leq |Q| - 1$ can be obtained from the trivial partition $\pi_\perp$ of Q via a sequence of partitions $\pi_\perp = \pi_0 \preceq \pi_1 \preceq \ldots \preceq \pi_n = \pi'$ such that the index of any two successive partitions differs by 1. To determine if $L(A) \subset L(A/\pi')$, it suffices to determine if $L(A/\pi_i) \subset L(A/\pi_{i+1})$ for some $0 \leq i < n$. Let t, t' be the two states in the same block of π and r, with subscripts as required, range over the other states in Q.

4.2.1 Finite Automata

We provide a necessary and sufficient condition for extrapolating the language of a DFA. As transitions are not added in the quotient construction, each transition in A/π corresponds to a transition in A. If $L(A) \subset L(A/\pi)$, there exists a word $w \in L(A/\pi) \setminus L(A)$ such that either the sequence of transitions exercised in A/π by reading w is not defined in A or the run of A/π on w ends in a final state but the run of A does not. A word $w \in L(A/\pi) \setminus L(A)$ can be written as $u_1 \cdots u_n$, where for each u_i, there exists a sequence of transitions respecting δ in A corresponding to the sequence of transitions in A/π.

Consider the prefixes and suffixes of a state $[t]$ in A/π. The set $\mathit{Pre}([t])$ includes the prefixes of t and t' as well as all words in $L(\{t, t'\})$ Similarly, $\mathit{Suff}([t])$ includes the suffixes of t and t' and the words in $L(\{t, t'\})$. The change in the language of the quotient automaton can be described in terms of $\mathit{Pre}([t])$ and $\mathit{Suff}[t])$.

Remark 11. $L(A) \subset L(A/\pi) \Leftrightarrow \mathit{Pre}(\{t, t'\}) \cdot L(\{t, t'\})^* \cdot \mathit{Suff}(\{t, t'\}) \nsubseteq L(A)$

Proof. $\Rightarrow$: If $L(A) \subset L(A/\pi)$, there exists a word $w \in L(A/\pi) \setminus L(A)$. If the run of A/π on w does not visit $[t]$, it is also defined in A, so the run must visit the state $[t]$ at least once. Hence, $w \in \mathit{Pre}([t]) \cdot \mathit{Suff}([t])$. The set of words $\mathit{Pre}([t]) \cdot \mathit{Suff}([t])$ consists of prefixes of t and t', concatenated

with zero or more repetitions of words between any pair of states in $\{t, t'\}$, concatenated with the suffixes of t and t'. This is precisely the set of words $\mathit{Pre}(\{t, t'\}) \cdot L(\{t, t'\})^* \cdot \mathit{Suff}(\{t, t'\})$, so this direction follows.
$\Leftarrow$: From the previous case, we may rewrite the right hand side of the remark as $\mathit{Pre}([t]) \cdot \mathit{Suff}([t]) \not\subseteq L(A)$. Consider a word $w \in \mathit{Pre}([t]) \cdot \mathit{Suff}([t]) \setminus L(A)$. Clearly, $w \in L(A/\pi) \setminus L(A)$. □

If the automaton A is deterministic, we can make a stronger statement.

Lemma 3. *If A is deterministic, $L(A) \subset L(A/\pi) \Leftrightarrow \mathit{Suff}(t) \neq \mathit{Suff}(t')$.*

Proof. $\Leftarrow$: We show the equivalent statement that $\mathit{Suff}(t) \neq \mathit{Suff}(t') \Rightarrow L(A/\pi) \backslash L(A) \neq \emptyset$. As $\mathit{Suff}(t) \neq \mathit{Suff}(t')$, at least one of the following conditions must hold:

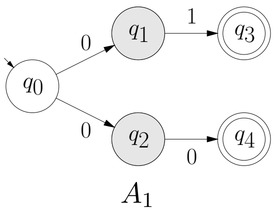
(a) $\mathit{Suff}(t) \backslash \mathit{Suff}(t') \neq \emptyset$

(b) $\mathit{Suff}(t') \backslash \mathit{Suff}(t) \neq \emptyset$.

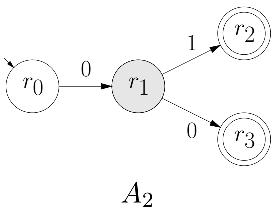
Without loss of generality, say (a) holds and $v \in \mathit{Suff}(t) \setminus \mathit{Suff}(t')$. As A is deterministic, $\mathit{Pre}(t) \cap \mathit{Pre}(t') = \emptyset$. As we only consider the reachable states in an automaton, $\mathit{Pre}(t) \neq \emptyset$ and $\mathit{Pre}(t') \neq \emptyset$, so there must exist $u \in \mathit{Pre}(t') \setminus \mathit{Pre}(t')$. The word $uv \in L(A/\pi) \backslash L(A)$, which is not empty as required.
$\Rightarrow$: To prove $L(A) \subset L(A/\pi) \Rightarrow \mathit{Suff}(t) \neq \mathit{Suff}(t')$, we may equivalently prove: $\mathit{Suff}(t) = \mathit{Suff}(t') \Rightarrow L(A) \supseteq L(A/\pi)$, which reduces to proving that $\mathit{Suff}(t) = \mathit{Suff}(t') \Rightarrow L(A) = L(A/\pi)$ as A cannot accept more words than A/π. As merging states with the same suffixes does not increase the language of the automaton, the implication follows. □

Providing a similar characterisation for NFAs is not as straightforward. States with different suffixes may have the same prefixes, so merging states in an NFA may not change the language of the automaton.

Example 8. The automaton A_2 below is obtained from A_1 by merging the states r_1 and r_2. Note that $L(A_1) = L(A_2)$.

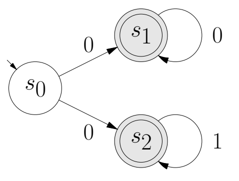


A_1

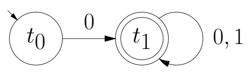


A_2

The automaton A_4 below is obtained from A_3 by merging the states s_1 and s_2 but $L(A_3) \neq L(A_4)$.



A_3



A_4

To provide a similar result to Lemma 3 for nondeterministic automata, we only provide conditions that are satisfied if $L(A) \subset L(A/\pi)$. We first consider the case that $L(\{t, t'\} = \emptyset$.

Lemma 4. *If $L(\{t, t\}) = \emptyset$ and $L(A) \subset L(A/\pi)$, one of the following conditions must hold:*

1. *$Pre(t) \setminus Pre(t') \neq \emptyset \wedge Suff(t') \setminus Suff(t) \neq \emptyset$*
2. *$Pre(t') \setminus Pre(t) \neq \emptyset \wedge Suff(t) \setminus Suff(t') \neq \emptyset$*

Proof. Consider $w \in L(A/\pi) \backslash L(A)$. As any run of A/π that does not visit the state $[t]$ is also defined in A, the run of A/π on w must visit $[t]$. By the antecedent of the lemma, the run of A/π on w can visit the state $[t]$ only once, so we can write w as uv, where $u \in Pre(t) \cup Pre(t')$ and $v \in Suff(t) \cup Suff(t')$. We consider two cases:

(1) Let $u \in Pre(t')$. If $v \in Suff(t')$, $uv \in L(A)$, hence, $v \in Suff(t) \setminus Suff(t')$. If $u \in Pre(t)$, $uv \in L(A)$, hence $u \in Pre(t') \setminus Pre(t)$, satisfying Condition 2.

(2) Let $u \in Pre(t)$, by a symmetric argument, it must be that $u \in Pre(t) \setminus Pre(t')$ and $v \in Suff(t') \setminus Suff(t)$, satisfying Condition 1. $\square$

If the set $L(t, t')$ is not empty, There exists a sequence of transitions from $[t]$ to $[t]$ for every word in $L(\{t, t'\})^*$.

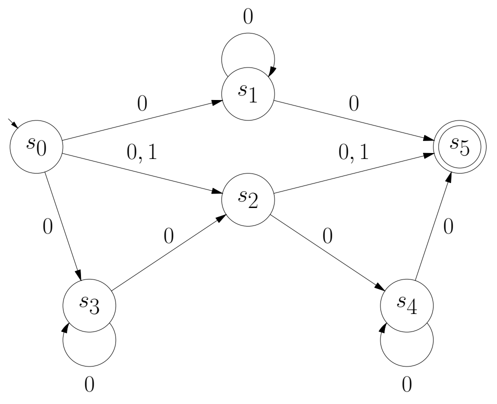
Lemma 5. *If $L(t, t) = L(t', t') = \emptyset$ and $L(t, t') \neq \emptyset$ and $L(A) \subset L(A/\pi)$, the following must hold: $Pre(t') \setminus Pre(t) \neq \emptyset \wedge Suff(t) \setminus Suff(t') \neq \emptyset$.*

Proof. Consider $w \in L(A/\pi) \backslash L(A)$. We show by contradiction that if the consequent of the lemma does not hold, $w \in L(A)$. The word w is of the form uxv where $u \in Pre(t) \cup Pre(t')$, $v \in Suff(t) \cup Suff(t')$ and $x = y_1 \cdots y_k$, where for each $1 \leq i \leq k$, $y_i \in L(t, t')$. Consider two conjuncts in the lemma.
(1) Assume $Pre(t') \setminus Pre(t) = \emptyset$, which implies that $Pre(t') \subseteq Pre(t)$. For all $y \in L(t, t')$, $Pre(t) \cdot y \subseteq Pre(t')$. Therefore, if $Pre(t') \subseteq Pre(t)$, it must be that $Pre(t) \cdot L(t, t')^* \subseteq Pre(t)$. Consider the word uxv with $x = y_1 \cdots y_k$. If $v \in Suff(t)$, as $ux \in Pre(t)$, $uxv \in L(A)$. If $v \in Suff(t')$, as $uy_1 \cdots y_{k-1} \in Pre(t)$ and $y_k \in L(t, t')$, $ux \in Pre(t')$ and $uxv \in L(A)$, yielding a contradiction.
(2) Assume $Suff(t) \setminus Suff(t') = \emptyset$, which implies that $Suff(t) \subseteq Suff(t')$. For all $y \in L(t, t')$, $y \cdot Suff(t') \subseteq Suff(t)$. Therefore, if $Suff(t) \subseteq Suff(t')$, it

must be that $L(t,t')^* \cdot \mathit{Suff}(t) \subseteq \mathit{Suff}(t')$. Consider again the word uxv with $x = y_1 \cdots y_k$. If $u \in \mathit{Pre}(t')$, as $xv \in \mathit{Suff}(t')$, $uxv \in L(A)$. If $u \in \mathit{Pre}(t)$, as $uy_1 \in \mathit{Pre}(t')$ and $y_2 \cdots y_k \cdot v \in \mathit{Suff}(t')$, $uxv \in L(A)$, yielding a contradiction. □

The case with only $L(t',t)$ not being empty is identical. It remains to identify the conditions that must be satisfied if t and t' are merged and $L(t,t) \neq \emptyset$. In this case, we find that the conditions we identified in Lemmas 4 and 5 are insufficient.

Example 9. Consider the states s_1 and s_2 in the automaton A below.



A

For the state s_1, $\mathit{Pre}(s_1) = \mathit{Suff}(s_1) = 0^+$. For the state s_2, $\mathit{Pre}(s_2) = (1 + 0^+)$ and $\mathit{Suff}(s_2) = (1 + 0^+)$. We see that $\mathit{Pre}(s_1) \subseteq \mathit{Pre}(s_2)$ and $\mathit{Suff}(s_1) \subseteq \mathit{Suff}(s_2)$, so the conditions of the form in Lemmas 4 and 5 do not hold. However, merging s_1 and s_2 adds the words 10^+1 to the language of A.

Lemma 6. *If $L(\{t,t'\}) = L(t,t) \neq \emptyset$ and $L(A) \subset L(A/\pi)$, one of the following conditions must hold:*

1. $\mathit{Pre}(t) \setminus \mathit{Pre}(t') \neq \emptyset \wedge \mathit{Suff}(t') \setminus \mathit{Suff}(t) \neq \emptyset$
2. $\mathit{Pre}(t') \setminus \mathit{Pre}(t) \neq \emptyset \wedge \mathit{Suff}(t) \setminus \mathit{Suff}(t') \neq \emptyset$
3. $$\begin{array}{ll} & \mathit{Pre}(t) \subseteq \mathit{Pre}(t') \\ \wedge & \mathit{Suff}(t) \subseteq \mathit{Suff}(t') \\ \wedge & \exists w \in L(t,t)^* : \mathit{Suff}(\delta(t',w)) \neq \mathit{Suff}(t') \end{array}.$$

Proof. Consider a word $w \in L(A/\pi) \setminus L(A)$. This word must be of the form uxv, where $u \in \mathit{Pre}([t]) \setminus L([t],[t])$, $x \in L([t],[t])$ and $v \in \mathit{Suff}([t]) \setminus L([t],[t])$. If $u \in \mathit{Pre}(t)$, as $ux \in \mathit{Pre}(t)$, it must be that $v \in \mathit{Suff}(t')$. As $uxv \notin L(A)$, it must be that $ux \notin \mathit{Pre}(t')$, hence the first condition holds. If $u \in \mathit{Pre}(t')$, there are two possibilities. If $v \in \mathit{Suff}(t)$, and either

$v \notin \mathit{Suff}(t')$ or $xv \notin \mathit{Suff}(t')$ the second condition holds. If $v \in \mathit{Suff}(t')$, there are two possibilities. If $\delta^*(t', x)$ is not defined, for all $z \in \mathit{Suff}(t)$, $xz \notin \mathit{Suff}(t')$ and the second condition holds. If $\delta^*(t', x)$ is defined, it must be that $v \notin \mathit{Suff}(\delta^*(t', x))$, and the third condition holds. □

We have identified conditions that must be satisfied if the language of an automaton is extrapolated by merging exactly two states. On arriving at these conditions, we discovered that the problem is of interest for developing NFA minimisation algorithms. Let us state the problem in our setting. If π is the partition of the states of an NFA, under what conditions is $L(A) = L(A/\pi)$? Alternatively, which widening operators are non-extrapolating for NFAs? Ilie and Yu [2002] prove that if either the relation $=_p$ or the relation $=_s$ is used to merge states, the language of the automaton does not change. Champarnaud and Coulon [2004] suggest the use of the preorders for merging states and claim that merging two states t and t' that only satisfy the negation of conditions 1 and 2 in Lemma 6 does not change the language accepted. However, this claim has since been retracted [Champarnaud and Coulon 2005]. The partition of the states of A in Example 9 refutes this claim. Ilie et al. [2005] identified three conditions that correspond to the negation of the conditions in terms of the prefixes and suffixes of states that are merged and also provide an example refuting the claim of Champarnaud and Coulon [2004], which only differs from Example 9 in the labels of the transitions.

4.2.2 Weak Deterministic Büchi Automata

We identify extrapolation criteria for weak deterministic Büchi automata. If we restrict ourselves to a normal form for WDBAs introduced by Löding [2001], we obtain a fairly straightforward characterisation of when states can be merged without changing the language.

Definition 18 (k-Colouring). Let $B = (Q, \Sigma, \delta, r_0, F)$ be a WDBA and $k \in \mathbb{N}$ be an even number. A k-colouring is a function $c : Q \to \{0, \ldots, k\}$ such that

1. $c(r)$ is even for recurrent states $r \in F$.
2. $c(r)$ is odd for recurrent states $r \notin F$
3. For all $r, r' \in Q$ and $a \in \Sigma$ with $\delta(r, a) = r'$, $c(r) \leq c(r')$.

A k-colouring c is *maximal* iff for any k-colouring c' and for every state $r \in Q$, it holds that $c'(r) \leq c(r)$. Observe that a run ρ of a WDBA is accepting iff $\max\{c(q) | q \in \mathit{Inf}(\rho)\}$ is even. Given a k-colouring c, let $F_c = \{q \in Q | c(q) \text{ is even }\}$.

Definition 19 (Coloured Normal Form). A WDBA B with final states F is in coloured normal form iff there exists a maximal k-colouring of B and $F = F_c$.

Löding [2001] proved that for any maximal k-colouring, two states in a WDBA with the same ω-suffixes have the same colour. In addition, if the automaton is in coloured normal form and two states have the same finite suffixes, they have the same ω-suffixes. See Example 7 for examples of automata in coloured normal form.

Lemma 7 (Löding 2001, Lemma 7). *Let c be a maximal k-colouring of a WDBA B. For all states $q, q' \in Q$, if $\mathit{Suff}_\omega(q) = \mathit{Suff}_\omega(q')$, then $c(q) = c(q')$.*

Lemma 8 (Löding 2001, Lemma 10). *Let B be a WDBA in coloured normal form. For any states $q, q' \in Q$, if $\mathit{Suff}(q) \neq \mathit{Suff}(q')$, then $\mathit{Suff}_\omega(q) \neq \mathit{Suff}_\omega(q)$.*

We may infer from these two lemmas that if a WDBA B is in coloured normal form, merging states that do not have the same suffixes extrapolates the ω-language accepted.

Corollary 3. *Let B be a WDBA in coloured normal form and t, t' be two states in the same block of a partition π. We have that $\mathit{Suff}(t) \neq \mathit{Suff}(t') \Leftrightarrow L_\omega(B) \subset L_\omega(B/\pi)$.*

Proof. $\Rightarrow$: If $\mathit{Suff}(t) \neq \mathit{Suff}(t')$, by Lemma 8, $\mathit{Suff}_\omega(t) \neq \mathit{Suff}_\omega(t')$. Without loss of generality say $\beta \in \mathit{Suff}_\omega(t') \setminus \mathit{Suff}_\omega(t)$. As B is deterministic, there exists $u \in \mathit{Pre}(t) \setminus \mathit{Pre}(t')$. The word $u\beta \in L_\omega(B/\pi) \setminus L_\omega(B)$, hence $L_\omega(B) \subset L_\omega(B/\pi)$.

$\Leftarrow$: Assume that $L_\omega(B) \subset L_\omega(B/\pi)$ but $\mathit{Suff}(t) = \mathit{Suff}(t')$. By Lemma 3, $\mathit{Suff}(t) = \mathit{Suff}(t') \Rightarrow L(B) = L(B/\pi)$. Further, if two states accept the same language, they also accept the same ω-language, so we have $L_\omega(B) = L_\omega(B/\pi)$, yielding a contradiction, so this direction follows. $\square$

We make a few observations that are useful for identifying extrapolations. Let $B = (Q, \Sigma, \delta, r_0, F)$ be a WDBA in coloured normal form with a maximal k-colouring c. Let π be a partition of index $|Q| - 1$ and t and t' be two states in the same block of π.

Remark 12. If $t \in F$ and $t' \notin F$, $L_\omega(B) \subset L_\omega(B/\pi)$.

Remark 13. If $t, t' \in F$ and there exists $uv \in \Sigma^*$ such that $\delta(t, u) = r$, $\delta(r, v) = t'$ and $r \notin F$, $L_\omega(B) \subset L_\omega(B/\pi)$.

Remark 12 follows from Lemma 8 as $\mathit{Suff}(t) \neq \mathit{Suff}(t')$ if only one of the states t, t' is final. To see Remark 13, consider the colour of the states t, t' and r. By conditions 1 and 2 of Definition 18, $c(t)$ and $c(t')$ are even and $c(r)$ is odd. By condition 3 of the same definition, we have that $c(t) < c(r) < c(t')$. From Lemma 7, if $c(t) \neq c(t')$, $\mathit{Suff}_\omega(t) \neq \mathit{Suff}_\omega(t')$, so we have $L_\omega(B) \subset L_\omega(B/\pi)$.

At this juncture, we highlight an important point about merging states in a WDBA. Unlike finite automata, weak Büchi automata are not closed under the quotient operation. We provide a necessary and sufficient condition for the quotient of a WDBA to be a WDBA.

Lemma 9. *Let $B = (Q, \Sigma, \delta, r_0, F)$ be a WDBA and π be a partition of index $|Q| - 1$, with the states t and t' in the same block. Let Q_{merge} be the set of states $\{r \in Q | \exists uv \in \Sigma^* : \delta^*(t, u) = r \wedge \delta^*(r, v) = t'\}$. The Büchi automaton B/π is weak iff $(Q_{merge} \setminus \{t, t'\}) \subset F$ or $(Q_{merge} \setminus \{t, t'\}) \cap F = \emptyset$.*

Proof. Let $Q_1, \ldots, Q_m$ be the SCCs of B. Define the partial order $\leq$ between SCCs as $Q_i \leq Q_j$ iff there exists $r \in Q_i$, $r' \in Q_j$ and $w \in \Sigma^*$ such that $\delta(r, w) = r'$. Let Q_t and $Q_{t'}$ denote the SCCs of the states t and t'. There are three possibilities:

1. $Q_t \leq Q_{t'}$ and $Q_{t'} \leq Q_t$
2. $Q_t < Q_{t'}$ or $Q_{t'} < Q_t$
3. Q_t and $Q_{t'}$ are incomparable.

We consider each possibility in turn.

1. If t and t' are in the same SCC, B and B/π have the same number of SCCs. As t and t' have the same accepting status, all states in each SCC of $[t]$ in B/π also have the same accepting status and B/π is weak.
2. Without loss of generality say $Q_t < Q_{t'}$. Consider the SCC of $[t]$ in B/π. The states in this SCC are all states that can be visited in a run that begins in t and ends in t' and are the states in Q_{merge}. If B/π is weak, all these states should have the same accepting status. As t and t' are merged, if either state is final, $[t]$ is final, so it suffices to check if all the other states in Q_{merge} have the same accepting status.
3. If Q_t and $Q_{t'}$ are incomparable, there is one less SCC in B/π than in B as $Q_t \cup Q_{t'}$ is one SCC. Again, it is sufficient to check if all states except t and t' in this SCC have the same accepting status.

$\square$

It follows from part (1) of the proof above that if each block of a partition only includes states in the same SCC, the quotient Büchi automaton is also weak. If the conditions of Lemma 9 are not satisfied, we can choose not to compute the quotient automaton. However, if this situation occurs too frequently, a widening operator may not be applicable often enough. The alternative is to compute the quotient and make it weak by marking all states $[r]$ such that $r \in Q_{merge}$ as final states of B/π.

Definition 20 (Weak Quotient). Let $B = (Q, \Sigma, \delta, q_0, F)$ be a weak Büchi automaton and π be a partition of Q. The weak quotient $B /\!\!/ \pi = (Q_\pi, \Sigma, \delta, [q_0], F_\pi)$ is the Büchi automaton with $Q_\pi = \{[q] | q \in Q\}$ and for all $[q] \in Q_\pi$, $a \in \Sigma$, $\delta([q], a) = \{[q'] | q \in \delta(q, a)\}$. Let $Q_1, \dots, Q_m$ be the SCCs of the transition graph of $B /\!\!/ \pi$. $F_\pi = \cup_i Q_i$ where for each i, there exists $[r] \in Q_i$ such that $r \in F$.

Remark 14. The weak quotient of a weak Büchi automaton is weak.

Remark 15. For any partition π, $L_\omega(B) \subseteq L(B /\!\!/ \pi)$.

4.2.3 Convergence Criteria

We have identified necessary criteria for a widening operator to be extrapolating. Another important question relates to convergence of a fixpoint computation with widening. As we see in subsequent sections, most widening operators do not enforce termination for all fixpoint computations. In such cases, especially if the widening operator is extrapolating, we are interested in the fixpoint of the computation with widening. Specifically, under what conditions does a fixpoint computation with widening converge to the precise fixpoint. In general, the answer depends on the widening seed used to define the widening operator. In this section, we identify a general condition for precise convergence of a computation with widening.

As we frequently require a condition relating the prefixes of A to those of A_∞, we define it here.

Definition 21 (Prefix property). Let A be an automaton in a fixpoint computation with a regular fixpoint represented by a minimal, deterministic automaton A_∞. A state r in A has the prefix property with respect to A_∞ iff there exists t in A_∞ such that $\mathit{Pre}(r) \subseteq \mathit{Pre}(t)$. The automaton A has the prefix property if every state in A has the prefix property.

Recall from Theorem 4 that $A_\infty \in \mathit{Lattice}(A)$ only if $L(A)$ is structurally complete with respect to A_∞ and has the prefix property. If A is deterministic and has the prefix property, for every state q in A, there is a unique state t in A_∞ such that $\mathit{Pre}(q) \subseteq \mathit{Pre}(t)$. We denote this state $f(q)$.

Remark 16. Let $A_0, A_1, \dots$ be the automata in a computation with widening with a regular fixpoint A_∞, such that A_i is either $\mathcal{T}^{=}(A_{i-1})$ or $\left(A_{i-1}/\equiv_{\mathcal{S}}^{A_j}\right)$ for some widening seed A_j with $j \leq i$. If for each A_i, A_j, $L\left(A_i/\equiv_{\mathcal{S}}^{A_j}\right) \subseteq L(A_\infty)$, then the fixpoint computation with widening converges to the precise fixpoint.

Proof. If for each automaton A_i in the fixpoint computation with widening, $L(A_i) \subseteq L(A_\infty)$, then, the sequence converges to the least fixpoint. This statement can be proved by induction over the steps in the computation. For the base case, consider the initial automaton A_0. By definition of a fixpoint computation, $L(A_0) \subseteq L(A_\infty)$. For the induction hypothesis, assume that for any $i-1$, $L(A_{i-1}) \subseteq L(A_\infty)$. We need to show that $L(A_i) \subseteq L(A_\infty)$. If $A_i = \mathcal{T}^{=}(A_{i-1})$, we are done. It remains to prove that $L\left(A_i / \equiv_{\mathcal{S}}^{A_j}\right) \subseteq L(A_\infty)$ for any A_i and A_j, which is the statement of the remark. $\square$

The statement in Remark 16, is essentially a common step that is required to prove a statement about the convergence for various widening operators. Another common step we require involves identifying partitions such that the language of the quotient automaton is contained in the fixpoint of the computation.

Lemma 10. *Let π be a partition of the states of an automaton A and A_∞ be a deterministic automaton representing a regular fixpoint. If $L(A) \subseteq L(A_\infty)$ and for any two states q, q' in the same block of π the prefix property holds and $f(q) = f(q')$, then $L(A/\pi) \subseteq L(A_\infty)$.*

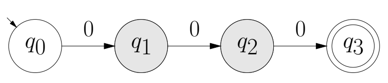
Proof. The partition π is of index $|Q| - n$ for $0 \leq n \leq |Q| - 1$ and can be written as a sequence $\pi_\perp = \pi_0 \preceq \ldots \preceq \pi_n = \pi$, where consecutive partitions differ in index by 1. The proof is by induction over the language accepted by the quotient automaton A/π_i. For the base case, consider π_0. Clearly, if $L(A) \subseteq L(A_\infty)$, then $L(A/\pi_\perp) \subseteq L(A_\infty)$. For the induction hypothesis, assume that for $i < n$, $L(A/\pi_i) \subseteq L(A_\infty)$. We show that merging two state in A/π_i satisfying the conditions of the lemma does not extrapolate the language beyond the fixpoint.

Denote the two states that are merged in A/π_i as q, q' and the corresponding state in A/π_{i+1} as $[q]$. From Remark 11, it is sufficient to examine $\mathit{Pre}([q]) \cdot \mathit{Suff}([q])$. Note that if a state r in A/π_i has the prefix property with respect to A_∞, as $\mathit{Pre}(r) \subseteq \mathit{Pre}(f(r))$ and $L(A/\pi_i) \subseteq L(A_\infty)$ by assumption, it must be that $\mathit{Suff}(r) \subseteq \mathit{Suff}(f(r))$. Consider a word $uxv \in \mathit{Pre}([q]) \cdot \mathit{Suff}([q])$ such that $u \in \mathit{Pre}([q]) \setminus L([q],[q])$, $x \in L([q],[q])$ and $v \in \mathit{Suff}([q]) \setminus L([q],[q])$. From the previous observation, we have that $v \in \mathit{Suff}(f(r))$. We show that $ux \in \mathit{Pre}(f(r))$.

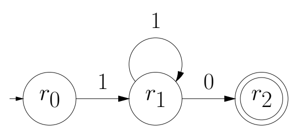
Without loss of generality, say $u \in \mathit{Pre}(q)$. Consider $x = y_1 \ldots y_k$, where each $y_k \in L(\{q, q'\})$. For each y_i, for some $q_1, q_2 \in \{q, q'\}$ it must be that $f(q_2) \in \delta^*_\infty(f(q_1), y_i)$. As $f(q_1) = f(q_2)$ it must be that $y_i \in L(f(q_1), f(q_2))$, so it follows that $x \in L(f(q_1), f(q_2))$. As $u \in \mathit{Pre}(f(q))$, we have that $ux \in \mathit{Pre}(f(q))$. Thus, $uxv \in L(A/\pi)$. As each word in $L(A/\pi)$ is also in $L(A_\infty)$, we have that $L(A/\pi) \subseteq L(A/\pi)$ as required. $\square$

Lemma 10 only provides a sufficient condition for when an extrapolation still allows for the precise fixpoint to be computed. There exist widening operators that induce partitions containing states that do not satisfy the conditions of the lemma though the language of the quotient automaton is contained in the fixpoint of the computation.

Example 10. The automata A occurs in a fixpoint computation.

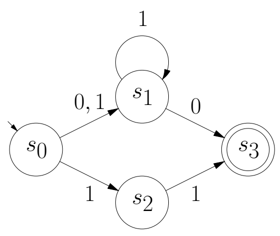


A

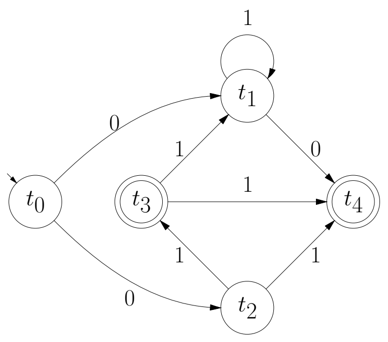


A_∇

The widened automaton A_∇ is obtained from A by merging the states q_1 and q_2. Consider the two automata A_∞ and M_∞ representing the language of the fixpoint of the computation.



M_∞



A_∞

Note that $L(A_\infty) = L(M_\infty)$ and that A_∞ is minimal and deterministic but M_∞ is not. As each state in A has only one prefix, A has the prefix property with respect to A_∞. Observe that $f(q_0) = t_0$, $f(q_1) = t_2$, $f(q_2) = t_3$ and $f(q_3) = t_4$. As $f(q_1) \neq f(q_2)$, Lemma 10 is not applicable.

Consider M_∞. For all states s_i, s_j in this automaton, $\mathit{Suff}(s_i) \cap \mathit{Suff}(s_j) = \emptyset$. We compare the suffixes of states in A to suffixes of states in M_∞. $\mathit{Suff}(q_0) \subseteq \mathit{Suff}(s_0)$, $\mathit{Suff}(q_1) \subseteq \mathit{Suff}(s_1)$, $\mathit{Suff}(q_2) \subseteq \mathit{Suff}(s_1)$ and $\mathit{Suff}(q_2) = \mathit{Suff}(s_3)$. The suffixes of the states that are merged in A are contained in the suffixes of a unique state in M_∞.

Our choice of the automaton M_∞ in Example 10 is not arbitrary. This automaton is isomorphic to the syntactic left congruence for $L(A_\infty)$. Recall that a minimal deterministic automaton is isomorphic to the Nerode equivalence, which is the syntactic right congruence for regular languages. Just as the sufficient condition for convergence to a fixpoint represented by a minimal deterministic automaton is described in terms of the prefix property, we identify another sufficient condition in terms of suffixes of an automaton and the *left automaton* representing the fixpoint.

Definition 22. The syntactic left congruence for a regular language $L \subseteq \Sigma^*$ is a relation ${}_L\!\sim\, \subseteq \Sigma^* \times \Sigma^*$ such that for $u, v \in \Sigma^*$, $u \;{}_L\!\sim\; v \Leftrightarrow \forall w \in \Sigma^* : wu \in L \Leftrightarrow \in L$.

The automaton isomorphic to the left congruence has one final state and a set of initial states.

Definition 23 (Left Canonical Automaton). Let π be the partition of a regular language L induced by the syntactic left congruence. The left canonical automaton for L, $M_L = (Q, \Sigma, \delta, I, q_F)$ is defined as follows: $Q = \{[w]_\pi | w \in \Sigma^*\}$, $q_F = [\lambda]_\pi$, $I = \{[w]_\pi | w \in L\}$ and $\delta([wa]_\pi, a) = \{[w]_\pi\}$.

The second sufficient condition we provide is in terms of suffixes of states in an automaton A and a left canonical automaton M_∞.

Definition 24 (Suffix property). Let A be an automaton in a fixpoint computation with a regular fixpoint represented by the left canonical automaton M_∞. A state r in A has the suffix property with respect to M_∞ iff there exists t in M_∞ such that $\mathit{Suff}(r) \subseteq \mathit{Suff}(t)$. The automaton A has the suffix property if every state in A has the suffix property.

If a state r has the suffix property with respect to M_∞, as the suffixes of states in M_∞ are pairwise disjoint, the state t in M_∞ such that $\mathit{Suff}(r) \subseteq \mathit{Suff}(t)$ is unique. Denote this state as $g(r)$. We provide the analogue of Lemma 10 for suffix properties.

Lemma 11. *Let π be a partition of the states of A and M_∞ be a left-canonical automaton representing the fixpoint. If $L(A) \subseteq L(M_\infty)$ and for any two states q, q' in the same block of π the suffix property holds and $g(q) = g(q')$, then $L(A/\pi) \subseteq L(M_\infty)$.*

Proof. We proceed as in the proof of Lemma 10. By the same inductive argument as in Lemma 10, it suffices to show that if π is of index $|Q| - 1$ and the conditions of the current lemma hold, then $L(A/\pi) \subseteq L(M_\infty)$.

Denote the two states the same block of π as q, q' and the corresponding state in A/π as $[q]$. If a state r in A has the suffix property with respect to M_∞, as $\mathit{Suff}(r) \subseteq \mathit{Suff}(g(r))$ and $L(A) \subseteq L(M_\infty)$ and because the states in M_∞ have disjoint suffixes, it must be that $\mathit{Pre}(r) \subseteq \mathit{Pre}(g(r))$.

By Remark 11, it is sufficient to examine $\mathit{Pre}([q]) \cdot \mathit{Suff}([q])$. Consider a word $uxv \in \mathit{Pre}([q]) \cdot \mathit{Suff}([q])$ such that $u \in \mathit{Pre}(\{q, q'\}) \setminus L(\{q, q\})$, $x \in L(\{q, q'\})$ and $v \in \mathit{Suff}(\{q, q'\}) \setminus L(\{q, q\})$. From the previous observation, As $\mathit{Pre}(q) \subseteq \mathit{Pre}(g(q))$ and $\mathit{Pre}(q') \subseteq \mathit{Pre}(g(q'))$ and $g(q) = g(q')$, $u \in \mathit{Pre}(g(q))$. We have to show that $xv \in \mathit{Suff}(g(q))$.

Without loss of generality, say $v \in \mathit{Suff}(q)$. Consider $x = y_1 \dots y_k$, where each $y_k \in L(\{q, q'\})$. For each y_i, for some $q_1, q_2 \in \{q, q'\}$ it must be that $g(q_2) \in \delta^*_\infty(g(q_1), y_i)$. As $g(q_1) = g(q_2)$, $y_i \in L(g(q_1), g(q_2))$. It follows that $x \in L(g(q_1), g(q_2))$. As $v \in \mathit{Suff}(g(q))$, we have that $xv \in \mathit{Suff}(g(q))$. Thus, $uxv \in L(A/\pi)$. As each word in $L(A/\pi)$ is also in $L(M_\infty)$, we have that $L(A/\pi) \subseteq L(M_\infty)$ as required. □

The conditions we provide use either the canonical or the left-canonical automaton to characterise when merging states does not extrapolate the language of the widened automaton beyond the fixpoint. Both conditions are only sufficient conditions. In Section 4.3, we provide an example of a fixpoint computation with automata that neither have the prefix property nor the suffix property. However, as the language of such automata is contained in the fixpoint, there may be some other representation of the fixpoint with respect to which a widening candidate satisfies either the prefix or suffix property. The final condition we provide in this section is in terms of *any* automaton representing a fixpoint.

Lemma 12. *Let A be an automaton with states Q in a computation with the language L_∞ as the fixpoint. There exists an automaton N_∞ with states Q_∞ such that $L(N_\infty) = L_\infty$ and for each state $q \in Q$, there exists $t \in Q_\infty$ such that $Pre(q) \subseteq Pre(t)$ or $Suff(q) \subseteq Suff(t)$.*

Proof. We prove by contradiction that if the statement does not hold, then $L(A) \nsubseteq L_\infty$. Assume the lemma does not hold. We have that for all N_∞ representing L_∞, there exists $q \in Q$ such that for all $t \in Q_\infty$, $Pre(q) \supset Pre(t)$ and $Suff(q) \supset Suff(t)$. Consequently, there exists a word $u \in Pre(q) \setminus Pre(Q)$ and a word $v \in Suff(q) \setminus Suff(Q)$. The word $uv \in L(A)$ but $uv \notin L(N_\infty)$ for any N_∞ representing L_∞. Thus, $L(A) \nsubseteq L_\infty$, contradiction the premise of the lemma. $\square$

Given an automaton A with states Q in a fixpoint computation, let N_∞ be an automaton with states Q_∞ representing the fixpoint as in Lemma 12. For each state q in A, let $F(q) = \{t \in Q_\infty | \, Pre(q) \subseteq Pre(t)\}$ be the set of states in N_∞ reachable by all words in $Pre(q)$. Similarly, let $G(q) = \{t \in Q_\infty | \, Suff(q) \subseteq Suff(t)\}$ be the states that accept all words in $Suff(q)$. We know by Lemma 12 that for any state q in A, either $F(q)$ or $G(q)$ must be defined.

Remark 17. If $F(q)$ is defined for a state q, then $Pre(q) \subseteq \bigcap_{t \in F(q)} Pre(t)$.

Remark 18. If $G(q)$ is defined for a state q, then $Suff(q) \subseteq \bigcap_{t \in G(q)} Suff(t)$.

Theorem 7. *Let A be an automaton in a fixpoint computation with states Q and N_∞ be an automaton with states Q_∞ representing the fixpoint and satisfying the conditions of Lemma 12. Let π be a partition of Q. For any two states q and q' in the same block of π, if one of the following conditions hold:*

1. $F(q) = F(q')$
2. $F(q) = G(q')$
3. $G(q) = G(q')$

then $L(A/\pi) \subseteq L(N_\infty)$.

Proof. The initial induction step is similar to that in Lemma 10. We consider π to be a partition of index $|Q|-1$ and show that $L(A/\pi) \subseteq L(N_\infty)$.

We know from Lemma 12 that either $F(q)$ or $G(q)$ must be defined and that the same applies for q'. Consider a word $uxv \in \mathit{Pre}([q]) \cdot \mathit{Suff}([q])$ such that $u \in \mathit{Pre}([q]) \setminus L([q],[q])$ and $v \in \mathit{Suff}([q]) \setminus L([q],[q])$ and $x = y_1 \dots y_k$ with $y_i \in L([q],[q])$ for $0 \le i \le k$. We consider the three conditions in the lemma and show that in each case, $uxv \in L(N_\infty)$.

1. $F(q) = F(q')$: The word v must be in $\mathit{Suff}(q) \cup \mathit{Suff}(q')$. Say v is in $\mathit{Suff}(q)$. As $L(A) \subseteq L(N_\infty)$, it must be that $\mathit{Suff}(q) \subseteq \bigcup_{t \in F(q)} \mathit{Suff}(t)$. Thus there exists a state $t \in Q_\infty$ with $v \in \mathit{Suff}(t)$ and $u \in \mathit{Pre}(t)$. It remains to show that $y_i \in L(t,t)$. Note that each $y_i \in L(\{q,q'\})$ and is thus either contained in a prefix of q or a prefix of q'. As $\mathit{Pre}(q) \subseteq \mathit{Pre}(t)$ and $\mathit{Pre}(q') \subseteq \mathit{Pre}(t)$, we have that $uy_i \in \mathit{Pre}(t)$ for each y_i. Further, because $u \in \mathit{Pre}(t)$ and $uy_i \in \mathit{Pre}(t)$, we have that $uy_i^* \in \mathit{Pre}(t)$. As each $y_i \in L(t,t)$, $L([q],[q]) \subseteq L(t,t)$. It follows that $uxy \in L(N_\infty)$.

2. $F(q) = G(q')$: For any $t \in F(q)$, $u \in \mathit{Pre}(t)$ and $v \in \mathit{Suff}(t)$. It remains to show that for each $y_i \in L(q,q')$, $y_i \in L(t,t)$. If $y_i \in L(q,q)$, as $u' \in \mathit{Pre}(t)$ for any $u' \in \mathit{Pre}(q)$ and $u'y_i \in \mathit{pre}(q)$ we have that $u'y_i \in \mathit{Pre}(t)$ and hence $y_i \in L(t,t)$. If $y_i \in L(q,q')$, as any $u' \in \mathit{Pre}(q)$ is also in $\mathit{Pre}(t)$ and $u'y_i \in \mathit{Pre}(q')$ and $\mathit{Pre}(q') \subseteq \mathit{Pre}(t)$, $u'y_i \in \mathit{Pre}(t)$. A similar argument applies for $y_i \in L(q',q) \cup L(q',q')$. Thus, $x \in L(t,t)$ and $uxv \in L(N_\infty)$ as required.

3. $G(q) = G(q')$: The word u must be in $\mathit{Pre}(q) \cup \mathit{Pre}(q')$. Say u is in $\mathit{Pre}(q)$. As $L(A) \subseteq L(N_\infty)$, it must be that $\mathit{Pre}(q) \subseteq \bigcup_{t \in G(q)} \mathit{Pre}(t)$. Thus there exists a state $t \in Q_\infty$ with $u \in \mathit{Pre}(t)$ and $v \in \mathit{Suff}(t)$. It remains to show that $y_i \in L(t,t)$. Note that each $y_i \in L(\{q,q'\})$ and is thus either contained in a suffix of q or a suffix of q'. As $\mathit{Suff}(q) \subseteq \mathit{Suff}(t)$ and $\mathit{Suff}(q') \subseteq \mathit{Suff}(t)$, we have that $y_iv \in \mathit{Suff}(t)$ for each y_i. Further, because $v \in \mathit{Suff}(t)$ and $y_iv \in \mathit{Suff}(t)$, we have that $y_i^*v \in \mathit{Suff}(t)$. As each $y_i \in L(t,t)$, $L([q],[q]) \subseteq L(t,t)$. It follows that $uxy \in L(N_\infty)$.

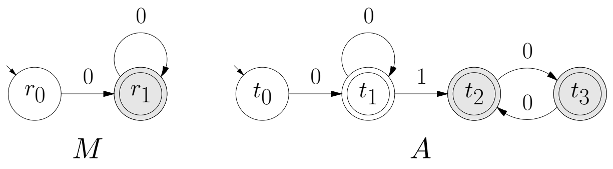
□

4.3 Elementary Widening Seeds

The conditions we identified can be used to study different widening operators. We recall our notation. $\mathcal{S}$, is a widening seed, M, a widening parameter, A the widening candidate, A_∞, the minimal, deterministic automaton representing the fixpoint of the computation, if it is regular, and A_∇, the widened automaton. We analyse each widening seed as indicated in Section 4.1. In addition, we provide examples to illustrate the effect of widening. For each seed $\mathcal{S}$, we also compute $\mathcal{R}_\nabla$ and $\equiv^M_\mathcal{S}$ and indicate the elements in these relations that affect the widened automaton.

The first widening seed we consider is $\{=_s\}$. Two states t and t' in A are merged if there exists a state r in M such that $t =_s r$ and $t' =_s r$. The following example illustrates the use of this widening seed.

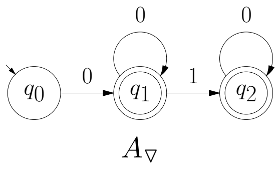
Example 11. Consider the automaton A with the widening parameter M.



The relations computed are:

$$
\begin{aligned}
=_s &= \{\langle r_1, t_2\rangle, \langle r_1, t_3\rangle\}\\
\mathcal{R}_\nabla &= \{\langle t_2, t_3\rangle, \langle t_3, t_2\rangle\}\\
\equiv^M_\mathcal{S} &= id \cup \mathcal{R}_\nabla
\end{aligned}
$$

The states in the same equivalence class of $\equiv^M_\mathcal{S}$ are shaded in the illustration. The widened automaton A_∇ below is obtained by merging t_2 and t_3.



Note that $L(A_\nabla) = L(A)$ and that A_∇ is minimal.

Lemma 13 (Properties of $\{=_s\}$). *Let $\mathcal{S}$ be $\{=_s\}$. $\mathcal{S}$ has the following properties:*

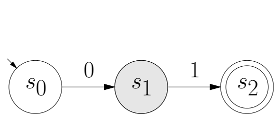
1. *It is not extrapolating for finite automata or WDBAs.*
2. *If A is the widening parameter, $A/\equiv^A_\mathcal{S}$ is minimal.*
3. *It does not enforce termination.*
4. *If the fixpoint is regular, the computation with widening converges in the limit to the precise fixpoint.*

Proof. (1) All states in an equivalence class accept the same language, hence the finite language of the widened automaton does not increase. As two states with the same suffixes have the same ω-suffixes, the ω-language accepted does not change.
(2) If A is the widening parameter, all states in A accepting the same language are in the same equivalence class of $\equiv_{\mathcal{S}}^{M}$. Therefore, no two states in $A_\triangledown$ accept the same language and it is minimal.
(3) As the widening seed is not extrapolating, the sequence of languages in the computation with widening is identical to the sequence in the computation without. Hence, termination is not enforced.
(4) As the sequence of languages in a fixpoint computation with and without widening is the same, if the computation without widening converges to a regular fixpoint. □

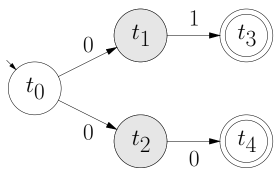
A computation with minimisation steps is an instance of Algorithm 1 (Fixpoint Computation With Widening). At step i of the computation, the widening parameter is A_i and the widening seed is $\{=_s\}$. By Lemma 13, part 2 the widened automaton is minimal. Note that the automaton $A_\triangledown$ may not be deterministic and hence, not canonical.

The widening seed $\{=_p\}$ has similar properties to $\{=_s\}$. It can be used to reduce the size of an automaton but does not change the language accepted. In addition, for deterministic automata, the widening equivalence is the identity relation.

Example 12. A widening parameter M and automaton A are shown below.



M

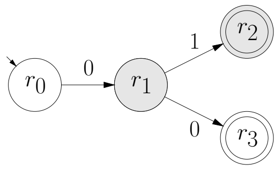


A

The relations computed are:

$$\begin{aligned} =_p &= \{\langle s_0, t_0\rangle, \langle s_1, t_1\rangle\langle s_1, t_2\rangle, \langle s_2, t_3\rangle, \} \\ \mathcal{R}_\triangledown &= \{\langle t_1, t_2\rangle, \langle t_2, t_1\rangle\} \cup id \setminus \{\langle t_4, t_4\rangle\} \\ \equiv_{\mathcal{S}}^{M} &= id \cup \mathcal{R}_\triangledown \end{aligned}$$

The pair $\langle t_4, t_4\rangle \notin \mathcal{R}_\triangledown$ because no state in M is related to t_4. The shaded states are merged to obtain $A_\triangledown$ below.



$A_\triangledown$

Observe that A and A_∇ are the MCA and the trie for the sample $\{01, 00\}$.

Lemma 14 (Properties of $\{=_p\}$). *Let $\mathcal{S}$ be $\{=_p\}$. $\mathcal{S}$ has the following properties:*

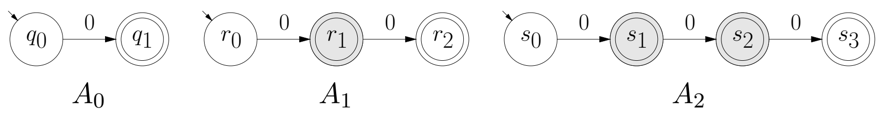
1. *It is not extrapolating for finite automata.*
2. *If A is a deterministic, $A\nabla$ is isomorphic to A. Further, $\mathcal{S}$ is not extrapolating for WDBAs.*
3. *If A is an MCA for a sample S, $A/\equiv_{\mathcal{S}}^{A}$ is $Trie(S)$.*
4. *It does not enforce termination.*
5. *If the fixpoint is regular, the computation with widening converges in the limit to the precise fixpoint.*

Proof. (1) All states in an equivalence class of $\equiv_{\mathcal{S}}^{A}$ have the same prefixes. Merging states with the same prefixes does not change the language of the automaton, therefore, this seed is not extrapolating.
(2) If A is deterministic, no two states have the same prefixes. Therefore, each equivalence class of $\equiv_{\mathcal{S}}^{A}$ has only one state and A_∇ is isomorphic to A. As WDBAs are deterministic, this seed is not extrapolating for WDBAs.
(3) As A is the widening parameter, all states in A with the same prefixes are related. For any sample S, $Trie(S)$ is obtained from $MCA(S)$ by merging states with the same prefixes. Hence, if A is an MCA and the widening parameter, A_∇ is a trie.
(4) Identical argument to part 3 of Lemma 13.
(5) Identical argument to part 4 of Lemma 13. □

As this widening seed does not merge states in a deterministic automaton, the partition it induces is finer than the partition induced by $\{=_s\}$. An NFA might have states reachable by identical prefixes and states with different prefixes but identical suffixes, so the partitions induced by $\{=_p\}$ and $\{=_s\}$ may be incomparable.

The next widening seed we consider is $\{=_s^k\}$. Two states are in the relation $=_s^k$ if they have the same suffixes of length at most k. Bierman and Feldman [1972] introduced the *k-tails heuristic*, in which states in a non-deterministic Mealy machine related by $=_s^k$ are merged. This heuristic has since been used in various applications such as regular inference, extracting formal specifications from protocol code [Ammons et al. 2002], inference of document structure [Sankey and Wong 2001] and synthesis [Choi 2002] and testing [Maadani and Geffroy 1991] of VLSI circuits. We refer to $\{=_s^k\}$ as the k-tails widening seed and illustrate its use with the following example.

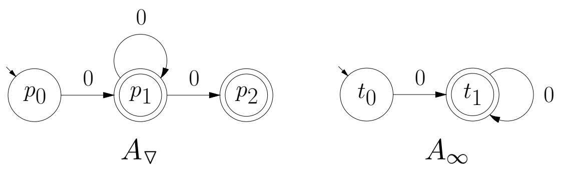
Example 13. Consider the following sequence of automata in a fixpoint computation with the regular fixpoint A_∞ as shown.



Let A_2 be the widening candidate and A_1 be the widening parameter. Choose the widening seed $\mathcal{S}$ as $\{=^1_s\}$. The relations computed are:

$$
\begin{aligned}
=^1_s &= \{\langle r_0, s_0\rangle, \langle r_1, s_1\rangle\langle r_1, s_2\rangle, \langle r_2, s_3\rangle, \}\\
\mathcal{R}_\nabla &= id \cup \{\langle s_1, s_2\rangle, \langle s_2, s_1\rangle\}\\
\equiv^{A_1}_{\mathcal{S}} &= id \cup \mathcal{R}_\nabla
\end{aligned}
$$

The states s_1 and s_2, which are shaded, are in the same equivalence class as they are both related to r_1. The shaded states are merged. The widened automaton and the fixpoint of the sequence are shown below.



We can see that $L(A_\nabla) = L(A_\infty)$.

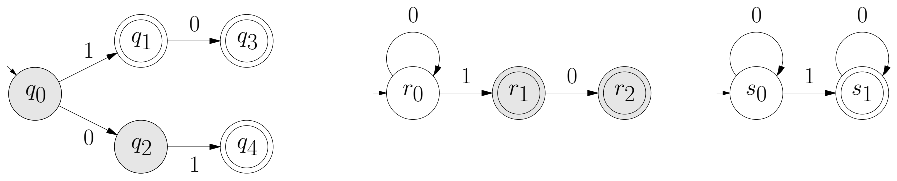
The k-tails widening seed has many interesting properties. If an over-approximation is too coarse for a certain analysis, using a larger k will result in a more precise over-approximation. As the number of different languages containing strings of length at most k is finite, there exists a choice of the widening parameter that always ensures termination. However, we show that an arbitrary choice of the widening parameter may not guarantee termination, so this choice should be made carefully. By adapting a regular inference algorithm of Trakhtenbrot and Barzdin [1973], we identify computations in which the k-tails widening seed can be used to compute precisely compute a regular fixpoint. We illustrate the algorithm using an example and then discuss our modification.

Let A_∞ be an automaton with n states. The algorithm requires as input a finite, prefix-complete sample S that includes all words of length up to $2n-1$ accepted by A_∞. The algorithm begins with $Trie(S)$ and proceeds by merging states related by $=^k_s$ for decreasing values of k. The initial value of k for the relation $=^k_s$ is $2n-1$. If a state r in the trie has a predecessor r' such that $r =^k_s r'$, the sub-tree from r is deleted and r is merged with r'. If all pairs of states have been compared, the value of k is decremented and the steps described above are repeated. If all states have been compared using the relation $=^0_s$, the algorithm terminates.

Example 14. Let $S = \{1, 01, 10\}$ be the set of strings in a sample. The algorithm begins with the trie A_0 accepting S and with $k = 1$. Consider the suffixes of each state of length at most 1. $\mathit{Suff}_1(q_0) = \{1\}$, $\mathit{Suff}_1(q_1) =$

$\{\lambda, 0\}$, and $\mathit{Suff}_1(q_2) = \{1\}$. The states q_0 and q_1 are compared. As $q_0 \neq_s^1 q_1$, q_0 and q_2 are compared. As $\mathit{Suff}_1(q_0) = \mathit{Suff}_1(q_2)$, the two states are merged to obtain A_1.

We compare the states in A_1. $\mathit{Suff}_1(r_0) = \{1\}$, $\mathit{Suff}_1(r_1) = \{\lambda, 0\}$, and $\mathit{Suff}_1(r_2) = \{\lambda\}$. As no two states are in the relation $=_s^1$, the states are compared using the relation $=_s^0$. The states r_1 and r_2 both accept $\{\lambda\}$ and are merged to obtain A_2. As all states have been compared using $=_s^0$, the algorithm terminates.



A_0 A_1 A_2

Trakhtenbrot and Barzdin [1973] proved the following theorem about the correctness of the algorithm.

Theorem 8 (Trakhtenbrot and Barzdin 1973, Theorem 4.1). *Let A be an automaton with n states.*

1. *For any states q, q' in A, if $\mathit{Suff}_{2n-1}(q) \neq \mathit{Suff}_{2n-1}(q')$, $\mathit{Suff}(q) \neq \mathit{Suff}(q')$.*
2. *Given a finite, prefix complete sample that includes all words of length up to $2n-1$ accepted by A, there exists an algorithm for constructing A.*

We have the following lemma about the properties of the k-tails widening seed.

Lemma 15 (Properties of $\{=_s^k\}$). *The k-tails widening seed $\{=_s^k\}$ has the following properties:*

1. *If $k = 0$, $A_\triangledown$ has at most two states.*
2. *It is extrapolating.*
3. *Let $p = |\Sigma|$ be the size of the alphabet with $p \geq 2$ and $\ell(p,k) = 2^{\left(\frac{p^{k+1}-1}{p-1}\right)}$. For any k, there exists a choice of the widening parameter which ensures termination in at most $\ell(p,k)$ steps.*
4. *There exists a choice of the widening parameter such that the computation does not terminate.*

5. *Let A_∞ be an automaton with n states representing a regular fixpoint. If there exists an automaton A in the fixpoint computation accepting a prefix complete sample with words of length at least $4n-2$, the k-tails widening seed can be used to precisely compute A_∞ using A as the widening parameter.*

Proof. (1) There are only two suffixes of length 0: $\emptyset$ and $\{\lambda\}$. If $k = 0$, for all widening parameters M, $\equiv_{\mathcal{S}}^{M}$ has at most two equivalence classes: those corresponding to final and non-final states. Hence, A_∇ has at most two states.
(2) Consider the minimal automaton accepting the language $\{01\}$. Choose k as zero. Using the same automaton as the widening parameter, the widened automaton accepts 0^*1.
(3) Let $\ell(p,k)$ denote the number of languages over a p symbol alphabet containing words of length at most k. The value $\ell(p,k)$ is the number of different choices of final and non-final states in a p-ary tree of depth k. If no states are marked final, the language is $\emptyset$, if only the root of the tree is marked as final, the language accepted is $\{\lambda\}$ and so forth. The total number of nodes in a p-ary tree of depth k with $p \geq 2$ is $\left(\frac{p^{k+1}-1}{p-1}\right)$. As every state can be final or non-final, $\ell(p,k) = 2^{\left(\frac{p^{k+1}-1}{p-1}\right)}$.

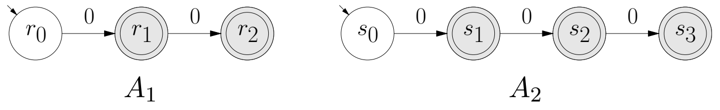
Let A be the automaton to be widened. Choose A as the widening parameter. If A has more than $\ell(p,k)$ states, by the Pigeon hole principle, at least two states have the same language. As A is the widening parameter, these two states are related and are in the same equivalence class of $\equiv_{\mathcal{S}}^{A}$.
(4) Consider the sequence of automata in Example 13. Let $k = 1$ and the widening parameter always be A_0. For all $i \geq 0$, no two states in the automaton A_i have the same suffixes as the states q_0 or q_1 in the automaton A_0. The fixpoint computation with merging is thus identical to the one without and does not terminate.
(5) Every state in an automaton can be reached by a word of length at most $n-1$ and reached twice by a word of length at most $2n-1$ (consider an automaton with a ring structure). By Theorem 8, any two states in a minimal n state automaton can be distinguished by a suffix of length at most $2n-1$. If two states in a trie corresponding to the same state in A_∞ are to be related only to each other, the sub-tree below each state should be of depth $2n-1$. As a word of length $2n-1$ may be required to visit a state twice, the minimum depth of the trie required is $4n-2$. This proves the bound on the size of the tree.

Let $\mathcal{S} = \{=_s^{2n-1}\}$ and M be a widening parameter. Consider the equivalence classes of $\equiv_{\mathcal{S}}^{M}$. If a state at depth at most $2n-1$ is related to a state at the same or lower depth in the trie, by Theorem 8, they must be identical in A_∞. If a state at depth greater than $2n-1$ is related to a state at a lower depth, they must once again by Theorem 8 be indistinguishable in A_∞. We only need to consider states at depth greater than $2n-1$ that are related. Let r and r' be two such states. If they are final states and there exists $w \in \Sigma^*$ such that $\delta^*(r, w) = r'$, $w \cdot \mathit{Suff}(r') \subseteq \mathit{Suff}(r)$. Therefore, a pair of states r, r' at depth greater than $2n-1$ are related only if $L(r, r') = L(r', r) \neq \emptyset$. As r and r' accept words of length less than $2n-1$, they are related by $r =_s^{2n-1} r'$ iff $\mathit{Suff}(r) = \mathit{Suff}(r')$, and by Lemma 3, merging such states in a deterministic automaton does not change the language accepted. Consider any two non-final states r, r' at depth less than $2n-1$. By a similar argument, we can show that if two non-final states at depth greater than $2n-1$ are merged, the language of the automaton does not change. Therefore, the language of the automaton is extrapolated only by merging states at dept at most $2n-1$. As these states can be uniquely identified, $L(A_\nabla) = L(A_\infty)$. $\square$

Our use of the k-tails widening seed to precisely compute a fixpoint as in Lemma 15 differs from the standard use of a widening operator. A widening operator is typically used to detect and generalise an increment between at least two arguments. If a single automaton is used as both the widening candidate and parameter, the widening operator has only one argument. This difference in usage arises because unlike representations such as intervals or polyhedra, an automaton implicitly contains information about the previous steps in the fixpoint computation. If the fixpoint is regular, it may suffice to detect a pattern in the words accepted by a single automaton rather than detect an increment between the languages of two automata. Nevertheless, using a widening operator with a single argument may be too restrictive. As we may have very little information about the fixpoint, it may be difficult to decide when an automaton contains sufficient information to be used as both the widening candidate and parameter. For example, to precisely compute a regular fixpoint using the k-tails widening seed, we need to guess the number of states in the fixpoint automaton.

The next widening seed we consider is $\{=_p^k\}$, which is complementary to the k-tails widening seed. Two states in the widening candidate are related if there exists a state in the widening parameter with the same prefixes of length at most k. As with the k-tails widening seed, there exists a choice of the widening parameter that enforces termination. In contrast to the k-tails widening seed, We show that there exist a family of regular fixpoints that cannot be computed precisely using this widening seed for any choice of the widening candidate and parameter. As usual, we begin with an example.

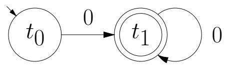
Example 15. The two automata below occur in the fixpoint computation in Example 13.



Let A_2 be the widening candidate and A_1 be the widening parameter. Choose the widening seed $\mathcal{S}$ as $\{=^1_p\}$. The relations computed are:

$$
\begin{aligned}
=^1_p &= \{\langle r_0, s_0\rangle, \langle r_i, s_j\rangle\} \text{ where } i \in \{1,2\}, j \in \{1,2,3\} \\
\mathcal{R}_\triangledown &= \{\langle s_0, s_0\rangle, \langle s_i, s_j\rangle\} \text{ where } i, j \in \{1,2,3\} \\
\equiv^{A_1}_{\mathcal{S}} &= id \cup \mathcal{R}_\triangledown
\end{aligned}
$$

The shaded states in A_2 are in the same equivalence class of $\equiv^{A_1}_{\mathcal{S}}$ and are related to the shaded states in A_1 by $=^1_p$. The widened automaton is shown below.



$A_\triangledown$

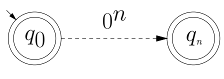
The fixpoint is computed precisely and is isomorphic to A_∞ in Example 13. Note that the states related and the widened automaton are different from those obtained using the k-tails widening seed.

Lemma 16 (Properties of $\{=^k_p\}$). *The widening seed $\{=^k_p\}$ has the following properties:*

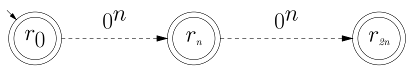
1. *If A has two or more states and $k = 0$, $A_\triangledown$ has two states.*
2. *It is extrapolating.*
3. *Let $p = |\Sigma|$ be the size of the alphabet Σ and $\ell(p,k) = 2^{\left(\frac{p^{k+1}-1}{p-1}\right)}$. For every k, there exists a choice of the widening parameter that ensures termination in at most $\ell(p,k)$ steps.*
4. *There exists an infinite family of fixpoint computations with regular fixpoints A^n_∞ such that no choice of k or the widening parameter allows for A^n_∞ to be computed precisely.*

Proof. (1) There are only two prefixes of length 0: $\{\lambda\}$ and $\emptyset$. If A has more than one state, the initial state has the prefix λ and all other states have the empty prefix.
(2) Let $k = 0$ and A be a minimal automaton with more than 2 states. We know from Lemma 3 that merging states with different suffixes extrapolates the language of an automaton. From the previous part, the widened automaton has only two states, so $L(A) \subset L(A_\triangledown)$.
(3) The value of the function $\ell(p,k)$ is the number of ways to mark the nodes in a p-ary tree of depth k as initial or non-initial. If a node in the tree is initial, the reverse of the path from that node to the root corresponds to a

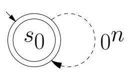
prefix of a state in the automaton. The same argument as in Lemma 15, part 3 applies.
(4) Consider the family of sequences of automata A_i^n with $L(A_i^n) = \{0^{nj} | 0 \leq j \leq i\}$ where $n > 1$. Two automata A_1^n and A_2^n are shown below.



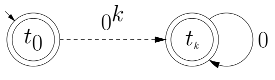
A_1^n



$^nA_2$



A_∞^n



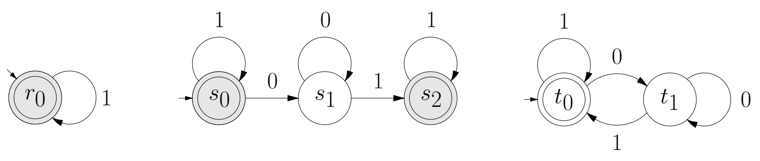
A_∇

For each n, the fixpoint of the sequence is the automaton A_∞^n accepting the language $(0^n)^*$. However, for any choice of k for the relation $=_p^k$, for any automaton A_i^n with more than k states, for all states r, r' with a prefix w such that $|w| \geq k$, $r =_p^k r'$. Hence, the widened automaton A_∇ as shown above accepts the language 0^k0^*. We see that $0^k0^* \neq (0^n)^*$ for all $n > 1$. $\square$

We have demonstrated in Lemma 16 that the widening seed $\{=_p^k\}$ can be extremely imprecise. The imprecision is not specific to the family of automata we mention. If the automata in the fixpoint computation are tries, each state has exactly one prefix and for any k in the relation, if the trie has depth $n > k$, there will be $n - k$ states with the same prefix that will be merged. The extrapolation introduced is fairly arbitrary as different tries having the same structure will have the same structure after widening irrespective of the language accepted.

Next, we consider the widening seed $\{\subseteq_s\}$. It is the first widening seed we study that uses a preorder. In the examples presented so far, the widening equivalence was of the form $id \cup \mathcal{R}_\nabla$. If the widening seed uses a preorder, all states in the same equivalence class of the widening equivalence may not be directly related to each other and the effect of the transitive closure comes into play. We provide multiple examples to illustrate the varying effects of widening. This widening seed can be used to compute the precise fixpoint for interesting examples. Conversely, if any state in the widening parameter has only finitely many suffixes, the extrapolation introduced may be arbitrarily large. We also provide an example of non-terminating computations that are accelerated using widening but still do not terminate. We conclude our analysis of this widening seed by identifying computations with widening that converge to the precise fixpoint.

Example 16. The automaton A_0 below is a widening parameter and A_1, a widening candidate. Let $\mathcal{S}$ be $\{\subseteq_s\}$.



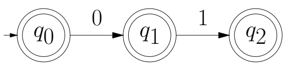
A_0 A_1 A_∇

Observe that *Suff*$(r_0) \subseteq$ *Suff*(s_0) and *Suff*$(r_0) \subseteq$ *Suff*(s_2). The relations $\subseteq_s$, $\mathcal{R}_\nabla$ and $\equiv_{\mathcal{S}}^{A_0}$ are:

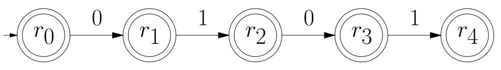
$$\begin{aligned}
\subseteq_s &= \{\langle r_0, s_0\rangle, \langle r_0, s_2\rangle\} \\
\mathcal{R}_\nabla &= \{\langle s_0, s_2\rangle, \langle s_2, s_0\rangle\} \\
\equiv_{\mathcal{S}}^{A_0} &= id \cup \mathcal{R}_\nabla
\end{aligned}$$

The shaded states in A_1 related to r_0 and are in the same equivalence class of $\equiv_{\mathcal{S}}^{A_0}$. The widened automaton A_∇ is obtained by merging s_0 and s_2, and is the precise fixpoint. Our choice of A_0 and A_1 is not special. The i^{th} automaton A_i in this computation accepts the language $\cup_{0\leq j\leq i}1^*(00^*11^*)^j$. For any two automata computation are used as the widening parameter or candidate, the widened automaton is the precise fixpoint.

Example 17. Let the automaton A_1 be the widening candidate and A_0, the widening parameter.



A_0

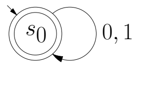


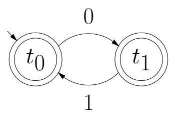
A_1

As *Suff*$(q_2) = \{\lambda\}$ and for every state r_i, $\{\lambda\} \subseteq$ *Suff*(r_i), q_2 is related to every state in A_1. The computed relations are:

$$\begin{aligned}
\subseteq_s &= \{\langle q_0, r_0\rangle, \langle q_0, r_2\rangle, \langle q_1, r_1\rangle, \langle q_1, r_3\rangle, \langle q_2, r_i\rangle\} \text{ where } 0 \leq i \leq 4 \\
\mathcal{R}_\nabla &= \{\langle r_i, r_j\rangle\} \text{ where } 0 \leq i, j \leq 4 \\
\equiv_{\mathcal{S}}^{A_0} &= \mathcal{R}_\nabla
\end{aligned}$$

The widening equivalence has unit index and the widened automaton A_∇, shown below, accepts Σ^*. The fixpoint of the computation is also shown below.

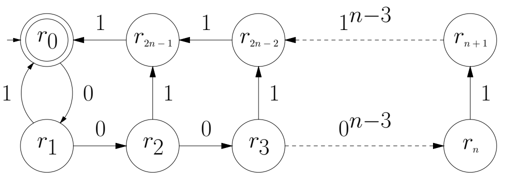




A_∇ A_∞

In this computation, there exists no choice of the widening candidate or seed that for the fixpoint A_∞ to be computed precisely.

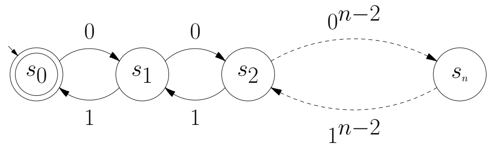
Example 18. Consider the sequence of deterministic automata A_n as shown below. The language accepted by A_n is described by the regular expression $(01 + \ldots + 0^n1^n)^*$. The limit of the sequence of languages $L(A_0), \ldots, L(A_n)$ is a context free language.



A_n

The suffixes of the state r_2 are $0 \cdot \mathit{Suff}(r_3) \cup 1 \cdot \mathit{Suff}(r_{2n-1})$ and the suffixes of r_{2n-2} are $1 \cdot \mathit{Suff}(r_{2n-1})$. Thus, $\mathit{Suff}(r_{2n-2}) \subseteq \mathit{Suff}(r_2)$. In general, for $1 \leq i < n$, $\mathit{Suff}(r_i) = 1 \cdot \mathit{Suff}(r_{2n-i+1}) \cup 0 \cdot \mathit{Suff}(r_{i+1})$, and $\mathit{Suff}(r_{2n-i}) = 1 \cdot \mathit{Suff}(r_{2n-i+1})$. Hence, $\mathit{Suff}(r_i) \subseteq \mathit{Suff}(r_{2n-i})$. Also note that as the automaton is deterministic, for all states r and r', $\mathit{Pre}(r) \cap \mathit{Pre}(r') = \emptyset$.

Let A_n be the widening candidate and parameter. The relation $\subseteq_s$ is $(id \cup \{\langle r_{2n-1}, r_i\rangle\})$. The relation $\mathcal{R}_\nabla$ is $(id \cup \{\langle r_{2n-1}, r_i\rangle, \langle r_i, r_{2n-1}\rangle\})$. The widened automaton A_n^∇ is shown below.



A_n^∇

The effect of widening is interesting. The widening seed is extrapolating as $001011 \in L(A_n^\nabla \setminus L(A_n)$. The language of the widened automaton is neither an under or over-approximation of the fixpoint. The word $0^{n+1}1^{n+1}$, which is in the fixpoint, is not accepted by A_n^∇ and the word 001011 is not in the fixpoint.

Let the parenthetic subscripts in an expression of the form $(_i\ exp\ _i)$ denote the level of nesting of a sub-expression *exp* of a regular expression. The language $L(A_n^\nabla = (_1\, 0 \cdot \ldots (_n\, 01\, _n)^* \ldots \cdot 1\, _1)^*$ and can be interpreted as representing a sequence of matched parentheses with at most n levels of nesting. We find the effect of widening is interesting because though the language of the widened automaton is neither an under or over-approximation of the fixpoint, it satisfies non-trivial properties satisfied by language of the widening candidate (in this case, that the number of 1's and 0's is equal).

Let B_n be a sequence of WDBAs obtained from the automata A_n by marking all states final. The limit of this sequence of WDBAs is an ω-context free language. The ω-words in the fixpoint of the sequence consist of infinitely many finite sequences of 0s and 1s, where each sequence of 0s is followed by equally many 1s. The widened automaton B_n^∇ accepts infinite words containing finitely many sequences of 0s and 1s with at most finitely many more 0s than 1s. Once again, the language obtained after widening is neither an under or over-approximation of the fixpoint.

Lemma 17 (Properties of $\{\subseteq_s\}$). *Let $\mathcal{S}$ be $\{\subseteq_s\}$. The widening seed $\mathcal{S}$ has the following properties:*

1. *It is extrapolating.*
2. *If there exists a state r in the widening parameter M such that $\mathit{Suff}(r) = \{\lambda\}$, A_∇ has only one final state.*
3. *Enforces termination for some but not all fixpoint computations.*
4. *Let $\mathcal{T}^=$ be the reflexive closure of a transition relation $\mathcal{T}$ with a regular fixpoint A_∞. Let M be a widening parameter with states Q_M, A be the widening candidate with states Q and A_∞ be minimal and deterministic with states Q_∞. If the following conditions hold:*
   (a) for all $t, t' \in Q_\infty$, $t \neq t' \Rightarrow \mathit{Suff}(t) \not\subseteq \mathit{Suff}(t')$
   (b) for all $r \in Q_M$, there exists $t \in Q_\infty$ such that $\mathit{Suff}(r) \subseteq \mathit{Suff}(t)$ and for all $t' \in Q_\infty : t \neq t' \Rightarrow \mathit{Suff}(r) \not\subseteq \mathit{Suff}(t')$.

   the fixpoint computation with widening converges to the precise fixpoint.

Proof. (1) See Examples 16, 17 and 18.
(2) If $\mathit{Suff}(r) = \{\lambda\}$ for some state r, all final states are in the widening candidate are related to r and are in the same partition, hence A_∇ has only one final state.
(3) Widening ensures termination for the computations in Examples 16, and 18. We provide an example where the computation with widening does not terminate. Consider the sequence of automata A_n^∇ in Example 18. Each state s_i in this automaton accepts the word 1^i, not accepted by any other state. Thus for all s_i, s_j in A_n^∇, $\langle s_i, s_j\rangle \notin \subseteq_s$. As the widening parameter is also an automaton in this sequence, no two states in the widening candidate are related to the same state in the widening parameter and the computation does not terminate.
(4) From Remark 16, we know that it is sufficient to prove that $L(A/\equiv_{\mathcal{S}}^M) \subseteq L(A_\infty)$. We show that M has the prefix property and that the states in the same equivalence class satisfy the conditions of Theorem 7.

By condition 4b, for each state $r \in Q_M$, there exists a unique state in $t \in Q_\infty$ such that $\mathit{Suff}(r) \subseteq \mathit{Suff}(t)$. As there exists $v \in \mathit{Suff}(r)$ such that for all $t' \neq t$ in Q_∞, $v \notin \mathit{Suff}(t')$ and $L(M) \subseteq L(A_\infty)$, it must be that

$Pre(r) \subseteq Pre(t)$. For each r, this state t is unique, so each $r \in Q_M$ and consequently M has the prefix property.

Consider an equivalence class of $\equiv_{\mathcal{S}}^{M}$ with $n+1$ states. By definition of $\equiv_{\mathcal{S}}^{M}$, for any q, q' in the equivalence class, there exist two sequences of states $q = q_0, \dots q_k = q' \in Q$ and $r_1, \dots, r_k \in Q_M$ such that $0 \leq k \leq n$ and for each $1 \leq i \leq k$, $r_i \subseteq_s q_{i-1}$ and $r_i \subseteq_s q_i$. We show by induction on k that the conditions of Theorem 12 are satisfied. For the base case, take $k = 0$. By Lemma 12, either $F(q_0)$ or $G(q_0)$ is defined. Clearly, if an equivalence class has only one state, F or G agree for that state. For the induction hypothesis, assume that for $k = n-1$, $f(q_{n-1})$ is defined and that $f(q) = f(q_{n-1})$.

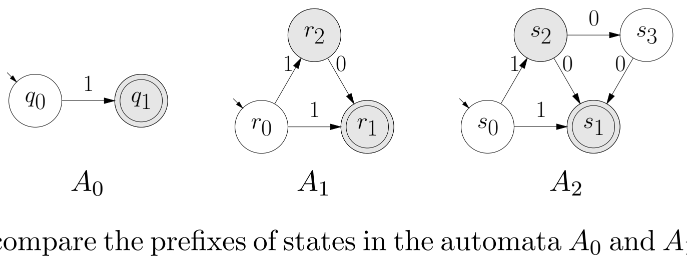
Consider the case $k = n$. We show that states q_{n-1} and q_n agree on f, which implies the conditions of Theorem 7. There exists $r_n \in Q_M$ such that $r_n \subseteq_s q_{n-1}$ and $r_n \subseteq_s q_n$. As r_n has the prefix property and condition 4b holds, we have that $Suff(r_n) \subseteq Suff(f(r_n))$. By the definition of $\subseteq_s$, we have that $Suff(r_n) \subseteq Suff(q_{n-1})$, which implies that there exists $v \in Suff(q_{n-1})$ such that $v \in f(r)$ and for all $t' \neq f(r)$ in Q_∞, $v \notin Suff(t')$. Hence, there exists a unique state $f(r)$ in Q_∞ such that $Suff(q_{n-1}) \subseteq Suff(f(r))$ and $Pre(q_{n-1}) \subseteq Pre(f(r))$. It follows that q_{n-1} has the prefix property and that $f(q_{n-1}) = f(r)$. By a similar argument, we have that q_n has the prefix property and that $f(q_n) = f(r)$ and hence, $f(q_n) = f(q_{n-1})$ as required. We have shown that for any two states q, q' in the same partition of the widening equivalence, the prefix property holds and $f(q) = f(q')$.

We know from Lemma 10 that if for all states in a block of a partition π the prefix property holds and $f(q) = f(q')$, then $L(A/\pi) \subseteq L(A_\infty)$. It follows that the computation with widening converges to the precise fixpoint.

□

The next widening seed we consider is $\{\subseteq_p\}$. Unlike the widening seeds $\{=_s^k\}$ and $\{=_p^k\}$, where the second seed introduced more imprecision this widening seed relates far fewer states than $\{\subseteq_s\}$. In fact, if all the automata in the fixpoint computation are deterministic, the computation with widening is identical to the one without! This can be verified by examining the states in any sequence of deterministic automata featured in previous examples. We show that this widening seed does extrapolate the language of NFAs and may even allow for computing a fixpoint precisely.

Example 19. Consider the following fixpoint computation with NFAs.

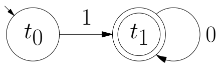


Let us compare the prefixes of states in the automata A_0 and A_1. $Pre(q_0) =$

$Pre(r_0) = \{\lambda\}$, $Pre(q_1) = Pre(r_2) = \{1\}$ and $Pre(r_1) = \{1, 10\}$. The relations required for widening are:

$$\begin{aligned} \subseteq_p &= \{\langle q_0, r_0\rangle, \langle q_1, r_1\rangle, \langle q_1, r_2\rangle\} \\ \mathcal{R}_\triangledown &= \{\langle r_0, r_0\rangle, \langle r_1, r_2\rangle, \langle r_2, r_1\rangle \\ \equiv_{\mathcal{S}}^{A_0} &= id \cup \mathcal{R}_\triangledown \end{aligned}$$

The states that are equivalent in A_1 are shaded. The widened automaton obtained by merging r_1 and r_2 is shown below and is the precise fixpoint for this computation.



$A_\triangledown$

If A_0 and A_3 are used as the widening parameter and candidate, the equivalent states in A_3 are shaded. The structure of the widened automaton differs from that of $A_\triangledown$ above, but the language accepted is the same. Let A_i be the i^{th} automaton in the computation above, obtained by adding a state s_{i+1} and transitions $(s_i, 0, s_{i+1})$ and $(s_{i+1}, 0, s_i)$ to A_{i-1}. For any choice of the widening candidate and parameter, only states s_1 and s_2 are merged, so the widened automaton obtained is always isomorphic to $A_\triangledown$ above.

Lemma 18. *The widening seed $\mathcal{S} = \{\subseteq_p\}$ has the following properties:*

1. *It is not extrapolating for fixpoint computations with DFAs or WDBAs.*
2. *It is extrapolating for fixpoint computations with NFAs.*
3. *Enforces termination but not of all fixpoint computations.*
4. *Let M_∞ be the left canonical automaton representing the fixpoint of a computation, A be the widening candidate and M the widening parameter. If the following conditions hold:*
   - *(a) for all $t, t' \in Q_\infty$, $t \neq t' \Rightarrow Pre(t) \not\subseteq Pre(t')$*
   - *(b) for all $r \in Q_M$, there exists $t \in Q_\infty$ such that $Pre(r) \subseteq Pre(t)$ and for all $t' \in Q_\infty : t \neq t' \Rightarrow Pre(r) \not\subseteq Pre(t')$.*

   the fixpoint computation with widening converges to the precise fixpoint.

Proof. (1) Let r and q be states in a widening parameter M and candidate A respectively. If $r \subseteq_p q$, $Pre(r) \subseteq Pre(q)$. As A is deterministic, for all states $t' \neq t$, $Pre(t) \cap Pre(t') = \emptyset$. It follows that for all $t \neq t', r \not\subseteq_p t'$. Hence, each state in the widening parameter is in a relation with at most

one state in the widening candidate, and the widening equivalence is the identity relation.
(2) See Example 19.
(3) In Example 19, we provided an example of a computation that terminates. For all fixpoint computation with deterministic automata, termination is not enforced.
(4) We proceed in a similar manner as in Lemma 17. It is sufficient to show that for any A, M and A_∞ satisfying the conditions of the lemma, $L(A/\equiv_{\mathcal{S}}^{M}) \subseteq L(A_\infty)$. We show that M has the suffix property and that for any two states q, q' in the same equivalence class of $\equiv_{\mathcal{S}}^{M}$, $g(q) = g(q')$.

By condition 4b, for each state $r \in Q_M$, there exists a unique state in $t \in Q_\infty$ such that $\mathit{Pre}(r) \subseteq \mathit{Pre}(t)$. As there exists $u \in \mathit{Pre}(r)$ such that for all $t' \neq t$ in Q_∞, $u \notin \mathit{Pre}(t')$ and $L(M) \subseteq L(M_\infty)$, it must be that $\mathit{Suff}(r) \subseteq \mathit{Suff}(t)$. For each r, this state t is unique, so each $r \in Q_M$ and consequently M has the suffix property.

Consider an equivalence class of $\equiv_{\mathcal{S}}^{M}$ with $n+1$ states. By definition of $\equiv_{\mathcal{S}}^{M}$, for any q, q' in the equivalence class, there exist two sequences of states $q = q_0, \dots q_k = q' \in Q$ and $r_1, \dots, r_k \in Q_M$ such that $0 \leq k \leq n$ and for each $1 \leq i \leq k$, $r_i \subseteq_p q_{i-1}$ and $r_i \subseteq_p q_i$. We show by induction on k that $g(q) = g(q')$. For the base case, take $k = 0$. Clearly, $g(q) = g(q)$. For the induction hypothesis, assume that for $k = n-1$, $g(q) = g(q_{n-1})$.

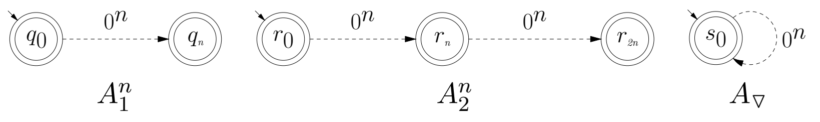
Consider the case $k = n$. There exists $r_n \in Q_M$ such that $r_n \subseteq_s q_{n-1}$ and $r_n \subseteq_s q_n$. As r_n has the suffix property and condition 4b holds, we have that $\mathit{Pre}(r_n) \subseteq \mathit{Pre}(g(r_n))$. By the definition of $\subseteq_p$, we have that $\mathit{Pre}(r_n) \subseteq \mathit{Pre}(q_{n-1})$, which implies that there exists $u \in \mathit{Pre}(q_{n-1})$ such that $u \in \mathit{Pre}(g(r))$ and for all $t' \neq g(r)$ in Q_∞, $u \notin \mathit{Pre}(t')$. Hence, there exists a unique state $g(r)$ in Q_∞ such that $\mathit{Pre}(q_{n-1}) \subseteq \mathit{Pre}(g(r))$ and $\mathit{Suff}(q_{n-1}) \subseteq \mathit{Suff}(f(r))$. It follows that q_{n-1} has the suffix property and that $g(q_{n-1}) = g(r)$. By a similar argument, we have that $g(q_n) = g(r)$ and hence, $g(q_n) = g(q_{n-1})$ as required. We have shown that for any two states q, q' in the same partition of the widening equivalence, the suffix property holds and $g(q) = g(q')$.

It follows from Lemma 11 that $L(A/\pi) \subseteq L(M_\infty)$. □

The next widening seed we consider is $\{\cap_s\}$. If two states are in the relation $\subseteq_s$, they are also in the relation $\cap_s$, thus we may already infer some properties about this widening seed from what we have proved so far. For example, as $\{\subseteq_s\}$ is extrapolating, so is $\{\cap_s\}$. If $\{\subseteq_s\}$ enforces termination for some fixpoint computation, so does $\{\cap_s\}$. For precision, the implication of results is in the other direction. That is, if a class of fixpoint computations with widening using $\{\cap_s\}$ converges to the precise fixpoint, the computation converges to the precise fixpoint if $\{\subseteq_s\}$ is used as the widening seed.

Angluin [1982] identified a subclass of regular languages called the reversible languages and provided an algorithm for inferring reversible languages from positive data. The algorithm is developed for *zero-reversible languages* and generalised to the entire class of reversible languages. A regular language L is zero-reversible iff the left-canonical automaton for L is deterministic. We show that the set of fixpoints that can be computed precisely using $\{\cap_s\}$ for any choice of the widening candidate and parameter is the set of zero-reversible languages. In fact, the algorithm of [Angluin 1982] for inferring a zero-reversible language from a sequence of examples corresponds precisely to a fixpoint computation with widening using $\{\cap_s\}$ as the widening seed. Thus, we see that another algorithm for manipulating automata in a fixpoint computation is an instance of our framework.

Example 20. We use the family of automata from the proof of Lemma 16, part 4. Two automata A_1^n and A_2^n from the fixpoint computation are shown below. The automaton A_i^n accepts the set of words $\{0^{nj} | 0 \leq j \leq i\}$.

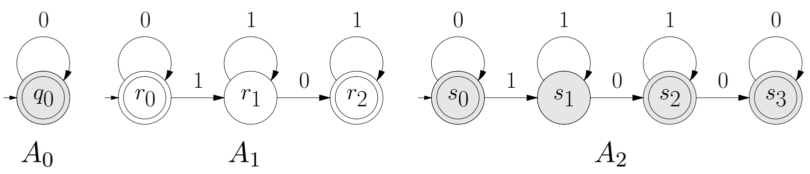


The state q_0 has two suffixes λ and 0^n. Every other state q_i has exactly one suffix, 0^{n-i}. Let A_1^n be the widening seed and A_2^n the widening parameter. The relations for computing the widening equivalence are:

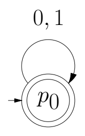
$$\begin{aligned} \cap_s &= \{\langle q_i, r_i\rangle, \langle q_i, r_{n+i}\rangle, \langle q_n, r_0\rangle\} \text{ where } 0 \leq i \leq n \\ \mathcal{R}_\nabla &= \{\langle r_i, r_i\rangle, \langle r_i, r_{i+n}\rangle\} \text{ where } 0 \leq i \leq n \\ \equiv_{\mathcal{S}}^{A_0} &= \mathcal{R}_\nabla \end{aligned}$$

The widened automaton is obtained by merging state r_i with r_{i+n} and r_0 with r_{2n} to obtain the automaton A_∇, which is the precise fixpoint. Recall that this fixpoint could not be computed precisely using the widening seed $\{=_p^k\}$.

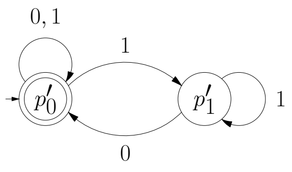
Example 21. We consider a fixpoint computation for which the widened automaton over-approximates the fixpoint.



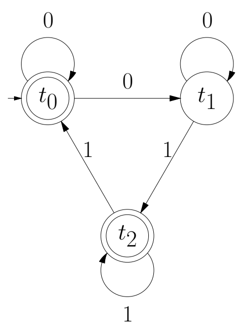
Let A_0 be the widening parameter and A_2 the widening seed. A_0 accepts the language 0^*. Observe that each state in A_2 accepts the word 0, so for all $0 \leq i \leq 3$, $\mathit{Suff}(q_0) \cap \mathit{Suff}(s_i) \neq \emptyset$. Every pair of states in A_2 is related by the widening equivalence and the widened automaton A_∇ is shown below. Let us compare this result with that obtained using the widening seed $\{\subseteq_s\}$. The states s_0, s_2 and s_3 all accept the language 0^* accepted by q_0 and are merged. The widened automaton is shown as A'_∇ below. In both cases, the widened automaton accepts Σ^*.



A_∇



A'_∇



A_∞

If A_1 is used as the widening parameter instead, the widened automaton isomorphic to A_∇. There is no choice of the widening candidate and parameter in this computation that improves the precision of the automaton computed by widening.

Lemma 19 (Properties of $\{\cap_s\}$). *Let $\mathcal{S}$ be $\{\cap_s\}$. The widening seed $\mathcal{S}$ has the following properties:*

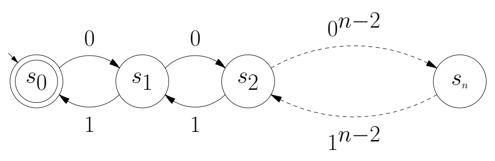
1. *It is extrapolating.*
2. *A_∇ has at most one final state.*
3. *Enforces termination but not for all computations.*
4. *Let A_∞ be the canonical automaton with states Q_∞ representing a regular fixpoint. Let M be a widening parameter with states Q_M and A be the widening candidate with states Q. If the following conditions hold:*

   (a) for all distinct $t, t' \in Q_\infty$, $\mathit{Suff}(t) \cap \mathit{Suff}(t') = \emptyset$

   (b) for all $r \in Q_M$, there exists $t \in Q_\infty$ such that $\mathit{Suff}(r) \subseteq \mathit{Suff}(t)$

   the fixpoint computation with widening converges to the precise fixpoint.

Proof. (1) See Examples 20 and 21.
(2) Let r be a final state in M. For all final states q in A, $\mathit{Suff}(r) \cap \mathit{Suff}(q) = \{\lambda\}$, so all final states are related and hence in the same equivalence class of the widening equivalence. Thus, the quotient automaton has only one final state.
(3) The computations in Examples 20 and 21 terminate with widening. For a non-terminating computation, we reproduce an automaton from Example 18.



A_n

Let A_n be the n^{th} automaton in a fixpoint computation. For any pair of states s_i, s_j, $\mathit{Suff}(s_i) \cap \mathit{Suff}(s_j) = \emptyset$. The widening equivalence is the identity relation and the computation does not terminate.
(4) The sequence of arguments we use is similar to that in Lemma 17, part 4, so we only present the steps that differ. We show that M has the prefix property and that any two states in the same partition agree on the function f.

By condition 4b, for each state $r \in Q_M$, there exists $t \in Q_\infty$ such that $\mathit{Suff}(r) \subseteq \mathit{Suff}(t)$. As the suffixes of states in A_∞ are disjoint, this state t is unique and as $L(A) \subseteq L(A_\infty)$, it must be that $\mathit{Pre}(r) \subseteq \mathit{Pre}(t)$. We have established that M has the prefix property.

Consider a pair of states q, q' in an equivalence class of $\equiv_S^M$. We only show the inductive step here. Assuming there exists $r \in Q_M$ such that $r \cap_s q$ and $r \cap_s q'$, we need to show that $f(q) = f(q')$. As $\mathit{Suff}(r) \cap \mathit{Suff}(q) \neq \emptyset$, and $\mathit{Suff}(r) \subseteq \mathit{Suff}(f(r))$, we conclude that $\mathit{Suff}(q) \cap \mathit{Suff}(f(r)) \neq \emptyset$. As no state besides $f(r)$ accepts words in this intersection, it must also be that $\mathit{Pre}(q) \subseteq \mathit{Pre}(f(r))$, whereby we further conclude that q has the prefix property and that $f(q) = f(r)$. By a similar argument, we have that $f(q') = f(r)$, hence $f(q) = f(q')$. Convergence to the precise fixpoint follows from Lemma 10. □

How do our results compare to those in [Angluin 1982] about zero-reversible inference? Let ZR-Inference be the algorithm for inference of zero-reversible languages, which takes a sequence of tries as input. Angluin [1982] provides the following theorem about the zero-reversible inference algorithm.

Theorem 9 (Angluin 1982, Theorem 27). *Let $S_0, S_1, \ldots,$ be a sequence of finite, positive samples, ordered by inclusion, of a zero-reversible regular language L. Let $A_0, A_1, \ldots$ be a sequence of tries with A_i accepting the sample S_i. Let $A'_i =$ ZR-Inference$(\{A_0, \ldots, A_i\})$. The sequence $A'_0, A'_1, \ldots$ converges to the canonical automaton accepting L.*

Let us compare the conditions of Lemma 19, part 4 to those of Theorem 9. Condition 4a in Lemma 19 is identical to the requirement that the samples S_i be from a reversible language. Let A_i be a trie as in Theorem 9. As each state in a trie has a unique prefix and the fixpoint of the computation is represented by a deterministic automaton, the trie A_i has the prefix property. As $L(A_i) \subseteq L(M_\infty)$, it follows that for each state q in A_i,

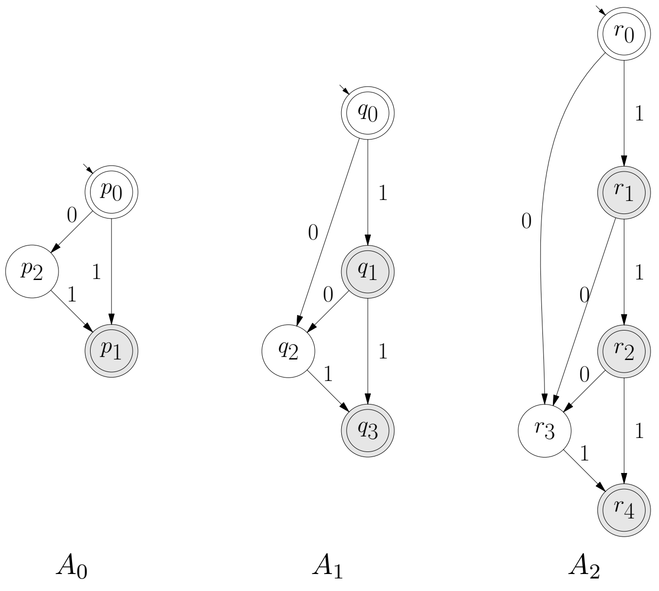


Figure 4.1: Fixpoint computation in Example 4.1

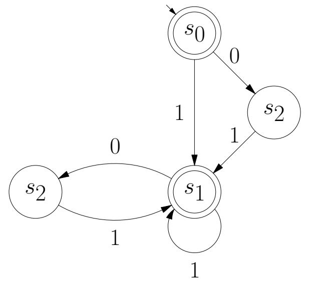
Suff $(q) \subseteq$ *Suff* $(f(q))$. Thus, if a sequence of tries satisfies the conditions of Theorem 9, condition 4b of Lemma 19 holds for each trie in the sequence. If the sequence of tries A_i is viewed as a fixpoint computation, we may say that if conditions of Theorem 9 hold for a fixpoint computation, the conditions of Lemma 19, part 4 hold for that computation. Thus, our result is more general than Theorem 9. To see that it is strictly more general, observe that we do not impose the restriction that the languages of the automata in the fixpoint computation be finite.

The last widening seed we study in this section is $\{\cap_p\}$. As we may expect, widening using this seed can be compared to widening using $\{\subseteq_p\}$. The seed $\{\subseteq_p\}$ is extrapolating for computations with nondeterministic automata and enforces termination and $\{\cap_p\}$ has these properties as well. In addition, the seed $\{\cap_p\}$ is also extrapolating for computations with deterministic automata. In addition, the class of languages for which the fixpoint can be computed precisely is once again the zero-reversible languages.

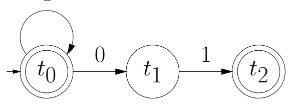
Example 22. The three minimal automata in Figure 4.1 occur in a fixpoint computation. The state p_1 in A_0 has the prefixes $\{1, 01\}$. The states q_1 and q_3 in A_1 have the prefixes $\{1\}$ and $\{01, 11, 101\}$ respectively. Let A_0 be the widening parameter and A_1, the widening candidate. The relations for computing the widening equivalence are:

$$\begin{aligned}
\cap_p &= \{\langle p_0, q_0\rangle, \langle p_2, q_2\rangle, \langle p_1, q_1\rangle, \langle p_1, q_3\rangle\} \\
\mathcal{R}_\nabla &= id \cup \{\langle q_1, q_3\rangle, \langle q_3, q_1\rangle\} \\
\equiv_{\mathcal{S}}^{A_0} &= \mathcal{R}_\nabla
\end{aligned}$$

The states q_1 and q_3 are in the same equivalence class. The widened automaton A_∇ and the fixpoint A_∞ are shown below.



A_∇



A_∞

The fixpoint of the sequence is the language $(1^* + 1^*01)$. The states s_0 and s_1 in A_∇ accept the same language, which is $(1 + 01)^*$, a non-trivial over-approximation of $L(A_\infty)$. If A_2 is used as the widening candidate and A_0 as the widening seed, the states that are shaded in A_2 are in the same equivalence class and the widened automaton is the same. If A_2 is the widening candidate and A_1, the widening parameter, the states r_2 and r_4 in A_2 are merged. The widened automaton is not isomorphic to A_∇ but accepts $L(A_\nabla)$.

Consider the sequence of tries $A'_0, A'_1, \ldots$ such that $L(A'_i) = L(A_i)$ for A_i in the computation above. Each state in any automaton A'_i has exactly one prefix, so a state in one automaton is related to at most one state in any other automaton in the computation and the widened automaton is isomorphic to the widening candidate. Thus, if this widening seed is being used, better over-approximations may be obtained by first minimising the automaton.

Lemma 20 (Properties of $\{\cap_p\}$). *The widening seed $\{\cap_p\}$ has the following properties:*

1. *It is extrapolating.*
2. *If A is the widening parameter and candidate, A_∇ is deterministic.*
3. *It enforces termination, though not for all fixpoint computations.*
4. *Let M_∞ be the left-canonical automaton with states Q_∞ representing a regular fixpoint. Let M be a widening parameter with states Q_M, A be the widening candidate with states Q. If the following conditions hold:*

(a) M_∞ is deterministic

(b) M has the prefix property

the fixpoint computation with widening converges to the precise fixpoint.

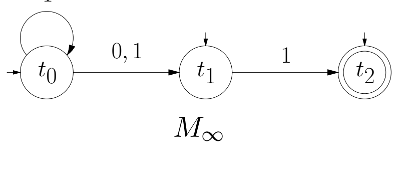
Proof. (1) See Example 22.
(2) Consider a block of the partition of Q induced by the widening equivalence. Let $q_1, \ldots, q_n$ the states therein. The prefixes of the state $[q_1]$ in the widened automaton are $Pre(\{q_1, \ldots, q_n\}) \cdot L(\{q_1, \ldots, q_n\})$. For any two states q and q' in different equivalence classes, $Pre(q) \cap Pre(q') = \emptyset$. Therefore, $Pre([q]) \cap Pre([q']) = \emptyset$. As the states in the widened automaton have disjoint prefixes, it is deterministic.

As A is the widening parameter, every state is related to itself. In addition, every pair of states with a common prefix are in the same equivalence class. The states in different
(3) See Example 22 for a fixpoint computation that terminates with widening. Consider any fixpoint computation with tries. As each state in each automaton has exactly one prefix, a state in one automaton is related to at most one state in any other automaton. The computation with widening is identical to the one without, so any non-terminating fixpoint computation does not terminate with widening.
(4) The initial steps are similar to the proof of Lemma 18, part 4. We only show that any two states in the an equivalence class of $\equiv_S^M$ have the suffix property and that $g(q) = g(q')$.

By condition 4b, for each state $r \in Q_M$, there exists $t \in Q_\infty$ such that $Pre(r) \subseteq Pre(t)$. As the prefixes of states in M_∞ are disjoint, this state t is unique and as $L(A) \subseteq L(A_\infty)$, it must be that $Suff(r) \subseteq Suff(t)$. We have shown that M has the suffix property.

Consider a pair of states q, q' in an equivalence class of $\equiv_S^M$. Assuming there exists $r \in Q_M$ such that $r \cap_p q$ and $r \cap_p q'$, we need to show that $g(q) = g(q')$. As $Pre(r) \cap Pre(q) \neq \emptyset$, and $Pre(r) \subseteq Pre(g(r))$, we conclude that $Pre(q) \cap Pre(g(r)) \neq \emptyset$. As no state besides $g(r)$ has prefixes in this intersection, it must also be that $Suff(q) \subseteq Suff(f(r))$, whereby we further conclude that q has the suffix property and that $g(q) = g(r)$. By a similar argument, we have that $g(q') = g(r)$, hence $g(q) = g(q')$. Convergence to the precise fixpoint follows from Lemma 11. □

Consider the language of A_∞ in Example 22. The left canonical automaton for this language is shown below.



M_∞

Recall that no choice of the widening parameter or candidate allowed for the fixpoint to be computed precisely. We can see that the second condition of Lemma 20, part 4 does not hold as M_∞ is not deterministic.

We conclude this section with a discussion on other widening seeds, in particular, how we use results about elementary widening seeds to study other widening seeds. We only comment on but do not analyse the use of simulation relations as widening seeds. Simulation relations and language inclusion are identical notions for deterministic automata, so the widening seed using direct simulation $\{\sqsubseteq^{di}\}$, has the same properties as $\{\subseteq_s\}$ for deterministic automata.

Additional widening seeds can be constructed from the union or intersection of the relations we introduced in Section 4.1. Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two binary relations in $\mathcal{U}$. If $\mathcal{R}_1 \subseteq \mathcal{R}_2$, the widening seed $\{\mathcal{R}_2\}$ has stronger extrapolation and termination properties and weaker precision properties than $\{\mathcal{R}_1\}$. That is, if a widening step using $\{\mathcal{R}_1\}$ is extrapolating or enforces termination, the widening step using $\{\mathcal{R}_2\}$ is also extrapolating and also enforces termination. As $\mathcal{R}_2$ may be a larger relation than $\mathcal{R}_1$, more states in the widening candidate might be related and a widening step using $\{\mathcal{R}_1\}$ that computes the precise fixpoint may only compute an over-approximation if $\{\mathcal{R}_2\}$ is used. The converse though is true. If a computation with widening converges to the precise fixpoint using $\{\mathcal{R}_2\}$, it converges to the precise fixpoint if $\{\mathcal{R}_1\}$ is used as the widening seed.

These observations are useful for constructing other widening operators. For example, if a widening seed $\mathcal{S}_1$ does not enforce termination for a class of computations, we can choose a widening seed $\mathcal{S}_2 \supseteq \mathcal{S}_1$ that does. On the other hand, if the widening seed using a relation $\mathcal{R}_1$ introduces too much imprecision, a more precise widening seed can be construct using the relation $\mathcal{R} = \mathcal{R}_1 \cap \mathcal{R}_2$. As fewer states are related, the imprecision introduced decreases.

4.4 The Bartzis-Bultan Widening Seed

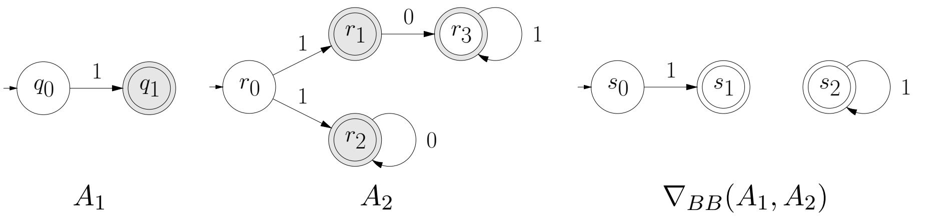
Bartzis and Bultan [2004] propose a widening operator that uses the relations $\cap_p$ and $=_s$ for accelerating computations with automata encoding arithmetic. Though these relations are used, we show that the widening operator and widened automaton proposed in [Bartzis and Bultan 2004] differ from those in our framework. Despite the difference, we refer to $\{\cap_p, =_s\}$ as the Bartzis-Bultan (BB) widening seed. Our analysis proceeds as in Section 4.3. We discuss and demonstrate the effect of widening with a few examples and make a formal statement about the properties of the widening seed.

Bartzis and Bultan [2004] define a widening relation, denoted $\equiv_\nabla$, using two automata A_1 and A_2 and claim it is an equivalence relation. Consider the relation $\mathcal{R}_\triangledown$, constructed from the widening seed $\{\cap_p, =_s\}$ using Construct Equivalence. The relation $\equiv_\triangledown$ as defined in [Bartzis and Bultan 2004] is *any* transitive relation that includes $\mathcal{R}_\triangledown$. Thus, if $\mathcal{R}_\triangledown$, is not reflexive, as per the definition in [Bartzis and Bultan 2004], $\equiv_\triangledown$ need not be either. Even if a $\equiv_\triangledown$ is required to be reflexive, it may not be well defined. A second problem arises in the definition of the widened automaton.

Definition 25 (Bartzis and Bultan 2004, Widened Automaton). Let $A_1 = (Q_1, \Sigma, \delta_1, r_0, F_1)$ and $A_2 = (Q_2, \Sigma, \delta_2, t_0, F_2)$ be two trim, deterministic automata in a fixpoint computation and $\pi \subseteq Q_1 \cup Q_2$ be the partition induced by the widening relation $\equiv_\triangledown$. The widened automaton $\nabla_{BB}(A_1, A_2) = (Q, \Sigma, \delta, q_0, F)$ is defined as follows: $Q = \{[q] | q \in Q_1 \cup Q_2\}$, $q_0 = [r_0]$, $F = \{[q] | q \in F_1 \cup F_2\}$ and $\delta([q], a) = [q']$ where $\forall r \in Q_1 \cap [q] : \delta(r, a) \in [q']$ and $\forall t \in Q_2 \cap [q] : \delta(t, a) \in [q']$.

Examine the definition of δ in the widened automaton. This definition is problematic because a transition on a symbol a between two states $[q]$ and $[q']$ in the widened automaton is defined only if the transition from *every state* in $[q]$ on the symbol a, if defined, ends in a state in $[q']$. If this condition is not satisfied, a transition that exists in an automaton may not exist in the widened automaton, which has the drastic consequence that the language of the widened automaton may not include the language of the widening candidates. The following example illustrates a computation with widening in which this occurs.

Example 23. Two automata in a fixpoint computation are shown below.



The relations computed are shown below.

$$
\begin{aligned}
\cap_p &= \{\langle q_0, r_0\rangle, \langle q_1, r_1\rangle, \langle q_1, r_2\rangle\} \\
=_s &= \emptyset \\
\mathcal{R}_\triangledown &= \{\langle r_0, r_0\rangle, \langle r_1, r_1\rangle, \langle r_1, r_2\rangle, \langle r_2, r_1\rangle, \langle r_2, r_2\rangle\} \\
[\equiv_\triangledown] &= \mathcal{R}^*_\triangledown = \{\{r_0\}, \{r_1, r_2\}, \{r_3\}\}
\end{aligned}
$$

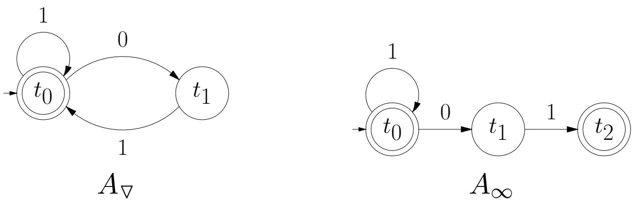
There are three equivalence classes. The states in the same equivalence class have the same shading pattern. The automaton computed using Definition 25 is also shown. It is obvious that language of this automaton does not include the language of A_2.

The BB-widening seed is constructed from two relations we have studied. If a widened fixpoint computation using the widening seed $\{\Cap_p\}$ is extrapolating, so is computation using the BB-widening seed. In addition, if the widened automaton computed using $\{\Cap_p\}$ over-approximates the fixpoint, so does the widening step using the BB-widening seed. We use the fixpoint computation from Example 22 to illustrate.

Example 24. Let $\mathcal{S}$ be the BB-widening seed. Consider the automata A_0 and A_1 in the fixpoint computation in Example 22. The relation $\mathcal{R}_\triangledown$ is computed using $\Cap_p$ and $=_s$. The relation $\Cap_p$ is identical to that in Example 22 and $=_s$ is as shown below.

$$\begin{aligned}
\Cap_p &= \{\langle p_0, q_0\rangle, \langle p_2, q_2\rangle, \langle p_1, q_1\rangle, \langle p_1, q_3\rangle\}\\
=_s &= \{\langle p_0, q_1\rangle, \langle p_2, q_2\rangle, \langle p_1, q_3\rangle\}\\
\mathcal{R}_\triangledown &= \left(\Cap_p^{-1} \circ \Cap_p\right) \cup \left(=_s^{-1} \circ \Cap_p\right) \cup \left(\Cap_p^{-1} \circ =_s\right) \cup \left(=_s^{-1} \circ =_s\right)\\
&= id \cup \{\langle q_1, q_3\rangle, \langle q_3, q_1\rangle\} \cup \{\langle q_0, q_1\rangle, \langle q_1, q_0\rangle\}\\
\left[\equiv_{\mathcal{S}}^{A_1}\right] &= \{\{q_0, q_1, q_3\}, \{q_2\}\}
\end{aligned}$$

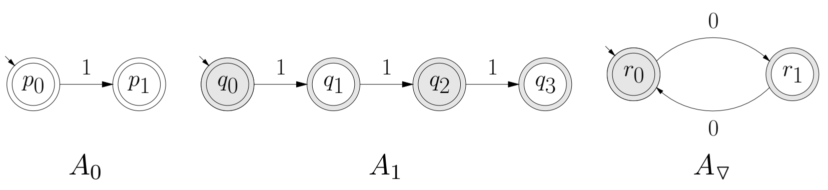
The states related by the two relations in $\mathcal{S}$ are different. The pair $\langle p_0, q_1\rangle$ is in the relation $=_s$ but not in $\Cap_p$. Consequently, $\mathcal{R}_\triangledown$ differs from Example 22. We explicitly show how $\mathcal{R}_\triangledown$ is constructed from the widening seed, following the steps in Algorithm 2. The relation $(id \cup \{\langle q_1, q_3\rangle, \langle q_3, q_1\rangle\})$ is computed using only $\Cap_p$. The pair $\langle q_0, q_1\rangle$ is in the relation $\left(\Cap_p^{-1} \circ =_s\right)$. We only show the equivalence classes of the widening equivalence. All the final states in A_1 are related and the equivalence partition is coarser than in Example 22. The widened automaton is shown below.



The language accepted by $A_\triangledown$ is identical to that accepted by the widened automaton in Example 22 but the automaton obtained here is minimal. For any choice of the widening parameter and candidate in this computation, the widening automaton obtained is isomorphic to $A_\triangledown$ above.

Recall from Lemma 13, part 2 that if $\{=_s\}$ is the widening seed and the same automaton is the widening parameter and candidate, the widened automaton is minimal. The same holds for the BB-widening seed. In general, the widened automaton may not be minimal.

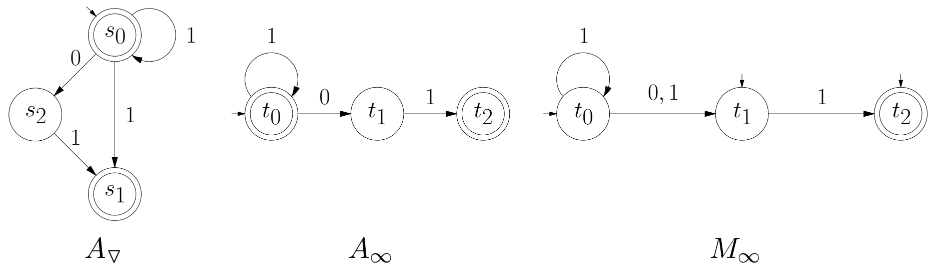
Example 25. Widening is applied to the fixpoint computation shown below, using the BB-widening seed.



The relation $\cap_p$ is $\{\langle r_0, t_0\rangle, \langle r_1, t_1\rangle\}$ and $=_s$ is $\{\langle r_0, t_2\rangle, \langle r_1, t_3\rangle\}$. The states in the same equivalence class have identical shading patterns. The widened automaton $A_\triangledown$ obtained by merging these states is deterministic but not minimal.

We provide a third example of a computation with widening that terminates with the exact fixpoint being computed.

Example 26. The tries A_0, A_1 and A_2 in Figure 4.2 represent the same fixpoint computation as in Example 24. Let A_0 be the widening parameter and A_1 be the widening candidate. The states p_0 and q_0 both have the prefix $\{\lambda\}$. As both A_0 and A_1 as tries, each state has exactly one prefix. The relation $\cap_p$ is $\{\langle p_i, q_i\rangle\}$ for $0 \leq i \leq 3$. The relation $=_s$ relates states with the same suffix. Of these, the pair $\langle p_0, q_1\rangle$ is of interest. The tries from the states p_0 and q_1 are isomorphic. The states q_0 and q_1 are in the same class of the widening equivalence as they both relate to p_0. The widened automaton is also shown below.



The widened automaton accepts the same language as A_∞. Unlike the widened automaton in the previous example, $A_\triangledown$ is nondeterministic but minimal. Unlike Example 24, the fixpoint is computed precisely! The choice of the automata is not important. Given any widening candidate, if any automaton that appeared previously in the computation is used as the widening parameter, the language accepted by the widened automaton is the same.

Our observation in the previous example, deserves emphasis. Given a sequence of languages, admitting different representations as automata, the effect of widening is heavily dependent on the structure of the automaton. In Example 24, we considered a sequence of minimal automata representing the sets in a fixpoint computation and saw that for any choice of the widening parameter and candidate, the fixpoint computation terminated and the

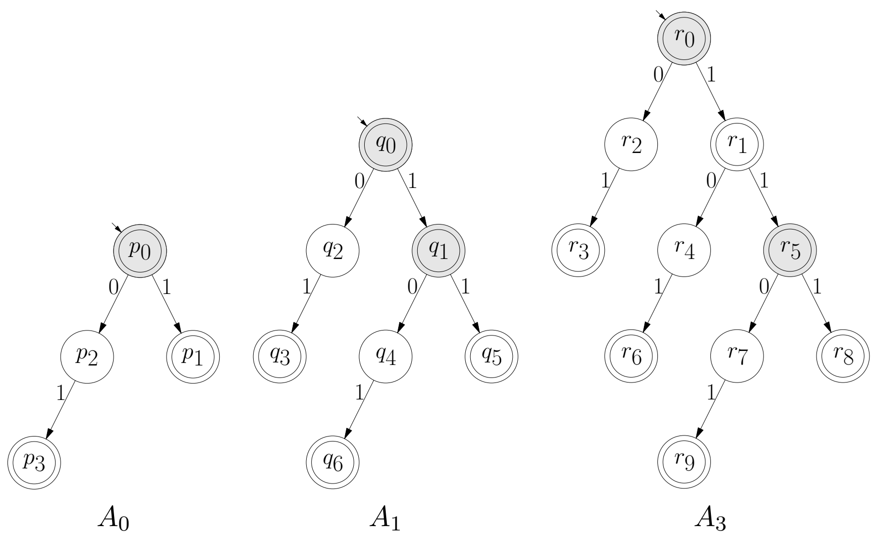


Figure 4.2: Automata in the fixpoint computation in Example 26

language of the widened automaton was an over-approximation of the fixpoint. Further, no choice existed for which the precise fixpoint could be computed. In Example 26, we considered the *same* fixpoint computation using tries instead of minimal automata to represent the sets. In this case, for any choice of the widening seed and widening parameter, the widened automaton accepts the language of the precise fixpoint.

Lemma 21 (Properties of $\{\cap_p, =_s\}$). *Let $\mathcal{S}$ be the BB widening seed. It has the following properties:*

1. *It is extrapolating.*
2. *If A is deterministic and is the widening parameter, then A_∇ is minimal.*
3. *If A is minimal and is the widening parameter, then A_∇ is deterministic.*
4. *It enforces termination, though not for all fixpoint computations.*
5. *Let M_∞ be the left-canonical automaton with states Q_∞ representing a regular fixpoint. Let M be a widening parameter with states Q_M, A be the widening candidate with states Q. If the following conditions hold:*
   - *(a) M_∞ is deterministic*
   - *(b) M has the prefix property*

the fixpoint computation with widening converges to the precise fixpoint.

Proof. (1) See Example 24.
(2) The states in A have disjoint prefixes, so $\cap_p$ is the identity relation. The widening equivalence is identical to that obtained with the widening seed $\{=_s\}$. It follows from Lemma 13, part 2 that A_∇ is minimal.
(3) If A is minimal, $=_s$ is the identity relation and the widening equivalence is determined by $\cap_p$. It follows from Lemma 20, part 2 that the widened automaton is deterministic.
(4) The computation in Example 24 terminates with widening. For an example where termination is not enforced, consider the sequence of automata A_n in Example 18, for which the fixpoint is context-free. As the prefixes and suffixes of each state in each automaton are unique, for any choice of the widening parameter and candidate, the widening equivalence is the identity relation. Widening does not enforce termination of this computation.
(5) We proceed in a similar manner as previous precision proofs. We have shown in Lemma 20, part 4 that given conditions 5a and 5b, M has the suffix property. We show that for each state $r \in Q_M$, $f(r) = g(r)$. Then, we show that all states in an equivalence class of $\equiv_{\mathcal{S}}^{M}$ agree on both f and g.

For each state $r \in Q_M$, by condition 5b, $f(r)$ is defined. As M_∞ is deterministic and $L(M) \subseteq L(M_\infty)$, $\mathit{Suff}(r) \subseteq \mathit{Suff}(f(r))$. As the suffixes of states in M_∞ are disjoint, $g(r)$ is also defined and $f(r) = g(r)$.

Consider an equivalence class of $\equiv_{\mathcal{S}}^{M}$ with $n+1$ states. By definition of $\equiv_{\mathcal{S}}^{M}$, for any q, q' in the equivalence class, there exist two sequences of states $q = q_0, \ldots q_k = q' \in Q$ and $r_1, \ldots, r_k \in Q_M$ such that $0 \leq k \leq n$ and for each $1 \leq i \leq k$, either $r_i \cap_p q_{i-1}$ or $r_i =_s q_{i-1}$. The states r_i and q_i are similarly related.

We show by induction on k that $f(q) = g(q) = g(q') = f(q')$. For the base case, take $k = 0$. As a singleton equivalence class does not affect the language accepted, we are done. For the induction hypothesis, assume that for $k = n-1$, $f(q) = g(q) = g(q_{n-1}) = f(q_{n-1})$. For the case $k = n$, we consider two possibilities for r_n and q_{n-1}

1. If $r_n \cap_p q_{n-1}$, as M_∞ is deterministic, it holds that $\mathit{Suff}(q_{n-1}) \subseteq \mathit{Suff}(f(r_n))$. As M_∞ is left-canonical, for all $t \in Q_\infty$ such that $t \neq f(r_n)$, we have that $\mathit{Suff}(q_{n-1}) \cap \mathit{Suff}(t) = \emptyset$. Thus, $g(q_{n-1})$ is defined and is $f(r_n)$. As M_∞ is deterministic, it also holds that $\mathit{Pre}(q_{n-1}) \subseteq \mathit{Pre}(g(q_{n-1})$ and that $g(q_{n-1})$ is unique. Hence, $f(q_{n-1})$ is defined and $f(q_{n-1}) = g(q_{n-1})$.

2. If $r_n =_s q_{n-1}$, then $\mathit{Suff}(r_n) = \mathit{Suff}(q_{n-1})$ and as M has the prefix property, $\mathit{Suff}(q_{n-1}) \subseteq g(r_n)$. As M_∞ is left canonical, this state is

unique and $g(q_{n-1}) = g(r_n)$. As $L(A) \subseteq L(M_\infty)$, we also have that $Pre(q_{n-1}) \subseteq Pre(g(q_{n-1})$ and from M_∞ being deterministic, $f(q_{n-1})$ is defined and is $g(q_{n-1})$.

In both cases, $f(r_n) = f(q_{n-1}) = g(q_{n-1})$. By the same argument, we can show that $f(r_n) = f(q_n) = g(q_n)$. As all state in the same partition agree on f, we have from Lemma 10 that $L(A/\pi) \subseteq L(M_\infty)$. Convergence to the precise fixpoint follows. $\square$

4.5 The Boigelot-Legay-Wolper Widening Seed

Boigelot et al. [2003] propose a widening seed that uses the relations $=_s$ and $=_p$. Their technique is aimed at accelerating fixpoint computations with transducers is not explicitly called widening. The acceleration algorithm in [Boigelot et al. 2003] involves comparing the states in a sequence of minimal, deterministic transducers using simulation relations. As the automata are deterministic and simulation relations are used, the comparisons can equivalently be made with the relations $=_s$ and $=_p$. The acceleration step that is proposed is not a quotient construction but involves adding transitions to the automaton. We conjecture that the effect of the construction proposed by Boigelot et al. [2003] when applied to any pair of automata in a fixpoint computation is the same as applying widening as defined in our framework using $\{=_p, =_s\}$ as the widening seed. We refer to this widening seed as the BLW-widening seed.

It may seem counter-intuitive that such a widening seed can be useful, since we have seen in Lemmas 4 and 14 that the seeds $\{=_s\}$ and $\{=_p\}$ are non-extrapolating and do not ensure termination for any fixpoint computations. However, Boigelot et al. [2003, Table 1] contains several examples of fixpoint that were precisely computed using the BLW-seed. In addition, we prove that for any computation with a regular fixpoint, the computation using the BLW-widening seed converges in the limit to the precise fixpoint. In [Boigelot et al. 2003], precise convergence of computations using the BLW-seed is reduced to a synchronisation problem for automata with counters and a statement is made without proof. We provide a direct proof similar to previous proofs of precision.

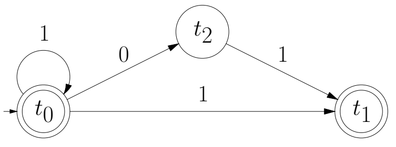
Consider the fixpoint computation in Example 26. The sets in the computation are represented as tries. Each state in a trie has a unique prefix, so if a pair of states in different automata are related by $\cap_p$, they have the same prefix. That is, for states r and q in different tries, $r \cap_p q \Leftrightarrow r =_p q$. Thus, for all fixpoint computations using tries as representations, the widened computations using the BB-seed and the BLW-seed are identical. We find it justified to make the bold claim that such an observation is a direct consequence of the framework we use. Boigelot et al. [2003] do observe that comparing automata occurring in a fixpoint computation "*makes our technique similar* *to the widening technique found in [Bouajjani et al. 2000; Touili 2001]*". Bartzis and Bultan [2004], in comparing their work to [Boigelot et al. 2003] say: "*Our technique is also generic but is based on widening instead of iterating relations.*" Despite these statements, no precise statement about the connection between the two techniques has been made.

As the BLW-widening seed and the BB-widening seed are identical for computations with tries, Example 26 also illustrates the operation of the BLW-widening seed. For a second example, we consider the same computation using minimal automata. Recall that neither the BB-widening seed nor $\{\cap_p\}$ could be used to compute the precise fixpoint of this computation.

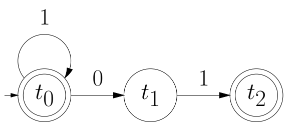
Example 27. Consider the fixpoint computation in Example 22. Let A_0 be the widening parameter and A_1, the widening candidate. The relations required to compute the widening equivalence are:

$$\begin{aligned}
=_p &= \{\langle p_0, q_0\rangle\} \\
=_s &= \{\langle p_0, q_1\rangle, \langle p_2, q_2\rangle, \langle p_1, q_3\rangle\} \\
\mathcal{R}_\nabla &= id \cup \{\langle q_0, q_1\rangle, \langle q_1, q_0\rangle\} \\
\left[\equiv_{\mathcal{S}}^{A_0}\right] &= \{\{q_0, q_1\}\{q_2\}\{q_3\}\}
\end{aligned}$$

Only two states have identical prefixes and as p_0 and q_0 have identical suffixes, the states q_0 and q_1 are in the same equivalence class. The widened automaton is shown below.



A_∇



A_∞

We see that $L(A_\nabla) = L(A_\infty)$. Recall that this fixpoint could not be computed precisely using $\{\cap_p\}$ or the BB-seed for any choice of the widening parameter or candidate.

Lemma 22 (Properties of $\{=_p, =_s\}$). *Let $\mathcal{S}$ be the BLW widening seed. It has the following properties:*

1. *It is extrapolating.*
2. *If A is the widening parameter, then A_∇ is minimal.*
3. *It enforces termination, though not for all fixpoint computations.*
4. *Let N_∞ be an automaton with states Q_∞ representing a regular fixpoint. Let M be a widening parameter with states Q_M, A be the widening candidate with states Q. If for each $r \in Q_M$, $F(r)$ is defined and $F(r) = G(r)$ the computation with widening converges to the precise fixpoint.*

Proof. (1) See Examples 26 and 27.
(2) If A is the widening parameter, all states in A with the same suffixes are related and the widened automaton is minimal.
(3) The computations in Examples 26 and 27 terminate with widening. The BLW seed has weaker termination properties than the BB-widening seed $\cap_p, =_s$. As the BB-seed does not enforce termination, neither does the BLW-seed.
(4) Let M be the widening parameter with states Q_M, A be the widening candidate with states Q and N_∞ be an automaton with states Q_∞ representing the fixpoint satisfying that for each $r \in Q_M$ there exists $t \in Q_\infty$ such that $Pre(r) \subseteq Pre(t)$ or $Suff(r) \subseteq Suff(t)$. We know from Lemma 12 that N_∞ exists. To examine the language of the widened automaton, it is sufficient to $Pre([q]) \cdot Suff([q])$ for each $q \in Q$.

Consider the equivalence class $[q]$ in the widened automaton with $n+1$ states. By definition of $\equiv_{\mathcal{S}}^{M}$, for any $q, q' \in [q]$, there exist two sequences of states $q = q_0, \ldots q_k = q' \in Q$ and $r_1, \ldots, r_k \in Q_M$ such that $0 \leq k \leq n$ and for each $1 \leq i \leq k$, $r_i \subseteq_s q_{i-1}$ and $r_i \subseteq_s q_i$. We show by induction on k that $Pre([q]) \cdot Suff([q]) \subseteq L(N_\infty)$. For the base case, consider $k = 0$. If the equivalence class has only one state, $Pre([q]) = Pre(q)$ and $Suff([q]) = Suff(q)$ and the language of the automaton does not change. For the induction hypothesis, assume that for $k = n-1$, one of the following conditions holds: $F(q) = F(q_{n-1})$ or $F(q) = G(q_{n-1})$ or $G(q) = F(q_{n-1})$ or $G(q) = G(q_{n-1})$. For the case $k = n$, we show that states q_{n-1} and q_n also satisfy these conditions.

The state r_n is related to q_{n-1} and q_n by either $=_s$ or $=_p$. If $r_n =_s q_n$, then $F(r_n) = F(q_n)$ and if $r_n =_p q_n$, then $G(r_n) = G(q_n)$. The same holds for q_{n-1}. We only consider the case $r =_s q_n$ and $r =_p q_{n-1}$. All other cases are similar. We have that $G(r_n) = G(q_n)$ and $F(r_n) = F(q_{n-1})$ and by the condition of the lemma, $F(r_n) = G(r_n)$, hence $F(q_n) = G(q_{n-1})$.

We have shown that for any states q and q' in an equivalence class of $\equiv_{\mathcal{S}}^{M}$, either $F(q) = F(q')$ or $G(q) = G(q')$. From Theorem 7, we have that if this condition holds $L(A/\equiv_{\mathcal{S}}^{M}) \subseteq L(N_\infty)$, which completes the proof. □

Chapter 5

Approximating Fixpoint Computations

We have introduced various widening seeds and studied their extrapolation, termination and precision properties. In the previous chapter, we illustrated the effect of widening on various least fixpoint computations. In this chapter, we show how widening can be used in other fixpoint computations. We begin with the use of dual widening for under-approximating greatest fixpoint computations and then study the use of widening and dual widening for model checking.

5.1 Dual Widening for Greatest Fixpoint Computations

Our focus till this point has been on least fixpoint computations. The lattice theoretic setting we use allows us to apply the duality principle and obtain results for greatest fixpoint computations. Recall the duality principle that the dual of any statement about a partially ordered set S also holds for S. If a statement is true of a sequence of automata, the dual statement holds for the sequence of complemented automata. We begin by briefly stating the dual of the problem we have studied so far and restrict our attention to fixpoint computations with regular languages.

The dual of an over-approximation of a least fixpoint is and under-approximation of a greatest fixpoint. Convergence of the greatest fixpoint computation is accelerated and termination enforced using dual-widening. If we take a learning theory view, a greatest fixpoint computation with dual-widening can be viewed as learning with subset queries using an oracle that either returns `Yes` or a negative example. This problem is also related to inductive inference of a regular language from a sequence of negative examples. Algorithms for inference of regular languages from negative examples, if they exist, can be used to design dual-widening operators. Conversely, a dual-widening operator that can be used to precisely compute a class of fixpoints directly provides an inference algorithm for that class of regular languages.

The greatest fixpoint computations we consider are infinite sequences of regular languages of the form, $L_0, L_1, \ldots$ such that L_0 is the initial set, and for each $i \in \mathbb{N}$ $L_i \supseteq L_{i+1}$. The languages are represented by a sequence of automata $A_0, A_1, \ldots$, where the automaton A_i accepts the language L_i. A dual-widening candidate A_i is an automaton in a fixpoint computation. A dual-widening parameter is an automaton A_j such that $j \leq i$. A dual-widening seed $\mathcal{S}$ is identical to a widening seed and is combined with A_j to construct an equivalence relation between the states of A_i. As we make no assumption about the language of the two automata for constructing the equivalence relation, this step does not change. Let M denote the dual-widening parameter and $\equiv_{\mathcal{S}}^{M}$ the dual-widening equivalence. The final step is computing the dual-widened automaton. For least fixpoint computations, the quotient automaton $A_i / \equiv_{\mathcal{S}}^{M}$ was defined as the widened automaton. If the aim is to extrapolate a greatest fixpoint computation, we have to decrease the language accepted by the widening candidate. Let $\overline{A}_i$ denote the complement of a complete, deterministic automaton A_i. The dual-widened automaton is defined as $\overline{(\overline{A}_i / \equiv_{\mathcal{S}}^{M})}$. The automaton $\overline{A}_i$ accepts the complement of the language $L(A_i)$ and as $L(\overline{A}_i) \subseteq L(\overline{A}_i / \equiv_{\mathcal{S}}^{M})$, we have that $L(A_i) \supseteq L\left(\overline{(\overline{A}_i / \equiv_{\mathcal{S}}^{M})}\right)$. This definition of the dual widening operator satisfies the convergence property of Definition 10 for dual widening operators. However, the nested complement operations obscure the changes in the structure of the automaton A_i. We propose an equivalent *dual quotient* operation for use with greatest fixpoint computations. Let $q_\bot$ be the unique *sink* state in any automaton. That is, $q_\bot$ is not final and for all $a \in \Sigma$, $\delta(q_\bot, a) = q_\bot$. The dual notion is the *universal state* $q_\top$, a final state such that for all $a \in \Sigma$, $\delta(q_\top, a) = q_\top$. Observe that $L(q_\bot) = \emptyset = \overline{\Sigma^*} = \overline{L(q_\top)}$.

Definition 26 (Dual Quotient). Let $A = (Q, \Sigma, \delta, q_0, F)$ be a complete finite automaton and π a partition of Q. The dual quotient $A \backslash \pi$ is the automaton $(Q_\pi, \Sigma, \delta_\pi, [q_0], F_\pi)$, where $Q_\pi = \{[q] | q \in Q\}$, $F_\pi = \{[q] | [q] \subseteq F\}$ and $\delta_\pi([q], a) = \{[q'] | \forall q \in [q] : \delta(q, a) \subseteq [q'] \cup \{q_\top\}\}$.

The set of states and the initial state in the dual quotient automaton are defined as in a quotient automaton. A block in π is final iff every state in the block is final. If a block contains final and non-final states, the corresponding state is not final. Thus, there may be a word w such that the run of A on w leads to a final state but the corresponding run of $A \backslash \pi$ does not. A transition between two blocks $[q]$ and $[q']$ is defined iff all transitions from states in $[q]$ end either in a state in $[q']$ or in $q_\top$. Though the introduction of the state $q_\top$ in Definition 26 may seem unexpected, there is an intuitive explanation. Recall the definition of the transition function of a quotient automaton. There exists a transition between two blocks $[q_1]$ and $[q_2]$ if there exists a transition from *any* state in $[q_1]$ to *any* state in $[q_2]$. The effect is that $\bigcup_{q\in[q_1]} \mathit{Suff}(q) \subseteq \mathit{Suff}([q_1])$. In the dual-quotient automaton, as all transitions from states in a block $[q_1]$ must either end in states in the same block or the state $q_\top$, we have that $\mathit{Suff}([q_1]) \subseteq \bigcup_{q\in[q_1]} \mathit{Suff}(q)$. Here, we only prove that the two definitions of a dual quotient are equivalent for deterministic, complete automata. We show that $w \notin L(A\backslash\pi)$ iff $w \in L(\overline{A}/\pi)$.

Lemma 23. *Let π be a partition of the states of a complete deterministic automaton $A = (Q, \Sigma, \delta, q_0, F)$. It holds that $L(A\backslash\pi) = \overline{L(\overline{A}/\pi)}$*

Proof. By definition of the complement of an automaton, we know that $L(A) = \overline{L(\overline{A})}$, so it is sufficient to examine $L(A) \setminus L(A\backslash\pi)$. It suffices to prove that for any $w \in \Sigma^*$, $w \in L(A) \setminus L(A\backslash\pi) \Leftrightarrow w \in L(\overline{A}/\pi) \setminus L(\overline{A})$. That is, any word accepted by $L(A)$ that is removed from the language of the dual-quotient automaton is a word that is not accepted by $\overline{A}$ but is added to the language of the dual-quotient automaton.

The partition π is of index at most $|Q|$. We prove the lemma by induction on the index of the partition. For the base case, let the index of π be $|Q|$. As $A\backslash\pi = A$ and $\overline{A}/\pi = \overline{A}$, we are done. For the induction hypothesis, assume that for π of index $k + 1 \leq |Q|$, $A\backslash\pi$ is deterministic and that $L(A\backslash\pi) = \overline{L(\overline{A}/\pi)}$.

We show that the same holds for a partition of index k. It is sufficient to consider a partition in which only two states are merged. Let q, q' be the states that are merged in A and $\overline{q}, \overline{q}'$ be the corresponding states in $\overline{A}$. Consider a word $w = uxv \in L(\overline{A}/\pi) \setminus L(\overline{A})$ such that $u \in \mathit{Pre}([\overline{q}]) \setminus L([\overline{q}], [\overline{q}])$ and $v \in \mathit{Suff}([\overline{q}]) \setminus L([\overline{q}], [\overline{q}])$. That is, the run of A on u visits q or q' for the first time and the run of A on v that begins in q or q' does not revisit these states. We show that $w \in L(A)\backslash L(A\backslash\pi)$.

Without loss of generality, let $\delta(q_0, u) = q$. We consider three cases. (1) $xv\lambda$, (2) $x = \lambda$ and $v \neq \lambda$ and (3) $x, v \neq \lambda$.
(1) If $xv = \lambda$, then, by assumption, the run of $\overline{A}$ ends in $\overline{q}$, which must be non-final as $u \notin L(\overline{A})$. As u is in $L\left(\overline{A}/\pi\right)$, the state $\overline{q}'$ must be final. In the automaton A, q must be final and q', non-final, so $[q]$ is not final and the run of $A\backslash\pi$ on u ends in a non-final state. Thus $w \in L(A)\backslash L(A\backslash\pi)$.
(2) If $x = \lambda$ and $v \neq \lambda$, as $\delta(\overline{q}_0, u) = \overline{q}$ by assumption, it must be that $v \in \mathit{Suff}(\overline{q}') \setminus \mathit{Suff}(\overline{q})$. Let $v = av'$ for some symbol a. If $\delta(\overline{q}, a) = q_\bot$, then $\delta(q, a) = q_\top$, so by the definition of the transition of the dual-quotient automaton, we have that $\underline{\delta_\pi([q], a) =} [\delta(q, a)]$. By the definition of the complement, we have that $\overline{\mathit{Suff}(\delta(q, a))} = \mathit{Suff}(\delta(\overline{q}, a))$, so if $v \in \mathit{Suff}(\overline{q}')$, then $v \notin \mathit{Suff}(q')$, hence $w \in L(A)\backslash L(A\backslash\pi)$. If $\delta(\overline{q}, a) = \overline{q_1}$ such that $\overline{q_1} \neq q_\bot$, it must be that $\overline{q_1} \neq \overline{q}'$. By the definition of the dual quotient, $\delta([q], a) = q_\bot$ as $\delta(q, a) \neq \delta(q', a)$, so $v \notin \mathit{Suff}([q])$ and $w \in L(A)\backslash L(A\backslash\pi)$.

(3) If $x \neq \lambda$, let $x = y_1 \cdots y_n$, where $y_i \in L(\{\overline{q}, \overline{q}'\})$. It is sufficient consider the case where $uv \in L(\overline{A})$ and $uxv \notin L(\overline{A})$. Note that for each y_i, either $\delta(\overline{q}, y_i) \in \{\overline{q}, \overline{q}'\}$ or $\delta(\overline{q}', y_i) \in \{\overline{q}, \overline{q}'\}$. Say $\delta(\overline{q}, y_i) \in \{\overline{q}, \overline{q}'\}$ and $y_i = ay_i'$ and $\delta(\overline{q}, a) = \overline{q}_1$. If $\delta(q', a) \notin \{q_1, q_\top\}$, then $\delta([q], a) = q_\bot$, which implies that for any $x' \in a \cdot \Sigma^*$, $x' \notin L(A \backslash \pi)$, so we have that $xv \notin \mathit{Suff}([q])$ and $uxv \in L(A) \setminus L(A \backslash \pi)$ as required. If $\delta([q], a) = q_1$, then $\delta([q], a) = [q_1]$ and $\delta([q], y_i) = [q]$. It remains to show that $v \notin \mathit{Suff}([q]) \cup \mathit{Suff}([q'])$. There are only two possibilities. Either v is accepted by both states $\overline{q}$ and $\overline{q}'$, or v is accepted by exactly one of these states. If $v \in \mathit{Suff}(\overline{q}) \cap \mathit{Suff}(\overline{q}')$, then $v \notin \mathit{Suff}(q) \cup \mathit{Suff}(q)$ by the definition of the complement and we are done. If v is accepted by exactly one of $\overline{q}$ and $\overline{q}'$, by a similar argument to the case where $x = \lambda$ in the word $w = uxv$, we can show that $v \notin \mathit{Suff}([q])$. Thus, if $x \neq \lambda$, $uxv \in L(A) \setminus L(A \backslash \pi)$.

We have shown that for any word $w = uxv$, if $uxv \in L(\overline{A}/\pi) \setminus L(\overline{A})$, then $uxv \in L(A) \setminus L(A \backslash \pi)$. The other direction is proved via a similar sequence of arguments. □

Having shown that the dual-quotient operation we have defined is the dual of the quotient operation, we may literally harvest the benefits of the duality principle. To appreciate the significance of this, we revisit a few results from the previous sections. We begin with the characterisation of the search space for dual-widening. Recall that the set of quotients of an automaton form a lattice. Similarly, the set of dual quotients also form a lattice. Define a negative sample of a language L as a set of words $S \subseteq \overline{L}$. The dual of the notion of structural completeness applies for negative samples.

Definition 27 (Dual-Structural Completeness). A negative sample $S \subseteq \Sigma^*$ is structurally complete with respect to a complete, deterministic automaton $A = (Q, \Sigma, \delta, q_0, F)$ iff

1. $S \subseteq L(\overline{A})$
2. For each $q \notin F$, there exists $w \in S$ such that $\delta(q_0, w) = q$.
3. For each $q, q' \in Q$ and $a \in \Sigma$ such that $q' \neq q_\top$ and $\delta(q, a) = q'$, there exists $w = uav \in S$ such that $\delta(q_0, u) = q$.

That is, a negative sample is structurally complete with respect to a complete automaton if it contains words corresponding to a run ending in each non-final state and if every transition not leading to the universal state is exercised. We now have the dual of the sufficient condition in Theorem 4.

Theorem 10. *Let the automata $A = (Q, \Sigma, \delta, r_0, F)$ and $A_\infty = (Q_\infty, \Sigma, \delta_\infty, t_0, F_\infty)$ be complete and deterministic. If $L(\overline{A})$ is a dual-structurally complete negative sample with respect to A_∞ and if for all $w \in L(A)$, it holds that $\mathit{Pre}(\delta(r_0, w)) \supseteq \mathit{Pre}(\delta_\infty(t_0, w))$, then A_∞ belongs to $\mathit{Lattice}(A)$.*

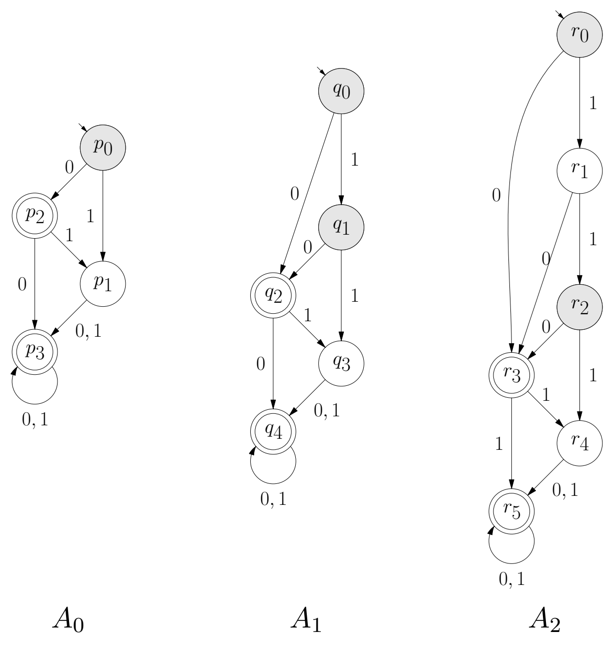


Figure 5.1: Fixpoint computation in Example 28

Proof. Follows from Theorem 4 and the duality principle. □

What is the intuition behind this theorem? The idea behind the characterisation in Theorem 4 was that if a state in an automaton A has more prefixes than any state in A_∞, a quotient of A may accept words that are not accepted by A_∞. Similarly, as the dual-quotient reduces the language accepted by an automaton, a fixpoint can be obtained by merging states in A only if each state in A has more prefixes than the corresponding state in A_∞.

We refrain from reproducing the contents of this thesis as a sequence of dual statements and only present a dual-widening operator for greatest-fixpoint computations. Lesens et al. [2001] introduce a widening operator for automata in a greatest fixpoint computation. Their widening technique, though presented as an informal heuristic, uses the dual widening seed $\{=_p, =_s\}$.

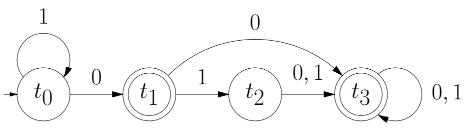
Example 28. The three minimal automata in Figure 4.1 occur in a greatest fixpoint computation. The complement of each of these automata appears in the least fixpoint computation in Example 22. Observe that $L(A_0) \supseteq L(A_1) \supseteq L(A_2)$.

Let A_0 be the widening parameter and A_1 the widening candidate. The initial states have the same prefixes and are related. The state q_1 accepts the same language as p_0. In fact, the part of the automaton reachable from q_1

is isomorphic to A_0. Hence, p_0 is related to two states in A_1. The relations required for computing the widening equivalence are shown below.

$$\begin{aligned}
=_p &= \{\langle p_0, q_0\rangle\}\\
=_s &= \{\langle p_0, q_1\rangle, \langle p_2, q_2\rangle, \langle p_1, q_3\rangle, \langle p_3, q_4\rangle\}\\
\mathcal{R}_\nabla &= id \cup \{\langle q_0, q_1\rangle, \langle q_1, q_0\rangle\}\\
\left[\equiv_{\mathcal{S}}^{A_0}\right] &= \{\{q_0, q_1\}, \{q_2\}, \{q_3\}, \{q_3\}\}
\end{aligned}$$

The relations $=_p$ and $=_s$ are identical to the relations computed in Example 27, in which we used the complement of the automata considered here. The shaded states in A_1 are in the same partition and are merged in the dual-quotient construction. The widened automaton shown below is the precise fixpoint of this computation.



A_∇

The widened automaton A_∇ is the complement of the widened automaton obtained in Example 27.

The purpose of this example was to demonstrate that greatest fixpoint computations with dual widening are the dual of least fixpoint computations with widening. We now focus on using widening and dual widening for model checking.

5.2 Model Checking

The problem we consider in this section is automatically determining if a system satisfies a formally specified property. A system *Sys* is described by a set of initial states S_0 from a domain D and a transition relation $\mathcal{T} \subseteq D \times D$. (We have previously considered $\mathcal{T}$ as a function from $\wp(D)$ to $\wp(D)$ but the formulation as a relation is notationally convenient for the discussion that follows.) Let $S \subseteq D$ be a set of states. The successors of states in S are denoted $post[\mathcal{T}](S)$. The predecessors of states in S are denoted $pre[\mathcal{T}](S)$. The states with *all* their predecessors in S is denoted $\widetilde{post}[\mathcal{T}](S)$. The states with *all* their successors in S is denoted $\widetilde{pre}[\mathcal{T}](S)$. These notions are formally defined below.

Definition 28. Let $\mathcal{T} \subseteq D \times D$ be a transition relation and $S \subseteq D$ be a set of states.

- $post[\mathcal{T}](S) = \{s' \in D | \exists s \in S : \langle s, s'\rangle \in \mathcal{T}\}$

- $\widetilde{post}[\mathcal{T}](S) = \{s' \in D | \forall s \in D : \langle s, s' \rangle \in \mathcal{T} \Rightarrow s \in S\}$
- $pre[\mathcal{T}](S) = \{s \in D | \exists s' \in S : \langle s, s' \rangle \in \mathcal{T}\}$
- $\widetilde{pre}[\mathcal{T}](S) = \{s \in D | \forall s' \in D : \langle s, s' \rangle \in \mathcal{T} \Rightarrow s' \in S\}$

If the transition relation used is clear, we do not write it. A state s is *reachable* iff $s \in post[\mathcal{T}^*](S_0)$. A *run* of *Sys* is a possibly infinite sequence of states $s_0, s_1, \ldots$ such that $s_0 \in S_0$ and for all s_i, s_{i+1} in the sequence, $\langle s_i, s_{i+1} \rangle \in \mathcal{T}$.

An *invariance property* or *invariant*, $I \subseteq D$, is a set of states. A system *Sys* with initial states S_0 satisfies an invariant I, denoted $Sys \models I$ iff $post[\mathcal{T}^*](S_0) \subseteq I$. A *counterexample* for I is a finite run of *Sys*, $s_0, s_1, \ldots s_k$ such that $s_0 \in S_0$ and $s_k \notin I$. There are different ways to check if a system satisfies an invariant.

- Forward Analysis: Involves computing the set of reachable states. Let F be the function $F(X) = post(X) \cup S_0$, with $F^0 = S_0$ and $F^{i+1} = post(F^i(F^0))$. The set of reachable states is the least fixpoint of F. The system satisfies the property if $lfp(F) \subseteq I$. If $Sys \not\models I$, there exists k such that $F^k \not\subseteq I$. A counterexample is constructed by finding a sequence $s_0, \ldots, s_k$ such that for $0 \leq i < k$, $s_i \in F^i$ and $s_k \in F^k \setminus I$. That is, a sequence of valid transitions leading to a state that does not satisfy the invariant.

- Backward Analysis: Involves computing the states from which a state violating the invariant is reachable. Let $\overline{I}$ denote the set $D \setminus I$ of states violating the invariant. The system satisfies the property if the set of states from which $\overline{I}$ is reachable does not include a state in S_0. Let B be the function $B(X) = pre(X) \cup \overline{I}$, with $B^0 = \overline{I}$ and $B^{i+1} = pre(B^i(B^0))$. The set of states from which $\overline{I}$ is reachable is the least fixpoint of B. The system satisfies the property if $S_0 \cap lfp(B) = \emptyset$. If $Sys \not\models I$, there exists i such that $S_0 \cap B^i \neq \emptyset$. A counterexample is constructed by finding a sequence $s_0, \ldots, s_k$ such that $s_0 \in B^k \cap S_0$ and for $0 \leq i < k$, $s_{k-i} \in B^i$.

- Greatest Fixpoint Analysis: Involves computing the is the largest set of states that never violates the invariant. This set is also called the *greatest inductive invariant.* Let G be the function $G(X) = \widetilde{pre}(X) \cap I$ with $G^0 = I$ and $G^{i+1} = G^i(G^0) \cap \widetilde{pre}(G^i(G^0))$. The set of states that always satisfy the invariant is $gfp(G)$. The system satisfies the property if $S_0 \subseteq gfp(G)$. If $Sys \not\models I$, this conclusion can only be reached after $gfp(G)$ has been computed because an initial state may not be in G^i but may be in G^{i+1}. If $S_0 \not\subseteq gfp(G)$, there exists a state s in $S_0 \setminus gfp(G)$ that may lead to a state in $\overline{I}$, but a counterexample cannot be directly constructed from the greatest fixpoint computation.

The preference for a given method depends on the system and the property that must be verified. Forward and backward analyses can be used to generate counterexamples if verification fails. Backward and greatest fixpoint analyses take the property into account, so the size of the sets in the fixpoint computations may be smaller. Lesens et al. [2001] exhibit systems for which the set of reachable states only has a context-free representation but the greatest inductive invariant has a regular representation. In such cases, the greatest fixpoint analysis is preferrable. The systems we consider have infinitely many states, so, in general none of the fixpoint computations above may terminate.

Wolper and Boigelot [1998] identify two approaches that have been adopted to tackle the termination problem. The first is to study classes of infinite state systems for which the fixpoint computation does terminate. Timed automata [Alur and Dill 1994] are one such class. The second approach is to consider larger classes of systems and provide an algorithm that may not terminate for all inputs. A third approach we see is that adopted in abstract interpretation and more recently in abstraction based techniques: Provide an algorithm that always terminates but may return a false negative if the system satisfies the property.

The widening techniques can be used with the second and third approaches mentioned above. A widening seed such as $\{\cap_s\}$, can be used to precisely compute a class of fixpoints, but may not enforce termination of all computations. In contrast, if the k-tails widening seed is used, the fixpoint computation is guaranteed to terminate but may return a false negative, particularly if the fixpoint is not regular or is regular but is represented by an automaton with more than k states.

Our widening framework is general, in that it can be combined with the forward, backward and greatest fixpoint analyses. Forward and backward analyses, being essentially least fixpoint computations, can be combined with widening and greatest fixpoint analysis can be combined with dual widening. We discuss the use of widening with forward and backward analysis.

In Algorithm 1, we described a least fixpoint computation with widening. Forward analysis is an extension of this algorithm that at each step includes a check for whether the invariant is satisfied. Let $S_0, \dots$ be the sequence of sets in a forward analysis with widening and let I be the invariant. If for some $k \in \mathbb{N}$, $S_k \cap \overline{I} \neq \emptyset$, we have found a state, say s, that violates the invariant. As a widening step may have been applied, we do not know if s is a reachable state. We tackle this problem in steps.

First, we do a *bounded backward analysis* for k steps, starting from the set $S_k \cap \overline{I}$. If the state s is reachable in k steps, an initial state is reached after at most k steps of the backward analysis. In this case, we can generate a counterexample and the analysis terminates. Let E be the set of states computed by the bounded backward analysis. If E does not contain initial

```
Forward Analysis With Widening($\mathcal{T}, S_0, I$)
Input: Transition relation $\mathcal{T}$, Initial States $S_0$,
       Invariant $I$
begin
    Define $F(X)$ as $(post(X) \cup S_0)$
    $i \leftarrow 1$
    repeat
        $S_i \leftarrow F(S_{i-1})$
        $S_i \leftarrow$ Widen($S_i$)
        if $\big(S_i \cap \overline{I}\big) \neq \emptyset$ then
            Define $B(X)$ as $\big(pre(X) \cup \big(S_i \cap \overline{I}\big)\big)$
            if $B^i \cap S_0 \neq \emptyset$ then
                Generate counterexample and stop
            else
                $S_i \leftarrow S_i \setminus B^i$
            end
        end
        $i \leftarrow i + 1$
    until $S_i \subseteq S_{i-1}$
end
```

Algorithm 3: Forward analysis with widening and counterexample generation

states, we continue the forward analysis with the set $S_k \setminus E$. This heuristic is sound. As E does not contain any initial states, $S_0 \subseteq S_k$. If some $s' \in E$ is reachable, there exists some n such that $s' \in F^n(S_0)$. Hence, there exists $m \leq n$ such that $s' \in F^m(S_k \setminus E)$ and the soundness of the analysis is preserved. Forward analysis the incorporating bounded backward analysis described above is shown in Algorithm 3.

Checking if a counterexample exists, is in general undecidable and in fact as hard as verifying an infinite state system. To see this, consider an invariant I with $\overline{I} = \{s\}$. Let the widening operation in the k^{th} step of a fixpoint computation extrapolate the set of states to the precise fixpoint and let this set S_k contain a state violating the invariant. Checking if a counterexample exists reduces to the backward analysis problem, which we know may not terminate. Generating counterexamples in a backward analysis with widening is similar, so we do not discuss the issue here.

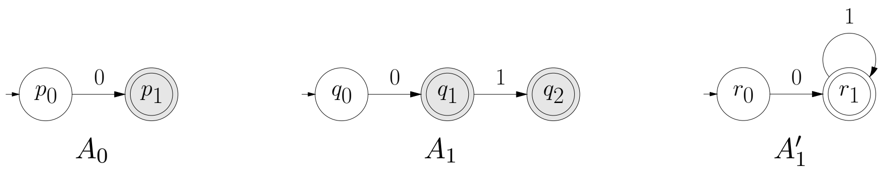
The heuristic we suggest is just one of many possible ways to search for a counterexample after a property violation is detected in an infinite state system. There is much work on model checking infinite state systems using abstraction, where a related problem arises. An abstract counterexample is a sequence of states in an abstract domain. Each abstract state corresponds to a possibly infinite set of concrete states. Let $a_1, \ldots, a_n$ be a sequence of abstract states and $C_1, \ldots, C_n$ be the corresponding sequence of sets of concrete states. How can counterexamples be generated from these sets?

Pace et al. [2004] obtain a set of constraints that the counterexample must satisfy and use a testing tool to generate sequences of inputs satisfying those constraints. If a test run leads to a state in $\overline{I}$, a concrete counterexample is found. [Erez et al. 2004] use bounded model checking (an efficient technique for exploring the set of states reachable within a bounded number of steps) to search for counterexamples, starting from states in C_0. [Clarke et al. 2003] attempt to find a transition from a state $s'_{i-1} \in C_{i-1}$ to a state $s_i \in C_i$, and a run from the state s_i to s'_i for each $1 \leq i \leq n$, where n is the number of states in the abstract counterexample. Only a few heuristics exist for finding counterexamples and the search for better heuristics is an area of active research.

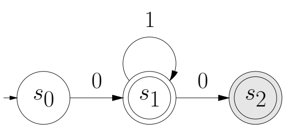
5.3 Selecting the Widening Parameter

The last topic we discuss in this section is the selection of the widening parameter. One possible choice is to always choose A_i as the widening parameter and A_{i+1} as the widening candidate, where A_i and A_{i+1} are automata in the fixpoint computation. We call such a choice the *naïve strategy*. We show in Example 29 that the naïve strategy may not always be the best choice.

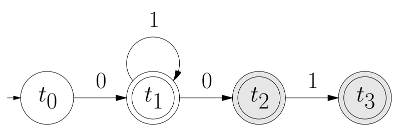
Example 29. Let $\mathcal{T}$ be a transition function such that for a set of words S over the alphabet $\{0, 10\}$, $\mathcal{T}(S) = S \cup \{w \cdot 0 | w \in S \wedge w = u \cdot 1 \text{ for some } u \in \Sigma^*\} \cup \{w \cdot 1 | w \in S \wedge w = u \cdot 0 \text{ for some } u \in \Sigma^*\}$. Let the initial set S_0 be $\{0\}$. We demonstrate the effect of using the naïve widening strategy with the Bartzis-Bultan seed in this computation.



A_0 and A_1 are the first two automata in the sequence. The states p_1 and q_1 have the prefix 0 and p_1 and q_2 have the suffix set $\{\lambda\}$. Hence, q_1 and q_2 are related and are merged to obtain the widened automaton A'_1. The transition function is applied to A'_1 to obtain A_2 below. No two states in A'_1 have the same prefix as a state in A_2 and no state in A_2 has the same suffixes as a state in A'_1, so every state in A_2 is only related to itself by the widening equivalence. The automaton A_3 appears next in the computation. The states s_2 and t_2 have the same prefixes and s_2 and t_3 have the same suffixes, so t_2 and t_3 are related in A_3.

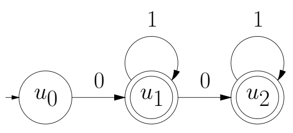


A_3

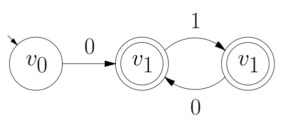


A_4

The widened automaton A'_4 is shown below. The pattern of the computation with widening should be clear. If the naïve widening strategy is used, the computation does not terminate and converges in the limit to the language $0 \cdot \Sigma^*$. If the computation is allowed to progress in the initial steps without applying widening, choosing A_3 as the widening candidate and A_0 as the widening parameter results in the widened automaton A_∞ below, which is also the fixpoint of the computation.



A'_4



A_∞

The previous example demonstrates that given the *same* widening operator and fixpoint computation, the widening parameter and candidate that are chosen affect the precision and termination properties of the fixpoint computation with widening. The naïve strategy may not always be the best.

As the effect of a widening differs with each widening seed, the widening parameter and candidate that are chosen depend on the widening seed. We do not undertake such a study here, but survey existing heuristics and adapt them to our framework.

Boigelot et al. [2003] consider automata over the alphabet $\{0, 1\}$, accepting words that are a binary encoding of the natural numbers. They observe that the naïve widening strategy does not introduce any extrapolation, but choosing the automata A_{2^i} and $A_{2^{i+1}}$ does. Using a sub-sequence is a general strategy that may work for regular fixpoints, but choosing the sequence A_{2^i} is not necessary, even for automata accepting binary encodings of natural numbers. We illustrate this in Example 30

Example 30. Let a word $w \in \{0, 1\}^*$ be a most significant bit, binary encoding of a natural number. Any word in the language 0^* represents 0, a word in 0^*1 represents 1 and so forth. Consider the transition relation in Example 29. The first four sets of words in the computation are, $\{0\}, \{0, 01\}, \{0, 01, 010\}$ and $\{0, 01, 010, 0101\}$. The corresponding sequence of numbers is $\{0\}, \{0, 1\}, \{0, 1, 2\}$ and $\{0, 1, 2, 5\}$. The fixpoint of this sequence can be considered as possible set of values of the variable x in the following program:

```
**begin**
    $x \leftarrow 0$
    **repeat**
        **if** $x$ *is odd* **then**
            $x \leftarrow 2x$
        **else**
            $x \leftarrow 2x + 1$
        **end**
    **until** *Eternity*
**end**
```

Thus, we have a sequence of automata corresponding to the kind of system considered in [Boigelot et al. 2003]. If the BLW seed is used with a candidate A_i, for $i \geq 2$, any widening parameter A_j such that $j < i$ and $(i - j)$ is a multiple of 2 suffices to precisely compute the fixpoint.

Boigelot et al. [2003] suggest an additional heuristic to be used after choosing the widening parameter and candidate but before applying widening. We discuss the heuristic within our framework. Let $Ind = \{s_0, \ldots, s_k\}$ be a set of indices and the automata A_i for $i \in Ind$ be a sub-sequence of a computation without widening. Let A'_i for $i \in Ind \backslash \{s_0\}$ denote the widened automaton computed using two automata from the subsequence as the widening candidate and parameter. If $L(A'_i) = L(A'_j)$ for all $i, j \in Ind$, the computation continues from A'_{s_k}. If $L(A'_i) \neq L(A'_j)$ for some i, j, then the computation proceeds without the widening step.

This heuristic is useful for checking that an extrapolation introduced by widening does correspond to a repeated pattern and helps avoid bad choices as in Example 30. If the fixpoint is not regular, it there may be no choice that satisfies the conditions of the heuristic and hence, widening may never be applied. In such a situation, if a choice between a set of possible widening parameters and candidates is to be made, we suggest the choice that results in a widening equivalence of smallest index. Simply put, we choose the pair that results in the greatest extrapolation.

Selecting the widening parameter is a heuristic. A formal study can establish the effect of a specific choice on the computation, but the true test of a heuristic is performance in practical applications. In addition, the heuristic used may vary with the problem domain and the sets that are encoded as automata. Determining the utility of a heuristic requires experimental evidence, which we neither have nor provide in this thesis.

Chapter 6

Conclusion

The original contributions of this thesis are several. We have developed a framework for designing widening operators and for using these operators to over-approximate fixpoint computations with regular and weak ω-regular sets. To the best of our knowledge, no such framework exists in the literature. We have established a connection between widening fixpoint computations and inductive inference. Three widening operators and two inductive inference algorithms in the literature are instances of our framework. In addition, each widening operator defined within our framework gives rise to a new inductive inference algorithm. The highlights of this thesis are: A sufficient condition for computing a regular fixpoint using widening, which generalises the results of [Dupont et al. 1994], and an analogous condition for computing weak ω-regular fixpoints. Criteria for any widening operator to be extrapolating and for any computation with widening to converge to the precise fixpoint. A study of extrapolation, termination and precision properties of eight different widening operators. In our opinion, this thesis is just a starting point in a fruitful and challenging research direction. We now discuss some open problems and conclude.

Several authors have commented that defining a widening operator that guarantees termination of all fixpoint computations without introducing too much imprecision is difficult [Lesens et al. 2001; Bartzis and Bultan 2004]. In fact, the widening operator obtained from the k-tails widening seed is the only one we are aware of, which does guarantee termination and can be used to compute a non-trivial over-approximation. It would be both interesting and useful to discover if other such operators exist.

A second open problem is the design of widening operators for automata on infinite words. We are only aware of one paper on this topic [Boigelot et al. 2004]. As many existing tools use ω-automata as a representation, the practical implications of useful widening operators for ω-automata is significant. Such widening operators can be defined within our framework. The challenge is to find the appropriate widening seed and analyse its properties.

The genesis of this thesis was the search for acceleration techniques for model checking infinite state systems. In Section 5.2 we indicated how our widening techniques can be combined with model checking algorithms and suggested heuristics for generating counterexamples and for selecting the widening parameter. A strong statement about the practical utility of the widening operators we have studied can only be made after they are integrated in an automata-based model checker.

On the same day that we began this thesis, Halbwachs [Halbwachs 2006] commented, in his tutorial on widening for polyhedra, that widening is treated with disdain, and began with the manifesto: "*This tutorial aims at correcting this opinion, by showing that the design of widening operators can follow some principles.*" Our efforts in this thesis show that such design principles can be both rigorous and conceptually simple and simultaneously lead to rich and exciting research problems.

BIBLIOGRAPHY

ALUR, R. AND DILL, D. L. 1994. A theory of timed automata. *Theoretical Computer Science 126,* 2, 183–235.

AMMONS, G., BODÍK, R., AND R.LARUS, J. 2002. Mining specifications. In *POPL '02: Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of Programming Languages*. ACM Press, New York, NY, USA, 4–16.

ANGLUIN, D. 1978. On the complexity of minimum inference of regular sets. *Information and Control 39*, 337–350.

ANGLUIN, D. 1980a. Finding patterns common to a set of strings. *Journal of Computer and System Sciences 21,* 1 (Aug.), 46–62.

ANGLUIN, D. 1980b. Inductive inference of formal languages from positive data. *Information and Control 45*, 117–135.

ANGLUIN, D. 1982. Inference of reversible languages. *Journal of the Association for Computing Machinery 29,* 3, 741–765.

ANGLUIN, D. 1987a. Learning regular sets from queries and counterexamples. *Information and Computation 75,* 2, 87–106.

ANGLUIN, D. 1987b. Queries and concept learning. *Machine Learning 2*, 319–342.

ANGLUIN, D. AND SMITH, C. 1983. Inductive inference: Theory and methods. *Computing Surveys 15,* 3, 237–269.

BAETEN, J. C. M. AND WEIJLAND, W. P. 1990. *Process algebra*. Cambridge University Press, New York, NY, USA.

BARDIN, S., FINKEL, A., LEROUX, J., AND PETRUCCI, L. 2003. Fast: Fast acceleration of symbolikc transition systems. In *Computer Aided Verification, 15th International Conference*. Lecture Notes in Computer Science, vol. 2725. Springer, 118–121.

BARTZIS, C. AND BULTAN, T. 2004. Widening arithmetic automata. In *CAV*. 321–333.

BIERMAN, A. W. AND FELDMAN, J. A. 1972. On the synthesis of finite-state machines from samples of their behavior. *IEEE Transactions on Computers C-21,* 6 (June), 592–597.

BLUMENSATH, A. AND GRÄDEL, E. 2000. Automatic Structures. In *Proceedings of 15th IEEE Symposium on Logic in Computer Science LICS 2000*. 51–62.

BOIGELOT, B., JODOGNE, S., AND WOLPER, P. 2005. An effective decision procedure for linear arithmetic over the integers and reals. *ACM Trans. on Computational Logic 6,* 3, 614–633.

BOIGELOT, B., LEGAY, A., AND WOLPER, P. 2003. Iterating transducers in the large (extended abstract). In *Computer Aided Verification*. 223–235.

BOIGELOT, B., LEGAY, A., AND WOLPER, P. 2004. Omega-regular model checking. In *Tools and Algorithms for the Construction and Analysis of Systems, 10th International Conference, TACAS 2004*. Lecture Notes in Computer Science, vol. 2988. Springer, 561–575.

BOUAJJANI, A., JONSSON, B., NILSSON, M., AND TOUILI, T. 2000. Regular model checking. In *CAV '00: Proceedings of the 12th International Conference on Computer Aided Verification*. Springer-Verlag, London, UK, 403–418.

Bruyère, V., Hansel, G., Michaux, C., and Villemaire, R. 1994. Logic and p-recognizable sets of integers. *Bulletein of the Belgian Mathematical Society 1*, 191–238. Corrigendum, *Bull. Belg. Math. Soc.* **1** (1994), 577.

Büchi, J. R. 1960. Weak secord-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik 6*, 66–92. Reprinted in S. Mac Lane and D. Siefkes, eds., *The Collected Works of J. Richard Büchi*, Springer-Verlag, 1990, pp. 398–424.

Büchi, J. R. 1962. On a decision method in restricted second order arithmetic. In *Proceedings of the International Conference on Logic, Methodology and Philosophy of Science*. Stanford University Press, Stanford, California, 1–11.

Bustan, D. and Grumberg, O. 2004. Applicability of fair simulation. *Inf. Comput. 194,* 1, 1–18.

Champarnaud, J.-M. and Coulon, F. 2004. Nfa reduction algorithms by means of regular inequalities. *Theoretical Computer Science 327,* 3, 241–253.

Champarnaud, J.-M. and Coulon, F. 2005. Erratum to "nfa reduction algorithms by means of regular inequalities" [tcs 327 (2004) 241-253]. *Theoretical Computer Science 347,* 1-2, 437–440.

Choi, B. 2002. Applying learning by examples for digital design automation. *Applied Intelligence 16,* 3, 205–221.

Clarke, E. M., Fehnker, A., Han, Z., Krogh, B. H., Ouaknine, J., Stursberg, O., and Theobald, M. 2003. Abstraction and counterexample-guided refinement in model checking of hybrid systems. *International Journal of Foundations of Computer Science 14,* 4, 583–604.

Cobham, A. 1969. On the base-dependence of sets of numbers recognizable by finite automata. *Mathematical Systems Theory 3,* 2, 186–192.

Cousot, P. 1978. Méthodes itératives de construction et d'approximation de point fixes d'oprateurs monotone sur un treilis, analyse sémantique des programmes. Ph.D. thesis, University of Grenoble, France.

Cousot, P. 2005. Abstract interpretation. MIT course 16.399.

Cousot, P. and Cousot, R. 1977. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages.* ACM Press, New York, NY, Los Angeles, California, 238–252.

Cousot, P. and Cousot, R. 1992a. Abstract interpretation and application to logic programs. *Journal of Logic Programming 13,* 2–3, 103–179.

Cousot, P. and Cousot, R. 1992b. Abstract interpretation frameworks. *Journal of Logic and Computation 2,* 4 (Aug.), 511–547.

Cousot, P. and Cousot, R. 1992c. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In *Proceedings of the International Workshop Programming Language Implementation and Logic Programming, PLILP*, M. Bruynooghe and M. Wirsing, Eds. Leuven, Belgium, 13–17 August 1992, Lecture Notes in Computer Science 631. Springer-Verlag, Berlin, Germany, 269–295.

Dams, D., Lakhnech, Y., and Steffen, M. 2001. Iterating transducers. In *CAV '01: Proceedings of the 13th International Conference on Computer Aided Verification.* Springer-Verlag, London, UK, 286–297.

Davey, B. A. and Priestley, H. A. 1990. *Introduction to Lattices and Order.* Cambridge University Press.

Dembiński, P., Penczek, W., and Pólrola, A. 2002. Verification of timed automata based on similarity. *Fundamenta Informatica 51,* 1, 59–89.

DILL, D. L., HU, A. J., AND WONG-TOI, H. 1992. Checking for language inclusion using simulation preorders. In *CAV '91: Proceedings of the 3rd International Workshop on Computer Aided Verification*. Springer-Verlag, London, UK, 255–265.

DUPONT, P., MICLET, L., AND VIDAL, E. 1994. What is the search space of the regular inference? In *Grammatical Inference and Applications, ICGI'94*. Number 862 in Lecture Notes in Artificial Intelligence. Springer Verlag, 25–37.

EREZ, G., YAHAV, E., AND SAGIV, M. 2004. Generating concrete counterexamples for sound abstract interpretation. Tech. rep., Tel-Aviv University, Tel-Aviv, Israel.

ETESSAMI, K., WILKE, T., AND SCHULLER, R. A. 2005. Fair simulation relations, parity games, and state space reduction for bu"chi automata. *SIAM Journal of Computing 34,* 5, 1159–1175.

GOLD, E. 1978. Complexity of automaton identification from given data. *Information and Control 37*, 302–320.

GOLD, E. M. 1967. Language identification in the limit. *Information and Control 10,* 5, 447–474.

GURUMURTHY, S., BLOEM, R., AND SOMENZI, F. 2002. Fair simulation minimization. In *CAV '02: Proceedings of the 14th International Conference on Computer Aided Verification*. Springer-Verlag, London, UK, 610–624.

HALBWACHS, N. 2006. On the design of widening operators. In *VMCAI 2006: 7th International Conference on Verification, Model Checking, and Abstract Interpretation, Charleston, SC, USA*. Lecture Notes in Computer Science, vol. 3855. Springer Verlag, Berlin, Germany.

HENZINGER, T. A., KUPFERMAN, O., AND RAJAMANI, S. K. 2002. Fair simulation. *Information and Computation 173,* 1, 64–81.

HENZINGER, T. A., MAJUMDAR, R., AND RASKIN, J.-F. 2005. A classification of symbolic transition systems. *ACM Transactions on Computational Logic 6,* 1, 1–32.

HIGUERA, C. D. L. AND JANODET, J.-C. 2001. Inference of ω-languages from prefixes. In *ALT '01: Proceedings of the 12th International Conference on Algorithmic Learning Theory*. Springer-Verlag, London, UK, 364–378.

HOLZMANN, G. J. 2004. *The SPIN model checker: Primer and reference manual*. Addison Wesley, Redwood City, CA, USA.

HOPCROFT, J. AND ULLMAN, J. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts.

HOPCROFT, J. E. 1971. An $n \log(n)$ algorithm for minimizing states in a finite automaton. Tech. rep., Stanford University, Stanford, CA, USA.

ILIE, L., SOLIS-OBA, R., AND YU, S. 2005. Reducing the size of NFAs by using equivalences and preorders. In *Combinatorial Pattern Matching, 16th Annual Symposium, CPM 2005, Jeju Island, Korea, June 19-22, 2005, Proceedings*. Springer, 310–321.

ILIE, L. AND YU, S. 2002. Algorithms for computing small nfas. In *MFCS '02: Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science*. Springer-Verlag, London, UK, 328–340.

KHOUSSAINOV, B. AND NERODE, A. 1995. Automatic presentations of structures. In *LCC '94: Selected Papers from the International Workshop on Logical and Computational Complexity*. Springer-Verlag, London, UK, 367–392.

KLARLUND, N., MØLLER, A., AND SCHWARTZBACH, M. I. 2002. Mona implementation secrets. *International Journal of Foundations of Computer Science 13,* 4, 571–586.

KNUTH, D. E. 1998. *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

KNUTH, D. E. 2005. *The Art of Computer Programming, Volume 4, Fascicle 3: Generating All Combinations and Partitions*. Addison-Wesley Professional, Redwood City, CA, USA.

Kupferman, O. and Vardi, M. Y. 2001. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic 2,* 3, 408–429.

Kupferman, O., Vardi, M. Y., and Wolper, P. 2000. An automata-theoretic approach to branching-time model checking. *Journal of the ACM 47,* 2, 312–360.

LASH. The Liège Automata-based Symbolic Handler. `http://www.montefiore.ulg.ac.be/~boigelot/research/lash/`.

Lesens, D., Halbwachs, N., and Raymond, P. 1997. Automatic verification of parameterized linear networks of processes. In *POPL '97: Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages.* ACM Press, New York, NY, USA, 346–357.

Lesens, D., Halbwachs, N., and Raymond, P. 2001. Automatic verification of parameterized networks of processes. *Theoretical Computer Science 256*, 113–144.

Löding, C. 2001. Efficient minimization of deterministic weak *omega*-automata. *Information Processing Letters 79,* 3 (July), 105–109.

Lynch, N. and Vaandrager, F. 1995. Forward and backward simulations I: Untimed systems. *Information and Computation 121,* 2, 214–233.

Maadani, K. and Geffroy, J.-C. 1991. Identification of structured automata for test evaluation. In *VLSI Test Symposium.*

Maler, O. and Pnueli, A. 1995. On the learnability of infinitary regular sets. *Information and Computation 118,* 2, 316–326.

Maler, O. and Staiger, L. 1997. On syntactic congruences for ω-languages. *Theoretical Computer Science 183,* 1, 93–112.

Miclet, L. 1980. Regular inference with a tail-clustering method. *IEEE Transactions on Systems, Man and Cybernetics 10*, 737–743.

Milner, R. 1971. An algebraic definition of simulation between programs. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence.* British Computer Society, London, 481–489.

Milner, R. 1995. *Communication and concurrency.* Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK.

Miyano, S. and Hayashi, T. 1984. Alternating finite automata on omega-words. *Theoretical Computer Science 32*, 321–330.

Murphy, K. 1996. Passively learning finite automata. Survey 96-04-017, Santa Fe Institute, Santa Fe, New Mexico, USA. Nov.

Myhill, J. 1957. Finite automata and the representation of events. Tech. Rep. WADD TR-57-624, Wright Patterson Air Force Base, Ohio.

Nerode, A. 1958. Linear automaton transformations. In *Proceedings of the American Mathematical Society.* Vol. 9. 541–544.

Pace, G., Halbwachs, N., and Raymond, P. 2004. Counter-example generation in symbolic abstract model-checking. *International Journal of Software Tools for Technology Transfer 5,* 2, 158–164.

Sankey, J. and Wong, R. K. 2001. Structural inference for semistructured data. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management.* ACM Press, New York, NY, USA, 159–166.

Saoudi, A. and Yokomori, T. 1994. Learning local and recognizable ω-languages and monadic logic programs. In *Euro-COLT '93: Proceedings of the first European conference on Computational learning theory.* Oxford University Press, Inc., New York, NY, USA, 157–169.

Semenov, A. L. 1977. Presburgerness of predicates regular in two number systems. *Sibirskii Matematicheskii Zhurnal 18*, 403–418. In Russian. English translation in *Siberian J. Math.***18** (1977), 289–300.

Staiger, L. 1983. Finite-state ω-languages. *Journal of Computer and System Sciences 27,* 3, 434–448.

Thomas, W. 1997. Languages, automata, and logic. *Handbook of formal languages: beyond words 3*, 389–455.

Touili, T. 2001. Regular model checking using widening techniques. *Electronic Notes in Theoretical Computer Science 50,* 4.

Trakhtenbrot, B. and Barzdin, Y. 1973. *Finite Automata: Behavior and Synthesis.* North Holland Publishing Company, Amsterdam.

Vardi, M. Y. and Wolper, P. 1986. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings, Symposium on Logic in Computer Science.* IEEE Computer Society, 332–344.

Vardi, M. Y. and Wolper, P. 1994. Reasoning about infinite computations. *Information and Computation 115,* 1, 1–37.

Wolper, P. and Boigelot, B. 1998. Verifying systems with infinite but regular state spaces. In *Computer Aided Verification, 10th International Conference, Proceedings.* Lecture Notes in Computer Science, vol. 1427. Springer, 88–97.

Yavuz-Kahveci, T., Bartzis, C., and Bultan, T. 2005. Action language verifier, extended. In *Computer Aided Verification, 17th International Conference, CAV 2005.* Springer, 413–417.