Tarski Fixed Point Computation and the Arrival Problem

Angus Joshi

4th Year Project Report
Computer Science and Mathematics
School of Informatics
University of Edinburgh

2024

Abstract

This skeleton demonstrates how to use the `infthesis` style for undergraduate dissertations in the School of Informatics. It also emphasises the page limit, and that you must not deviate from the required style. The file `skeleton.tex` generates this document and should be used as a starting point for your thesis. Replace this abstract text with a concise summary of your report.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Angus Joshi*)

Acknowledgements

Any acknowledgements go here.

Table of Contents

Chapter 1

Introduction

The preliminary material of your report should contain:

- The title page.
- An abstract page.
- Declaration of ethics and own work.
- Optionally an acknowledgements page.
- The table of contents.

As in this example `skeleton.tex`, the above material should be included between:

```
\begin{preliminary}
    ...
\end{preliminary}
```

This style file uses roman numeral page numbers for the preliminary material.

The main content of the dissertation, starting with the first chapter, starts with page 1. ***The main content must not go beyond page 40.***

The report then contains a bibliography and any appendices, which may go beyond page 40. The appendices are only for any supporting material that's important to go on record. However, you cannot assume markers of dissertations will read them.

You may not change the dissertation format (e.g., reduce the font size, change the margins, or reduce the line spacing from the default single spacing). Be careful if you copy-paste packages into your document preamble from elsewhere. Some LaTeX packages, such as `fullpage` or `savetrees`, change the margins of your document. Do not include them!

Over-length or incorrectly-formatted dissertations will not be accepted and you would have to modify your dissertation and resubmit. You cannot assume we will check your submission before the final deadline and if it requires resubmission after the deadline to conform to the page and style requirements you will be subject to the usual late penalties based on your final submission time.

1.1 Using Sections

Divide your chapters into sub-parts as appropriate.

1.2 Citations

Citations, such as **?** or (**?**), can be generated using `BibTeX`. We recommend using the `natbib` package (default) or the newer `biblatex` system.

You may use any consistent reference style that you prefer, including "(Author, Year)" citations.

Chapter 2

Background

2.1 Lattices, Monotone Functions, and Fixpoints

For the purposes of this project, I'm primarily concerned with an order-theoretic characterization of lattices. I begin with some definitions.

Definition 2.1.1 (Poset)**.** A *partially ordered set*, or *poset* is a set S with a binary relation $\leq$ on S such that the following axioms hold,

- For all $s \in S$, $s \leq s$ (reflexivity),
- for all $s,t,u \in S$, if $s \leq t$ and $t \leq u$ then $s \leq u$ (transitivity),
- for all $s,t \in S$, if $s \leq t$ and $t \leq s$ then $s = t$ (antisymmetry).

Definition 2.1.2 (Least/Greatest Upper/Lower Bounds)**.** Let $(S, \leq)$ be a partially ordered set and $A \subseteq S$. A *greatest lower bound* of A is a $l \in S$ such that for all $a \in A$ $a \geq l$, and whenever $l' \in S$ also satisfies for all $a \in A$ $a \geq l'$ I have $l \geq l'$. When it exists is denoted $l = \bigwedge A$. A *least upper bound* of A is a $u \in S$ such that for all $a \in A$ $a \leq u$, and whenever $u' \in S$ also satisfies for all $a \in A$ $a \leq u'$ I have $u \leq u'$. When it exists it is denoted $u = \bigvee A$.

Remark 2.1.3. Least upper bounds and greatest lower bounds need not exist in general. Take for example $(\mathbb{Z}, \leq)$ which is claimed to be a partial ordering. $\mathbb{Z}$ is a subset of $\mathbb{Z}$, but clearly $\mathbb{Z}$ has no upper bounds. Boundedness is also in general not suffiecient.

Lemma 2.1.4 (Least/Greatest Upper/Lower bounds are unique)**.** *Let L be a lattice, $A \subseteq L$ and u, u' least upper bounds of A. Then $u = u'$. If l, l' are greatest lower bounds of A then $l = l'$.*

Proof. The second part of the definition on a least upper bound gives $u \leq u'$ and $u' \leq u$. Then antisymmetry of a partial order gives $u = u'$. That $l = l'$ follows similarly. ■

Definition 2.1.5 (Complete Lattice)**.** Let $(L, \leq)$ be a partially ordered set. $(L, \leq)$ is a *complete lattice* if for all $A \subseteq L$ there is a least upper bound $u = \bigvee A$ and a greatest lower bound $l = \bigwedge A$.

Definition 2.1.6 (Product Lattice)**.** Given two complete lattices $(L, \leq)$ and $(L', \leq')$ the *product lattice* $(L \times L, \leq \times \leq')$ is the cartesian product of the underlying sets with $(l, l') \leq \times \leq' (m, m')$ if and only if $l \leq m$ and $l' \leq m'$ for all $l, m \in L$ and $l', m' \in L'$.

That a finite product of complete lattices is also a complete lattice is taken as fact. The notion of a complete sublattice will in turn be useful as well.

Definition 2.1.7 (Sublattice)**.** Let $(L, \leq)$ be a lattice. A complete sublattice $(M, \leq)$ is a subset $M \subseteq L$ such that for all $N \subseteq M$ I have $\bigvee N \in M$ and $\bigwedge N \in M$.

For the rest of the dissertation where the ordering is clear from context, I will denote a lattice purely by it's underlying set.

Notation 2.1.8. For $N \in \mathbb{Z}_{\geq 1}$ I use the notation $[N] = \{1, ..., N\}$.

It is clear that any finite totally ordered set is a complete lattice.

Corollary 2.1.9. *For $d \in \mathbb{Z}_{\geq 1}$ let $[N]^d = \prod_{i=1}^{d} [N]$. Then $[N]^d$ is a lattice. The ordering is defined as $(l_1, ..., l_d) \leq (l'_1, ..., l'_d)$ if and only if $l_i \leq l'_i$ for each $i \in \{1, ..., d\}$.*

At times I will also need continuous lattices. I take the result from elementary analysis that all bounded subsets of the real numbers have a least upper bound and greatest lower bound in the real numbers to find that $[a, b] \subseteq \mathbb{R}$ is a complete lattice for all $a, b \in \mathbb{R}$ under the usual ordering.

Corollary 2.1.10. *For $d \in \mathbb{Z}_{\geq 1}$ let $[0, 1]^d = \prod_{i=1}^{d} [0, 1]$. Then $[0, 1]^d$ is a lattice. The ordering is defined as $(l_1, ..., l_d) \leq (l'_1, ..., l'_d)$ if and only if $l_i \leq l'_i$ for each $i \in \{1, ..., d\}$.*

The focus will be on finding so-called fixpoints of monotone functions on a lattice.

Definition 2.1.11 (Monotone Function)**.** Let L be a lattice. Then a function $f : L \to L$ is *monotone* if whenever $l, l' \in L$ with $l \leq l'$ I have $f(l) \leq f(l')$.

Definition 2.1.12 (Fixpoint)**.** Let S be a set and $f : S \to S$. Then $s \in S$ is a *fixpoint* of f if $f(s) = s$.

The notion of a monotone point will also be useful.

Definition 2.1.13 (Monotone Point)**.** Let L be a lattice and $f : L \to L$ be monotone. Then $x \in L$ is a *monotone point* if either $f(x) \geq x$ or $f(x) \leq x$. It is *monotone up* if $f(x) \geq x$ and *monotone down* if $f(x) \leq x$.

Notation 2.1.14. Let L be a complete lattice and $a, b \in L$. The notation $[a, b] = \{x \in L : a \leq x \leq b\}$ is sometimes used. The reader can verify that this is indeed a complete sublattice of L.

And now for Tarski.

Theorem 2.1.15 (Tarski's Fixed Point Theorem (**?**))**.** *Let L be a complete lattice and $f : L \to L$ a monotone function. Define $Fix(f) = \{x \in L : f(x) = x\}$. Then $Fix(f)$ is a complete lattice under the same ordering as L. In particular, $Fix(f)$ contains a least fixpoint $l = \bigwedge Fix(f)$.*

2.2 Basic Algorithms

Although Tarski's theorem easily subsumes the finite case detailed below, I include a basic proof of the existence of fixpoints of monotone functions of finite lattices as the ideas involved will be useful in the development of algorithms.

Theorem 2.2.1 ((**?**)). *Let $f : [N]^d \to [N]^d$ be monotone. Then there is a point $x^* \in [N]^d$ such that $f(x^*) = x^*$.*

Proof. Firstly, note that for all $x \in [N]^d$ the point $\vec{1} = (1, ..., 1) \leq x$, and in particular $\vec{1} \leq f(\vec{1})$. By an induction combined with monotonicity I find for all $i \in \mathbb{Z}_{\geq 0}$, $f^i(\vec{1}) \leq f^{i+1}(\vec{1})$. Suppose for a contradiction that there is no point $x^* \in [N]^d$ such that $f(x^*) = x^*$. Then for all $i \in \mathbb{Z}_{\geq 0}$, $f^i(\vec{1}) \neq f^{i+1}(\vec{1})$ which implies that $f^i(\vec{1}) < f^{i+1}(\vec{1})$. This gives infinitely many distinct points in $[N]^d$. But $[N]^d$ is finite, which is a contradiction. It follows that there is a fixpoint of f in $[N]^d$. ■

Definition 2.2.2 (TARSKI). The problem TARSKI(N, d) is, given oracle access to a monotone function $f : [N]^d \to [N]^d$, find a point $x^* \in [N]^d$ such that $f(x^*) = x^*$.

The proof of **??** implicitly contains our first algorithm for fixpoint computation.

Notation 2.2.3. For $k \in \mathbb{Z}_{\geq 0}$ the notation $\vec{k} = (k, ..., k)$. It is assumed that the dimensionality of this 'vector' is clear from context.

Algorithm 1 Kleene Tarski Iteration

```
1: procedure KLEENETARSKI(monotone $f : [N]^d \to [N]^d$)
2:     $x \leftarrow \vec{1}$
3:     while $x \neq f(x)$ do
4:         $x \leftarrow f(x)$
       return $x$
```

Correctness of the algorithm if it terminates is clear, so all that is needed it a bound on it's runtime.

Lemma 2.2.4. KLEENETARSKI *always terminates in time $O(Nd)$.*

Proof. As in the proof of **??**, for all $i \in \mathbb{Z}_{\geq 0}$, $f^i(\vec{1}) \leq f^{i+1}(\vec{1})$. If $f^i(\vec{1}) = f^{i+1}(\vec{1})$ then $f^i(\vec{1})$ is a fixpoint. So suppose for a contradiction for some $j > Nd$ that for all $i \leq j$, $f^i(\vec{1}) < f^{i+1}(\vec{1})$. By integrality, $\|f^{i+1}(x)\|_1 \geq \|f^i(x)\|_1 + 1$. It follows that $\|f_j(x)\|_1 > Nd$. But this implies that $\|f_j(x)\|_1 > \|\vec{N}\|_1$, which is a contradiction of the definition of f. So for some $j \leq Nd$, $f^j(\vec{1}) = f^{j+1}(\vec{1})$. ■

Theorem 2.2.5. *The query complexity of* TARSKI(N, d) *is $O(Nd)$.*

The fixpoint returned by KLEENETARSKI isn't just any old fixpoint.

Lemma 2.2.6. *Let x be the fixpoint returned by* KLEENETARSKI. *Then x is the least-fixpoint of f. That is, for all other fixpoints $y \in [N]^d$, $y \leq x$.*

Proof. Let $(f^i(\vec{1}))_{i\in\mathbb{Z}_{\geq 0}}$ be the sequence of points generated in KLEENETARSKI. I will show inductively that $f^i(\vec{1}) \leq y$ for all $i \in \mathbb{Z}_{\geq 0}$. For the base case, by construction of the lattice $f^0(\vec{1}) = \vec{1} \leq y$. Suppose $f^{i-1}(\vec{1}) \leq y$. Then by monotonicity of f, $f(f^{i-1}(\vec{1})) = f^i(\vec{1}) \leq f(y)$. But y is a fixpoint so $f(y) = y$ and $f^i(\vec{1}) \leq y$. ■

It should be emphasized that this is algorithm does *not* run in time polynomial with the encoding size of problem instances. For numbers of size 2^n can be represented with a n bits of information. At the time of writing it is not known whether or not this problem is solvable in polynomial time. Etessami et al. gave the current best known lower bound on the query complexity of TARSKI, along with other complexity-theoretic results on the problem.

Theorem 2.2.7 ((**?**)). *The query complexity of* TARSKI(N,d) *is* $\Omega(\log^2 N)$.

Dang, Qi, and Ye gave an algorithm for solving the TARSKI problem(**?**) using a variant of the well known binary search algorithm. The details of their algorithm are instructive to the workings on the improved algorithms detailed later, so they are given below.

Notation 2.2.8. Given a tuple $x = (x_1,\ldots,x_n)$ for $i \in [n]$ the notation $x_{-i} = (x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$. That is, it drops the i-th coordinate of the tuple.

Definition 2.2.9 (Slice). Let $(S_i)_{i\in[d]}$ be totally ordered sets, $L = \prod_{i\in[d]} S_i$ be their product lattice, and $f : L \to L$ be monotone. Then a *slice* of f is a choice of coordinate $i \in [d]$, and a choice of value $x_i \in S_i$, defining a new function $f_s : L_{-i} \to L_{-i}$ with $f_s((l_1,\ldots,l_{d-1})) = f((l_1,\ldots,l_{i-1},x_i,l_i,\ldots,l_{d-1}))_{-i}$.

Lemma 2.2.10. *Let $f : [N]^d \to [N]^d$ be monotone. Then for any $i \in [d]$, $x_i \in [N]$ the slice $f_s : [N]^{d-1} \to [N]^{d-1}$ at i with value x_i is monotone.*

Proof. Suppose $l, l' \in [N]^{d-1}$ with $l = (l_1,\ldots,l_{d-1})$, $l' = (l'_1,\ldots,l'_{d-1})$, and $l \leq l'$. By reflexivity, $x_i \leq x_i$, so $(l_1,\ldots,x_i,l_i,\ldots,l_{d-1}) \leq (l'_1,\ldots,x_i,l'_i,\ldots,l_{d-1})$, and $f_s(l) \leq f_s(l')$ follows by monotonicity of f. ■

Notation 2.2.11. Let L be a lattice. Then for $x \in L$ the notation $L_{\geq x}$ is a sub-lattice of L with underlying set $\{l \in L | l \geq x\}$. It is clear that $L_{\geq x}$ is also a lattice, with the same joins, meets, and ordering as L.

Lemma 2.2.12. *Let $f : [N]^d \to [N]^d$ is monotone, and $x \in [N]^d$ be such that $x \leq f(x)$. Then f restricts to a monotone function $f|_{[N]^d_{\geq x}} : [N]^d_{\geq x} \to [N]^d_{\geq x}$. Similarly, if $x \geq f(x)$ then f restricts to a monotone function $f|_{[N]^d_{\leq x}} : [N]^d_{\leq x} \to [N]^d_{\leq x}$.*

Proof. I need to show that if $x \leq f(x)$ then for all $y \in [N]^d_{\geq x}$, $f(y) \in [N]^d_{\geq x}$. By construction, $y \geq x$, and by monotonicity $f(y) \geq f(x)$. But $f(x) \geq x$, so $f(y) \geq x$, and $f(y) \in [N]^d_{\geq x}$. The second part follows by duality. ■

Lemma 2.2.13. *Let $f : [N] \to [N]$ be monotone. Then a fixpoint of f can be found in $O(\log N)$ queries of f.*

Proof. Choose $x = \lfloor \frac{N}{2} \rfloor$. $[N]$ is totally ordered, so exactly one of the following hold; $x < f(x)$, $x = f(x)$, $x > f(x)$. If $x = f(x)$ then I'm done. If $x < f(x)$ then by **??** f restricts to a monotone function $f|_{[N]^d_{\geq x}}$, and a fixpoint of $f|_{[N]^d_{\geq x}}$ is clearly also a fixpoint of f. Similarly, if $x > f(x)$ then f restricts to $f|_{[N]^d_{\leq x}}$. This enables a recursion on the smaller sublattice. Finally, noting that a fixpoint can be found trivially in the one-point set in a constant number of queries, since the search space is halved every recursive call the algorithm terminates in $O(\log N)$ queries. ■

The algorithm of Dang, Qi, and Ye can be seen in **??**.

Algorithm 2 (**?**)

```
1: procedure DangQiYe(monotone $f : [N]^d \to [N]^d$)
2:     $\bot \leftarrow \vec{1}$
3:     $\top \leftarrow \vec{N}$
4:     return DangQiYeRec($f$, $\bot$, $\top$)
5: procedure DangQiYeRec(monotone $f : [N]^d \to [N]^d$, $\bot$, $\top$)
6:     while true do
7:         $\mathrm{mid}_d \leftarrow \lfloor \frac{\bot_d + \top_d}{2} \rfloor$
8:         $f_s \leftarrow$ the slice of $f$ at $d$ with value $\mathrm{mid}_d$
9:         $\vec{x_s} \leftarrow$ DangQiYe($f_s$, $\bot_{-d}$, $\top_{-d}$)
10:        $\vec{x} \leftarrow ((\vec{x_s})_1, \ldots, (\vec{x_s})_{d-1}, \mathrm{mid}_d)$
11:        if $\vec{x}_d = f(\vec{x})_d$ then
12:            return $\vec{x}$
13:        if $\mathrm{mid}_d < f(\vec{x})_d$ then
14:            $\bot \leftarrow \vec{x}$
15:        if $\mathrm{mid}_d > f(\vec{x})_d$ then
16:            $\top \leftarrow \vec{x}$
```

Lemma 2.2.14. DangQiYe *returns a fixpoint of f if it terminates.*

Proof. The algorithm only returns if it satisfies the condition on line 11. At this point, $\vec{x_s}$ is a fixpoint of f_s, so it follows that $f(\vec{x})_i = \vec{x}_i$ for $i \in [d-1]$. The condition ensures that also $\vec{x}_d = f(\vec{x})_d$, and $\vec{x}$ is a fixpoint of f. ■

Lemma 2.2.15. DangQiYe *terminates in at most $O(\log^d N)$ queries to f.*

Proof. By induction. The base case follows from **??**. Suppose DangQiYe uses at most $O(\log^{d-1} N)$ queries to solve the $d-1$ dimensional case. If the condition on line 11 fails, note that $\vec{x}$ is a monotone point, so **??** guarantees that the function restricts to the smaller lattice bounded by lines 14 or 16. The d-th dimension is shrunk by a factor of $\frac{1}{2}$ every iteration of the loop, so will require at most $\log N$ recursive calls to the $d-1$ dimensional algorithm. So the algorithm terminates using at most $O(\log N) \cdot O(\log^{d-1} N) = O(\log^d N)$ queries to f. ■

Theorem 2.2.16 ((**?**)). *The query complexity of* Tarski(N, d) *is* $O(\log^d N)$.

Simply combining the theorems of Dang, Qi, and Ye, and Etessami et al. gives a tight bound on the 2 dimensional problem.

Corollary 2.2.17 ((**?**)). *The query complexity of* TARSKI$(N,2)$ *is* $\Theta(\log^2 N)$.

There have recently been improvements on this upper bound. In paricular, Fearnley et al. gave an algorithm for solving the 3 dimensional version of the problem. This used the crucial fact that, in contrast to the Dang, Qi, and Ye algorithm, restricting to a half-space does *not* require a point which is fixed in $n-1$ dimensions, but a point which is monotone in all n dimensions. They gave a so-called 'inner algorithm' which in three dimensions can find a monotone point with a particular value in one coordinate in $O(\log N)$ queries. Chen and Li showed how to decompose the problem of finding monotone points in higher dimensions - crucially using the inner algorithm - to a product of sorts of 2 dimensional prolems, giving the current best known upper bound on the TARSKI problem.

Theorem 2.2.18 ((**?**)). *The query complexity of* TARSKI$(N,3)$ *is* $\Theta(\log^2 N)$.

Theorem 2.2.19 ((**?**)). *The query complexity of* TARSKI(N,d) *is* $O(\log^{\lceil (d+1)/2 \rceil} N)$.

The details of these algorithms will be shared in a later section.

Remark 2.2.20. The problem of computing any fixpoint of a monotone function on a general lattice is known to be intractable. In particular, it was shown in (**?**) that for any randomized algorithm there are general lattices of N elements which require $\Omega(N)$ queries with high probability to find a fixpoint. Focussing on a specific case such as $[N]^d$ is therefore justified. Further, the case of computing least fixpoints of even one dimensional functions $f : [2^n] \to [2^n]$ is proven to be NP-hard in (**?**) so would seem that focussing on find any fixpoint is the best course of action.

Chapter 3

State of the art Algorithms

3.1 Overview

Recent developments have been made in upper bounds for the TARSKI problem. The critical component to the algorithmic improvements is the surprising result from (**?**) that TARSKI$(N,3)$ can be solved in at most $O(\log^2 N)$ queries. The idea is roughly as follows; suppose I have a monotone function $f : [N]^3 \to [N]^3$ and an algorithm which given a coordinate $i \in \{1,2,3\}$ and a value $v \in [N]$ returns a monotone point $x \in [N]^3$ with the guarantee that $x_i = v$. If this algorithm takes $q(N)$ queries in the worst case, then a fixpoint of f can be found in $O(\log N \cdot q(N))$ in the same fashion as **??**. This sub-problem will prove to be important, so I will define it in it's own right.

Definition 3.1.1 (TARSKI*)**.** The problem TARSKI*(N,d) is, given oracle access to a monotone function $f : [N]^d \to [N]^d$, a coordinate $i \in \{1,...,d\}$, and value $v \in [N]$, find a monotone point $x \in [N]^d$ such that $x_i = v$.

The breakthrough in (**?**) was in detailing a so-called inner-algorithm which leads to the following result.

Theorem 3.1.2 ((**?**))**.** *The query complexity of* TARSKI*$(N,3)$ *is* $O(\log N)$.

Fearnley et al. also gave a method of decomposition, allowing for TARSKI in higher dimensions to be decomposed into a product of sorts of lower dimensional problems. This can be seen as a generalization of the method in **??**. Dang et al. solve TARSKI(N,d) by solving an instance A of TARSKI$(N,1)$ where every query of the monotone function from TARSKI$(N,1)$ at v is answered by recursively finding a fixpoint in the $d-1$ dimensional slice at the d-th coordinate with value v. It then follows that a soution to A can be used to recover a solution of the entire problem. The algorithm of Fearnley et al. shows that it is possible to make more general decompositions using a method which I call *fixpoint decomposition*. There is slight complication with making this precise which will be detailed in a later section of this chapter.

Theorem 3.1.3 (Fixpoint Decomposition, (**?**))**.** *For positive integers $a,b \in \mathbb{Z}_{>0}$, given an algorithm A which can solve* TARSKI(N,a) *in $p(N,a)$ queries, and an algorithm B which can solve* TARSKI(N,b) *in $p(N,b)$ queries, the problem* TARSKI$(N,a+b)$ *can* *be solved in $O(p(N,a)\cdot q(N,b))$ queries.*

Chen and Li take this one step further; instead of decomposing the TARSKI problem they show that it is possible to make a decomposition on TARSKI* using a method I will call *monotone decomposition*. The details of this are more complicated still than **??**, and will be shared in a later section.

Theorem 3.1.4 (Monotone Decomposition, (**?**))**.** *For positive integers $a,b \in \mathbb{Z}_{>0}$, given an algorithm A which can solve* TARSKI*(N,a) *in $p(N,a)$ queries, and an algorithm B which an solve* TARSKI*(N,b) *in $q(N,b)$ queries, the problem* TARSKI*$(N,a+b)$ *can be solved in $O((b+1)\cdot p(N,a)\cdot q(N,b))$ queries.*

3.2 The Inner Algorithm

I will not detail the inner algorithm in it's entirety, nor prove it's correctness; the reader is instead directed to (**?**). Instead I will introduce the main invariant, showing how the algorithm can make progress in the important cases. I require a weakening of the invariant used in **??**; whereas in **??** two opposing monotone points are maintained essentially defining the top and bottom of a sublattice on which the monotone function naturally restricts (and hence contains a fixpoint), the inner algorithm maintains an invariant so that only a monotone point is guaranteed within the current search space.

Definition 3.2.1 (Witness, (**?**))**.** Let $f : [N]^3 \to [N]^3$. A *down set witness* is a pair of points $(d,b) \in ([N]^3)^2$ such that $d_3 = b_3$ and for some $i, j \in \{1,2\}$ with $i \neq j$,

- $d_i = b_i$, $b_j \leq d_j$, $f(b)_j \geq b_j$, and if $b \neq d$ then $f(d)_j \leq d_j$,
- $f(d)_3 \geq d_3$ and $f(b)_3 \geq b_3$.

An *up set witness* is a pair of points $(a,u) \in ([N]^3)^2$ such that $a_3 = u_3$ and for some $i, j \in \{1,2\}$ with $i \neq j$,

- $a_i = u_i$, $a_j \leq u_j$, $f(a)_j \geq a_j$, and if $u \neq a$ then $f(u)_j \leq u_j$,
- $f(d)_3 \geq d_3$ and $f(b)_3 \geq b_3$.

I use a notational convenience from (**?**).

Notation 3.2.2. Let $f : L \to L$ be a monotone function. Then $\text{Up}(f) = \{x \in L : x \leq f(x)\}$ and $\text{Down}(f) = \{x \in L : x \geq f(x)\}$. At times an abuse of notation is used, where elements of the lattice $x \in [N]^d$ are said to be in the up set or down set of the slice f_s of a monotone function $f : [N]^d \to [N]^d$. The meaning is assumed to be clear from context.

[add ref to figure with witness diagrams] shows the different possible configurations of witnesses to aid the reader in digesting this definition. Now for the main invariant of the inner algorithm.

Definition 3.2.3 (Inner algorithm invariant)**.** Let $f : [N]^3 \to [N]^3$ be a monotone function. The *inner algorithm invariant* is an up set witness (d,b) and a down set witness (a,u) such that $u \leq d$.

Lemma 3.2.4. *Let $f : [N]^3 \to [N]^3$ be a monotone function with (a,u), (d,b) up and down set witnesses respectively satisfying the inner algorithm invariant. Then there is a point $x \in [N]^3$ with $a \leq x \leq b$ and $x \in Up(f)$ or $x \in Down(f)$.*

Proof. I will prove the case that $d_1 = b_1$ and $a_1 = u_1$, and the others follow similarly. Let f_s be the slice of f at coordinate 3 with value $b_3 = a_3 = d_3 = u_3$ (where all of these equalities follow from the witness definitions). If $a = u$ and $b = d$ then $a \in \mathrm{Up}(f_s)$ and $b \in \mathrm{Down}(f_s)$, so **??** guarantees fixpoint of f_s in the range $[a,b]$, which is then necessarily a monotone point.
If $a \neq u$ then **??** gives a point $c \in [N]^3$ with $a \leq c \leq u$ and $f(c)_2 = c_2$. By monotonicity and $c \leq u$ I have $f(c)_3 \leq c_3$. If $f(c)_1 \leq c_1$ then $c \in \mathrm{Down}(f)$ and I'm done. If $f(c)_1 \geq c_1$ then I consider b and d. Using a similar argument there is a $v \in [N]^3$ with $d \leq v \leq b$ such that $f(v)_3 \geq v_3$, and $f(v)_2 = v_2$. If $f(v)_1 \geq v_1$ then $v \in \mathrm{Up}(f)$, and if $f(v)_1 \leq v_1$ then **??** guarantees a fixpoint of f_s in the range $[c,v]$, which is neccessarily monotone. ■

The key is then to half the search instance in a constant amount of time by finding a new set of points satisfying the inner algorithm invariant. There are many distinct cases to be handled here and I will not cover them all. In particular, special cases are required for instances which have are narrow (that is, for some $i \in \{1,2,3\}$ $b_i - a_i \leq 1$). The reader is directed to (**?**) for these. The first case is under the assumption that u and d are not past the midpoint of the line they are on. Throughout I set $\mathrm{mid}_i = \left\lfloor \frac{a_i+b_i}{2} \right\rfloor$ for each $i \in \{1,2,3\}$, and take f_s to be the slice of f at the 3rd coordinate with value $a_3 = b_3$.

Lemma 3.2.5 ((**?**)). *Suppose for each $i \in \{1,2\}$ that $u_i \leq \left\lfloor \frac{a_i+b_i}{2} \right\rfloor$ and $d_i \geq \left\lfloor \frac{a_i+b_i}{2} \right\rfloor$. Then a pair of witnesses $(\overline{a},\overline{u})$, $(\overline{d},\overline{b})$ satisfying the invariant can be found such that for some $j \in \{1,2\}$ $\overline{b}_j - \overline{a}_j \leq \left\lceil \frac{a_j-b_j}{2} \right\rceil$ using a constant number of queries.*

Proof sketch. Evaluate $f(\mathrm{mid})$. If $\mathrm{mid} \in \mathrm{Up}(f)$ or $\mathrm{mid} \in \mathrm{Down}(f)$ then the inner algorithm can return immediately. If $\mathrm{mid} \in \mathrm{Up}(f_s)$ and $\mathrm{mid} \notin \mathrm{Up}(f)$ then $f(\mathrm{mid})_3 \leq \mathrm{mid}$ and I can set $\overline{a} = \overline{u} = \mathrm{mid}$. By assumption $u \geq \mathrm{mid}$ so the pair $(\overline{a},\overline{u})$, (d,b) will do. The case $\mathrm{mid} \in \mathrm{Down}(f_s)$ is similar. Suppose for some $i,j \in \{1,2\}$ with $i \neq j$ that $f(\mathrm{mid})_i \leq \mathrm{mid}_i$ and $f(\mathrm{mid})_j \geq \mathrm{mid}_j$. If $f(\mathrm{mid})_3 \geq \mathrm{mid}_3$ then put $p_j = b_j$, $p_i = \mathrm{mid}_i$ and $p_3 = \mathrm{mid}_3$ and evaluate $f(p)$. By monotonicity $f(p)_3 \geq p_3$, so if $f(p)_j \leq p_j$ then put $\overline{u} = \mathrm{mid}$ and $\overline{a} = p$ and I'm done. If $f(p)_j > p_j$ then by monotonicity and $b_j = p_j$ I have $f(b)_j > b_j$. It follows by definition of witnesses that if $b \neq d$ then $b_i \neq d_i$ and $b_j = d_j$. Then again by monotonicity and $d \leq p$ I have $d_j \leq f(d)_j$, and by definition of a witness $d_i \leq f(d)_i$ and $d_3 \leq f(d)_3$, so $d \in \mathrm{Up}(f)$ and can be returned immediately. The case $f(\mathrm{mid})_3 \leq \mathrm{mid}_3$ is similar and ommitted for brevity. ■

[insert ref to a figure here] should be examined to help understand what is happening in the proof sketch of **??**. I must now justify the assumption in **??** that $u_i \leq \mathrm{mid}$ and $d_i \geq \mathrm{mid}$.

Lemma 3.2.6 ((**?**)). *Suppose for some $i \in \{1,2\}$ that $u_i \nleq mid_i$. Then a point $p \in [N]^3$ can be found such that $p_i \leq mid_i$ and (a,p) is a valid down-set witness, or (p,u) is* *a valid down-set witness. If $d_i \not\geq$ mid then a point $q \in [N]^3$ can be found such that $q_i \geq$ mid and (q,b) is a valid up-set witness, or (d,q) is a valid up-set witness.*

Proof sketch. Suppose for some $i \in \{1,2\}$ I have $u_i > \text{mid}_i$. Then by definition of a witness, if $j \in \{1,2\}$ and $i \neq j$ then $u_j = a_j$. Put $p_j = u_j$, $p_i = \text{mid}_i$. Then since $u \geq p$ by monotonicity and definition of a witness I have $f(u)_3 \geq u_3$. If $f(p)_i \leq p_i$ then (a,p) is a valid down set witness that satisfies $p_i \leq \text{mid}_i$. If $f(p)_i \geq p_i$ then (p,u) is a valid down-set witness. The case with $d_i < \text{mid}_i$ is similar. ■

[ref to a figure] shows visually how the proof of **??** works. Combining the previous two lemmas, a 'normal' iteration of the inner algorithm is as follows. Check if $u <$ mid or $d >$ mid. If so, using **??** either find a new set of witnesses with a search space that is half the size of the previous, or a new of witnesses such that $u \leq$ mid and $d \geq$ mid. Then continue to the next iteration. If $u \geq$ mid and $d \leq$ mid then using **??** find a new set of witnesses such that the new search space is at most half the size of the previous. With the additional fact from (**?**) that the search space can be halfed in a constant amount of queries when the search space as a width of at most 1, this gives an $O(\log N)$ query algorithm for the TARSKI*$(N,3)$ problem.

3.3 Fixpoint Decomposition

Throughout I will fix $a,b \in \mathbb{Z}_{>0}$, $d = a+b$, and $f : [N]^a \times [N]^b \to [N]^a \times [N]^b$. Suppose I have an algorithm A which can solve TARSKI(N,a), and an algorithm B which can solve TARSKI(N,b). A naive approach for finding a fixpoint of f would be to define a function on the right hand side of the lattice $f_r : [N]^b \to [N]^b$ where given $x_r \in [N]^b$ the value of $f_r(x_r)$ is computed by defining the slice $f_l : [N]^a \to [N]^a$ such that $f_l(x_l) = f((x_l,x_r))$, finding a fixpoint $x_l^* \in [N]^a$ of f_l, then using $f(x_l^*,x_r)_{-l}$ as the result of $f_r(x_r)$. The punch-line is then that if x_r is a fixpoint of f_r, then if x_l^* was the fixpoint of f_l associated with x_r the point (x_l^*,x_r) is a fixpoint of f. This does not work however - there is no guarantee that f_r is monotone. That is, if points $x_r, x_r' \in [N]^b$ are queried by algorithm B with $x_r \leq x_r'$ it is not necessarily the case that the associated fixpoints of f_l x_l^* and $x_l'^* \in [N]^a$ satisfy $x_l^* \leq x_l'^*$, so monotonocity of f does not carry over. The trick will thus be to find a way to guarantee that x_l* and $x_l'^*$ satisfy $x_l^* \leq x_l'^*$ whenever $x_r \leq x_r'$. Fortunately this is achieveable; in **??** the issue is solved by carefully choosing bounds of the sublattice to search in based on previously computed points. The correctness of which will be the concern of the remainder of this section.

Lemma 3.3.1. *If $(p_l,p_r),(p_l',p_r') \in prev$ with $p_r \leq p_r'$ then $p_l \leq p_l'$*

Proof. Suppose without loss of generality that p_r was queried by algorithm B before p_r'. Then when p_r' is queried, p_l is an element of $\{pl : (pl,pr) \in \text{prev} : p_r \leq p_r'\}$ whence $\bot_l \geq p_l$, so the fixpoint found by algorithm A satisfies $x_l^* \geq p_l$. ■

Lemma 3.3.2. *At line 7 of* **??** $\bot_l \leq \top_l$.

Proof. By **??** I have for all $p_l \in \{p_l : (p_l,p_r) \in \text{prev} : p_r \leq x_r\}$ and $p_l' \in \{p_l : (p_l,p_r) \in \text{prev} : p_r \geq x_r\}$ that $p_l \leq p_l'$. Then by definition of $\wedge$ and $\vee$ $\bot_l \leq \top_l$. ■

Algorithm 3 (?)

```
1: procedure FixpointDecomposition(
     monotone $f : [N]^a \times [N]^b \to [N]^a \times [N]^b$,
     algorithm $A$ for solving Tarski$(N, a)$,
     algorithm $B$ for solving Tarski$(N, b)$ )
2:     prev $\subseteq [N]^a \times [N]^b \leftarrow \emptyset$
3:     procedure $f_r(x_r \in [N]^b)$
4:         procedure $f_l(x_l \in [N]^a)$ return $f((x_l, x_r))_{-r}$.
5:         $\bot_l \leftarrow \bigvee\{p_l : (p_l, p_r) \in \text{prev} : p_r \leq x_r\}$
6:         $\top_r \leftarrow \bigwedge\{p_l : (p_l, p_r) \in \text{prev} : p_r \geq x_r\}$
7:         using algorithm $A$ find a fixpoint $x_l^*$ of $f_l$ in the sublattice with bounds $\bot_l$, $\top_l$.
8:         prev $\leftarrow$ prev $\cup \{(x_l^*, x_r)\}$
9:         return $f((x_l^*, x_r))_{-l}$.
10:    run algorithm $B$ on $f_r$ to find fixpoint $x_r^*$
11:    return $(x_l^*, x_r^*)$ where $x_l^*$ is the fixpoint of $f_l$ found when evaluating $f(x_r^*)$.
```

Lemma 3.3.3. *At line 7 of* **??** *I have* $f_l(\bot_l) \geq \bot_l$ *and* $f_r(\top_l) \leq \top_l$.

Proof. I prove the first case and the second is similar. Suppose not. That is, for some $i \in [a]$ I have $f_l(\bot_l)_i < (\bot_l)_i$. If $L = \{p_l : (p_l, p_r) \in \text{prev} : p_r \leq x_r\}$ is empty then $\vee L = \vec{1}$ and the claim holds. If L is non-empty then by definition of finite joins, there is some $(p_l, p_r) \in$ prev such that $(p_l)_i = (\bot_l)_i$ and $(p_l, p_r) \leq (x_l, x_r)$. But $f(((p_l, p_r))_{-r})_i = (p_l)_i$ so I contradict monotinicity. ■

Lemma 3.3.4. *At line 7 of* **??** f_l *is a monotone function on the lattice* $[\bot_l, \top_l]$.

Proof. That f_l is monotone follows from an inductive application of **??**. Then the rest follows from **??** and **??**. ■

Proposition 3.3.5. *The point* (x_l^*, x_r^*) *returned by* **??** *is a fixpoint of* f.

Proof. **??** guarantees that f_r is monotone, and hence on line 10 a fixpoint x_r^* can be found. Then by construction (x_l^*, x_r^*) is clearly a fixpoint of f. ■

Proof of **??**. Suppose algorithm A takes at most $p(N, a)$ queries and algorithm B takes at most $q(N, a)$ queries to find a fixpoint. Then every query of f_r by algorithm A makes at most $q(N, b)$ queries to f to find a fixpoint of f_l, so given algorithm A makes a most $p(N, a)$ queries, the entire algorithms makes at most $p(N, a) \cdot q(N, b)$ queries to f. ■

3.4 Monotone Decomposition

Chapter 4

Related Problems

4.1 The Arrival Problem

The arrival problem is, given a directed graph with a particular structure and designated source and target vertex, decide whether or not a particular walk starting at the source ever reaches the target. The beginnings of this section will be in making this idea precise.

Definition 4.1.1 (Arrival Graph)**.** An *arrival graph* is a set of vertices V, a pair of vertices $s,t \in V$, and a pair of maps $s_0, s_1 : V \to V$.

Definition 4.1.2 (Arrival Walk)**.** Let (V,s,t,s_0,s_1) be an arrival graph. The *arrival walk* on this graph is a sequence of vertices $(v_i)_{i\in\mathbb{Z}_{\geq 0}} \in V$ such that $v_0 = s$, and $v_{i+1} = \begin{cases} s_0(v_i), & n_i \text{ even} \\ s_1(v_i), & n_i \text{ odd}, \end{cases}$ where n_i is the number of times v_i has appeared previously in the sequence.

A diagram of an example arrival graph is shown in in **??**. It is clear that the arrival walk for a particular arrival graph is entirely defined by the structure of the graph, which is what lead it to be called a zero player graph game in (**?**).

Definition 4.1.3 (ARRIVAL)**.** The ARRIVAL problem is, given an arrival graph (V,s,t,s_0,s_1), decide whether or not the arrival walk ever reaches t.

There is an obvious algorithm to solve the ARRIVAL problem; just simulate the walk. Cases of instances where t is not reachable pose a problem however - the walk must cycle infinitely and never terminate! The following content demonstrates that this is a non issue.

Definition 4.1.4 (Hopeful and Desperation)**.** Let (V,s,t,s_0,s_1) be an instance of the ARRIVAL problem. A vertex $v \in V$ is *hopeful* if there is a path $v \to t$ in the directed graph defined with the vertex set V and edge set $E \subseteq V \times V$ with $(u,v) \in E$ if and only if either $s_0(u) = v$ or $s_1(u) = v$. The *desperation* of a hopeful vertex v is the length of the shortest path from v to t.

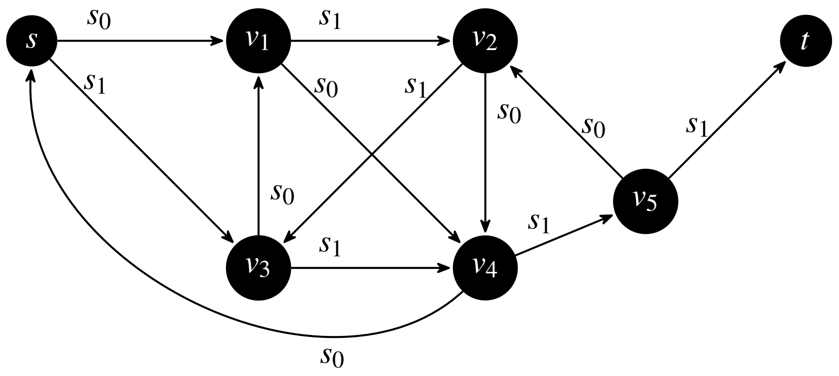

Figure 4.1: The goal of the arrival problem is to decide whether a partiular walk on a directed graph with a particular structure reaches the target. On successive visits to a particular vertex the outgoing edge taken alternates. In this example the walk begins $s \to v_1 \to v_4 \to s \to v_3 \to \ldots$.

Lemma 4.1.5 ((?)). *Let* (V,s,t,s_0,s_1) *be an instance of the* ARRIVAL *problem. If* $v \in V$ *is hopeful, the arrival walk passes through* v *at most* $2^{|V|}$ *times.*

Proof. Begin by noting that if a vertex is hopeful, it's desperation is at most $|V|$. I perform an induction on the desperation of v. Suppose the desperation of $v \in V$ is 1. Then either $s_0(v) = t$ or $s_1(v) = t$. If $s_0(v) = t$, t will be reached after passing through v once. If $s_1(v) = t$ and $s_0(v) \neq t$ t will be reached after passing through v twice. In both cases v is passed through at most $2^1 = 2$ times.
Suppose that all hopeful vertices with desperation $d-1$ are passed through at most 2^{d-1} times. Then if $v \in V$ is hopeful with desperation d, for some hopeful $w \in V$ with desperation $d-1$ either $s_0(v) = w$ or $s_1(v) = w$. So at least every second passing of v, the walk will proceed to w. But the walk can pass through w at most 2^{d-1} times, so the walk can pass through v at most $2 \cdot 2^{d-1} = 2^d$ times. ■

Corollary 4.1.6. *Let* (V,s,t,s_0,s_1) *be an instance of the* ARRIVAL *problem. Then the arrival walk either reaches t, or reaches a vertex which is not hopeful.*

From **??** it is clear that deciding an instance of the ARRIVAL problem is equivalent to deciding whether or not the arrival walk reaches t or reaches a vertex which is not hopeful.

Definition 4.1.7 (Processed Arrival). Let (V,s,t,s_0,s_1) be an instance of the arrival problem. Let $\sim$ be the equivalence relation on V generated by $u \sim v$ if u and v are both not hopeful. The *processed arrival problem* is a set of vertices $V' = V/\sim$, the canonical projections of s,t,s_0,s_1 into V', and a choice of representative $\bar{t} \in V'$ of all the non hopeful vertices in V.

Noting that the set of non hopeful vertices can be easily computed in linear time with a breadth first search from t, from this point on I will refer to instances of the ARRIVAL problem exclusively as tuples $(V,s,t,\bar{t},s_0,s_1)$ constructed as above.

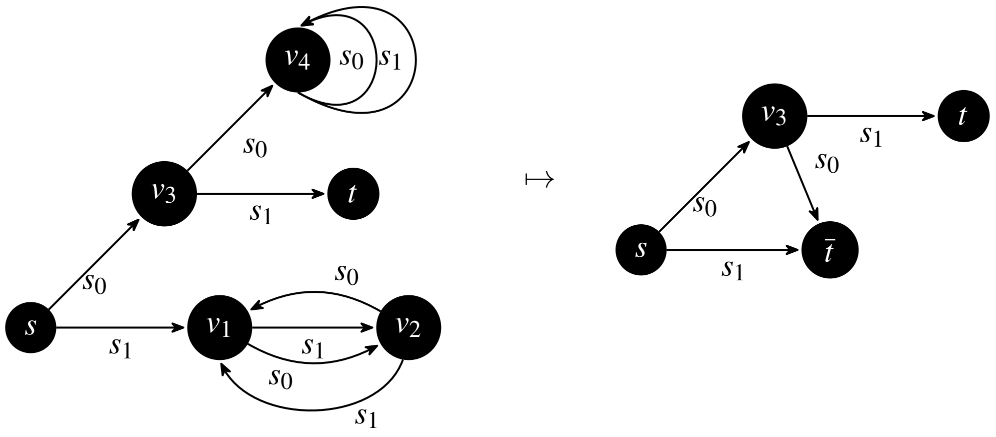

Figure 4.2: Instances of the arrival problem can be preprocessed so that every non-target vertex has a directed path to the target. An extra 'bad target' $\bar{t}$ is added which represents all the vertices with no directed path to the target t, and the problem becomes to decide which of the two targets is reached. From **??** the walk in the resultant graph must be finite.

Corollary 4.1.8 ((**?**)). *The time complexity of the* Arrival *problem is* $O(n \cdot 2^n)$.

Proof. I reason that the arrival walk on the processed instance $(V, s, t, \bar{t}, s_0, s_1)$ has it's walk length bounded by $O(n \cdot 2^n)$. Every vertex $v \in V$ with $v \neq \bar{t}$ is hopeful with desperation at most n, so by **??** can be passed through at most 2^n times. If the walk reaches t or $\bar{t}$ it terminates, and there are at most n vertices $w \in V$ such that $w \notin \{t, \bar{t}\}$, so the walk can take at most $n \cdot 2^n$ steps. ■

There are in fact instances of the Arrival problem with exponentially long walks - as seen in **??** - implying that the worst-case runtime of this algorithm is exponential. Recently a sub-exponential[1] upper bound for Arrival was given in (**?**). Interestingly, their algorithm involves a reduction from Arrival to Tarski. I will not detail the reduction used in the sub-exponential algorithm, but will spend the remainder of the section demonstrating a similar, yet simpler reduction from Arrival to Tarski.

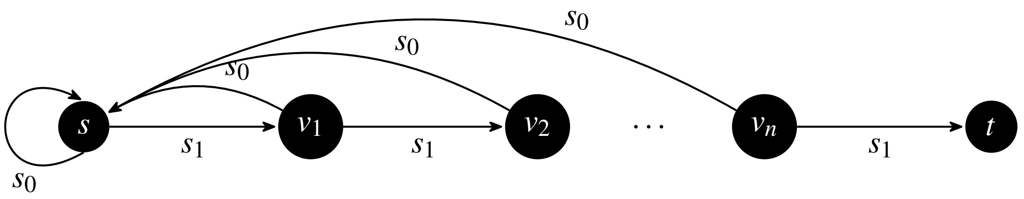

Figure 4.3: There are instances of the arrival problem with exponentially long walks. An induction on the number of steps to get from $s \to v_i$ shows that walk on the above takes at least $\Omega(2^n)$ steps to reach t.

[1]Specifically an algorithm running in time $O(2^{\sqrt{n}})$.

Definition 4.1.9 (Switching Flow)**.** Let $(V, s, t, \bar{t}, s_0, s_1)$ be an arrival graph. A *switching flow* is a pair of maps $f_0, f_1 : V \setminus \{t, \bar{t}\} \to \mathbb{Z}_{\geq 0}$ such that the following axioms hold. Let $f_{\text{in}}(v) = \sum_{\substack{w \in V \\ s_0(w) = v}} f_0(w) + \sum_{\substack{w \in V \\ s_1(w) = v}} f_1(w)$.

- For all $v \in V \setminus \{s, t, \bar{t}\}$, $f_{\text{in}}(v) = f_0(v) + f_1(v)$ (flow conservation),
- $f_{\text{in}}(s) = f_0(s) + f_1(s) - 1$ (source flow conservation),
- For all $v \in V$, $f_1(v) \leq f_0(v) \leq f_1(v) + 1$ (switching).

It was observed in (**?**) that the walk on an arrival graph can be characterized by a switching flow.

Lemma 4.1.10 ((**?**))**.** *Let $(V, s, t, \bar{t}, s_0, s_1)$ be an instance of the* Arrival *problem. Define $f_0 : V \setminus \{t, \bar{t}\} \to \mathbb{Z}_{\geq 0}$ by $f_0(v) =$ the number of times $s_0(v)$ is traversed in the arrival walk, and define f_1 similarly. Then (f_0, f_1) is a switching flow.*

Proof. Flow conservation and source flow conservation follow from the fact that the walk must walk out of a vertex if it walks in, minus the initial step it takes from the source. Switching follows from the nature of the walk taking the s_0 edge on even passes, and s_1 edge on odd passes. ∎

I next establish a correspondence from arrival instances to monotone functions.

Definition 4.1.11 (Arrival Monotone Function)**.** Let $(V, s, t, \bar{t}, s_0, s_1)$ be an instance of the arrival problem, $d = |V \setminus \{t, \bar{t}\}|$ and $(v_i)_{i \in [d]}$ be an enumeration of the vertices in $V \setminus \{t, \bar{t}\}$. The *arrival monotone function* is a function $f : \mathbb{Z}^d_{\geq 0} \to \mathbb{Z}^d_{\geq 0}$ defined coordinatewise as,

$$f((a_1, \ldots, a_d))_i = \begin{cases} \sum_{\substack{j \in [d] \\ s_0(v_j) = v_i}} \lceil \frac{a_j}{2} \rceil + \sum_{\substack{j \in [d] \\ s_1(v_j) = v_i}} \lfloor \frac{a_j}{2} \rfloor & v_i \neq s \\ 1 + \sum_{\substack{j \in [d] \\ s_0(v_j) = v_i}} \lceil \frac{a_j}{2} \rceil + \sum_{\substack{j \in [d] \\ s_1(v_j) = v_i}} \lfloor \frac{a_j}{2} \rfloor & v_i = s. \end{cases}$$

Lemma 4.1.12. *The arrival monotone function is monotone.*

Proof. Clearly the sum of monotone functions is also monotone, $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ are monotone, composition of monotone functions is monotone, linear functions with non-negative coefficients are monotone, and constant functions are monotone. This encompasses all components of the above function, which is therefore montone. ∎

The monotone function was constructed precisely so that the following proposition holds.

Proposition 4.1.13. *Let $f : \mathbb{Z}^d_{\geq 0} \to \mathbb{Z}^d_{\geq 0}$ be an arrival monotone function, and $a = (a_1, \ldots, a_d) \in \mathbb{Z}^d_{\geq 0}$. Define $g_0(v_i) = \lceil \frac{a_i}{2} \rceil$ and $g_1(v_i) = \lfloor \frac{a_i}{2} \rfloor$. Then (g_0, g_1) is a switching flow if and only if $f(a) = a$.*

Proof. ($\Longleftarrow$) For flow conservation, let $v_i \in V \setminus \{s,t,\bar{t}\}$. Then,

$$\begin{aligned}\sum_{\substack{j\in[d]\\ s_0(v_j)=v_i}} g_0(v_j) + \sum_{\substack{j\in[d]\\ s_0(v_j)=v_i}} g_1(v_j) &= \sum_{\substack{j\in[d]\\ s_0(v_j)=v_i}} \left\lceil \frac{a_j}{2} \right\rceil + \sum_{\substack{j\in[d]\\ s_0(v_j)=v_i}} \left\lfloor \frac{a_j}{2} \right\rfloor \\ &= f(a)_i \\ &= a_i \\ &= \left\lceil \frac{a_i}{2} \right\rceil + \left\lfloor \frac{a_i}{2} \right\rfloor \\ &= g_0(v_i) + g_1(v_1).\end{aligned}$$

Source flow conservation follows similarly. Switching is clear from the definition of $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$.
($\Longrightarrow$) Let $g_0(v_i) = \lceil \frac{a_i}{2} \rceil$, $g_1(v_i) = \lfloor \frac{a_i}{2} \rfloor$, and (g_0, g_1) be a switching flow. Then for each $i \in [d]$ if $v_i \neq s$,

$$\begin{aligned}f((a_1,\ldots,a_d))_i &= \sum_{\substack{j\in[d]\\ s_0(v_j)=v_i}} \left\lceil \frac{a_j}{2} \right\rceil + \sum_{\substack{j\in[d]\\ s_1(v_j)=v_i}} \left\lfloor \frac{a_j}{2} \right\rfloor \\ &= \sum_{\substack{j\in[d]\\ s_0(v_j)=v_i}} g_0(v_j) + \sum_{\substack{j\in[d]\\ s_1(v_j)=v_i}} g_1(v_j) \\ &= g_0(v_i) + g_1(v_i) \qquad \text{(by flow conservation)} \\ &= \left\lceil \frac{a_i}{2} \right\rceil + \left\lfloor \frac{a_i}{2} \right\rfloor = a_i,\end{aligned}$$

and if $v_i = s$,

$$\begin{aligned}f((a_1,\ldots,a_d))_i &= 1 + \sum_{\substack{j\in[d]\\ s_0(v_j)=v_i}} \left\lceil \frac{a_j}{2} \right\rceil + \sum_{\substack{j\in[d]\\ s_1(v_j)=v_i}} \left\lfloor \frac{a_j}{2} \right\rfloor \\ &= 1 + \sum_{\substack{j\in[d]\\ s_0(v_j)=v_i}} g_0(v_j) + \sum_{\substack{j\in[d]\\ s_1(v_j)=v_i}} g_1(v_j) \\ &= 1 + g_0(v_i) + g_1(v_i) - 1 \qquad \text{(by source flow conservation)} \\ &= \left\lceil \frac{a_i}{2} \right\rceil + \left\lfloor \frac{a_i}{2} \right\rfloor = a_i.\end{aligned}$$

■

The next proposition draws an intriguing connection between the arrival walk and monotone function.

Proposition 4.1.14. *Let $(V,s,t,\bar{t},s_0,s_1)$ be an instance of the problem, and f be the arrival monotone function. For $a \in \mathbb{Z}^d_{\geq 0}$ let $g_0(a) = \lceil \frac{a}{2} \rceil$ and $g_1(a) = \lfloor \frac{a}{2} \rfloor$. For $i \in \{x \in \mathbb{Z}_{\geq 0} | x < n\}$ where n number of steps to reach t or $\bar{t}$, let $(h^i_0 : [d] \to \mathbb{Z}_{\geq 0}, h_1 a^i : [d] \to \mathbb{Z}_{\geq 0})_{i \in \mathbb{Z}_{\geq 0}}$ be a sequence defined by $h^i_0(j) =$ the number of times the s_0 edge has been taken from v_j after i steps in the walk, and define $h^i_1(j)$ similarly for the s_1 edge. Then for each $j \in [d]$ $g_0(f^i(\vec{0}))_j = h^i_0(j)$ and $g_1(f^i(\vec{0}))_j = h^i_1(j)$.*

Proof. By induction. For the base case where $i = 0$ no edges have been crossed, so for each $j \in [d]$ $h_0^i(j) = h_1^i(j) = 0 = \frac{f^0(\vec{0})_i}{2} = 0$.
So suppose for some $i \in \mathbb{Z}_{\geq 0}$ the statement is true for all $j \in [i-1]$. Let $v_k \in V \setminus \{t, \bar{t}\}$ be the vertex the walk is at after $i-1$ steps, and v_l after $i-2$ steps. Then by the inductive hypothesis $f^{i-2}(\vec{0})_l + 1 = f^{i-1}(\vec{0})_l$ and for each $m \in [d] \setminus \{l\}$, $f^{i-2}(\vec{0})_m = f^{i-1}(\vec{0})_m$. It follows that $f^{i-1}(\vec{0})_k + 1 = f^i(\vec{0})_k$, and the switching property of the monotone function guarantees that the extra unit appears on the correct edge. ■

All of that is really just to say that in the case of monotone functions from the arrival problem, *iteration from the bottom of the lattice as in* **??** *is the walk.* This connection gives me some useful corollaries.

Corollary 4.1.15. *Let f be an arrival monotone function, (g_0, g_1) be the switching flow corresponding to an arrival walk as in* **??***, and $a \in \mathbb{Z}^d_{\geq 0}$ be the fixpoint corresponding to this switching flow as in* **??***. Then a is the least fixpoint of f.*

Proof. Using a similar argument to **??**, iteration from the bottom of the lattice gives the least fixpoint. But **??** says that iteration from the bottom of the lattice is the walk and will give the fixpoint corresponding to the switching flow of the walk. ■

Corollary 4.1.16. *Let $f : \mathbb{Z}^d_{\geq 0} \to \mathbb{Z}^d_{\geq 0}$ be an arrival monotone function, and $a \in \mathbb{Z}^d_{\geq 0}$ be a point such that $f(a) = a$. If $f_t = \sum_{\substack{i \in [d] \\ s_0(v_i) = t}} \lceil \frac{a_i}{2} \rceil + \sum_{\substack{i \in [d] \\ s_1(v_i) = t}} \lfloor \frac{a_i}{2} \rfloor$ and $f_{\bar{t}} = \sum_{\substack{i \in [d] \\ s_0(v_i) = \bar{t}}} \lceil \frac{a_i}{2} \rceil + \sum_{\substack{i \in [d] \\ s_1(v_i) = \bar{t}}} \lfloor \frac{a_i}{2} \rfloor$ are the inflows at t and $\bar{t}$ respectively. Exactly one of the following hold,*

- *$f_t = 1$ and $f_{\bar{t}} = 0$,*
- *$f_t = 0$ and $f_{\bar{t}} = 1$.*

Proof. The walk only terminates when it reaches either t or $\bar{t}$, so the switching flow corresponding to the walk must satisfy exactly one of $f_t = 1$ and $f_{\bar{t}} = 0$ or $f_t = 0$ and $f_{\bar{t}} = 0$. By **??** the walk corresponds to the least fixpoint, and for all other switching flows (g_0, g_1), at least one of $g_t \geq 1$ or $g_{\bar{t}} \geq 1$. Suppose for a contradiction that $g_t + g_{\bar{t}} \geq 2$ and let $a \in \mathbb{Z}^d_{\geq 0}$ be the point corresponding to (g_0, g_1). Then $\sum_{i \in [d]} f(a)_i \leq \sum_{i \in [d]} a_i - 1$. But this contradicts a being a fixpoint due to **??**, from which I find $g_t + g_{\bar{t}} = 1$, and the claim follows. ■

Remarkably, this implies that *any* switching flow is certificate to the walk reaching either t or $\bar{t}$. Further, fixpoints are switching flows so deciding the ARRIVAL problem is reducible to finding a fixpoint of a particular monotone function! I'm not quite finished yet however; the monotone functions of arrival instances considered so far have been on the infinite lattice $\mathbb{Z}^d_{\geq 0}$, but the TARSKI problem I defined is on the finite lattice $[N]^d$. This will turn out to not be an issue.

Notation 4.1.17. For $N \in \mathbb{Z}_{\geq 0}$ notation $[N]_0$ represents $[N] \cup \{0\}$.

Definition 4.1.18 (Bounded arrival monotone function)**.** Let $(V, s, t, \bar{t}, s_0, s_1)$ be an instance of the arrival problem and f be it's corresponding arrival function. Let $n = |V|$

and $N = 2^n$. The *bounded arrival monotone function* is a function $F : [N]_0^d \to [N]_0^d$ defined coordinatewise as $F(a)_i = \min(f(a)_i, N)$.

Lemma 4.1.19. *Let F be a bounded arrival monotone function. Then F is monotone.*

Proof. Similarly to the proof of **??**, $\min(\cdot, N)$ is clearly monotone. Monotonicity then follows monotonicity of the arrival monotone function, and monotonicity being preserved under composition. ■

Lemma 4.1.20. *Let $f : \mathbb{Z}^d_{\geq 0} \to \mathbb{Z}^d_{\geq 0}$ be a monotone arrival function. If $a \in \mathbb{Z}^d_{\geq 0}$ then $\sum_{i\in[d]} f(a)_i \leq 1 + \sum_{i\in[d]} a_i$.*

Proof. I have

$$\begin{aligned}\sum_{i\in[d]} f_i((a_1,\ldots,a_d)) &= \sum_{i\in[d]} \begin{cases} \sum_{\substack{j\in[d] \\ s_0(v_j)=v_i}} \left\lceil \frac{a_j}{2} \right\rceil + \sum_{\substack{j\in[d] \\ s_1(v_j)=v_i}} \left\lfloor \frac{a_j}{2} \right\rfloor & v_i \neq s \\ 1 + \sum_{\substack{j\in[d] \\ s_0(v_j)=v_i}} \left\lceil \frac{a_j}{2} \right\rceil + \sum_{\substack{j\in[d] \\ s_1(v_j)=v_i}} \left\lfloor \frac{a_j}{2} \right\rfloor & v_i = s \end{cases} \\ &= 1 + \sum_{i\in[d]} \sum_{\substack{j\in[d] \\ s_0(v_j)=v_i}} \left\lceil \frac{a_j}{2} \right\rceil + \sum_{i\in[d]} \sum_{\substack{j\in[d] \\ s_1(v_j)=v_i}} \left\lfloor \frac{a_j}{2} \right\rfloor.\end{aligned}$$

By construction of s_0 and s_1, for each $j \in [d]$ there is at most one $i \in [d]$ such that $s_0(v_j) = v_i$ and at most one $i' \in [d]$ such that $s_1(v_j) = v_{i'}$. So,

$$\begin{aligned}1 + \sum_{i\in[d]} \sum_{\substack{j\in[d] \\ s_0(v_j)=v_i}} \left\lceil \frac{a_j}{2} \right\rceil + \sum_{i\in[d]} \sum_{\substack{j\in[d] \\ s_1(v_j)=v_i}} \left\lfloor \frac{a_j}{2} \right\rfloor &\leq 1 + \sum_{j\in[d]} \left\lceil \frac{a_j}{2} \right\rceil + \sum_{j\in[d]} \left\lfloor \frac{a_j}{2} \right\rfloor \\ &= 1 + \sum_{i\in[d]} a_i.\end{aligned}$$

■

Lemma 4.1.21 ((?)). *Let $(V, s, t, \bar{t}, s_0, s_1)$ be an instance of the arrival problem, $n = |V|$, $N = 2^n$, $d = |V \setminus \{t, \bar{t}\}|$, $f : \mathbb{Z}^d_{\geq 0} \to \mathbb{Z}^d_{\geq 0}$ be the arrival monotone function, and $F : [N]_0^d \to [N]_0^d$ be the bounded arrival monotone function. If $a \in [N]_0^d$ satisfies $F(a) = a$, then $f(a) = a$.*

Proof. Begin by noting that for all $a \in [N]_0^d$, $f(a) \geq F(a)$. Suppose for a contradiction that $f(a) \neq F(a)$. Then $F(a) > f(a)$. From **??** combined with the fact that $F(a) = a$, I find that $\sum_{i\in[d]} f(a)_i = 1 + \sum_{i\in[d]} a_i$. It must therefore be the case that for some $i \in [d]$, $f(a)_j = \begin{cases} a_j + 1 & j = i \\ a_j & j \neq i. \end{cases}$ By definition $F(a)_i = \min(f(a)_i, N)$ from which it follows $a_i = N$ and $f(a)_i = N + 1$. For $\sum_{i\in[d]} f(a)_i = 1 + \sum_{i\in[d]} a_i$ I require that $f_{\text{in}}(t) = 0$ and $f_{\text{in}}(\bar{t}) = 0$. But **??** combined with $a_i = N = 2^n$ implies that the walk must have terminated. That is, either $f_{\text{in}}(t) > 0$ or $f_{\text{in}}(\bar{t}) > 0$, which is a contradiction. ■

Theorem 4.1.22. ARRIVAL *is polynomial time reducible to* TARSKI.

4.2 Simple Stochastic Games

Simple stochastic games, as defined in (**?**), are a class of zero-sum games played on graphs with two players called the maximizer and minimizer respectively. For the purposes of this dissertation I will consider only β-stopping simple stochastic games (the meaning of which will be defined in turn). Condon shows in (**?**) that these games necessarily have a rational value for rationally described instances of the problem, and further that the value can be achieved in pure stationary strategies (all of these concepts will be made precise). The relationship to TARSKI then comes from (**?**), where Etessami et al. show that computing the *exact* value of (not necessarily β-stopping) simple stochastic games as well as a pure stationary strategy profile to achieve this value is polynomial-time reducible to TARSKI. This section will lay out the required definitions, and describe the aforementioned reduction to TARSKI in the special case of β-stopping.

Definition 4.2.1 (Simple Stochastic Game)**.** A *simple stochastic game* is a directed graph $G = (V, V_p, V_{\max}, V_{\min}, E, v_0, t, \beta)$ with designated start vertex $v_0 \in V$, target vertex $t \in V$, $\beta \in (0,1] \cap \mathbb{Q}$, a partition of $V \setminus \{t\}$ into three disjoint subsets $V_p, V_{\min}, V_{\max}$, and a mapping $p : V_p \times V \to [0,1]$ such that for all $v_p \in V_p$, $v \in V$ if $(v_p, v) \notin E$ I have $p(v_p, v) = 0$, for each $v_p \in V_p$ $\sum_{v \in V} p(v_p, v) = 1$, and for every $v \in V \setminus \{t\}$ there is necessarily an edge $(v, w) \in E$ for some $w \in V$. A *play* in a simple stochastic game transpires as follows. A token is placed on the initial vertex of the game v_0. Let v_i be the vertex on which the token currently lies. Then at each step, the game halts with probability β. If it did not halt and $v_i \in V_{\max}$ ($V_{\min}$) then the maximizer (minimzer) chooses and edge $(v_i, v_{i+1}) \in E$ for some $v_{i+1} \in V$. If $v_i \in V_p$ then an edge $(v_i, w) \in E$ is chosen randomly with probability $p(v_i, w)$. If $v_i = t$ then the game halts. The payoffs of to the maximizer is 1 if the game reaches t, and 0 otherwise. The minimizer achieves the negative payoff of the maximizer.

It is clear that since $\beta > 0$ the game eventually halts with probability 1.

Definition 4.2.2 (Pure Stationary Strategy)**.** Let $G = (V, V_p, V_{\max}, V_{\min}, E, v_0, t)$ be a simple stochastic game. A *pure stationary strategy* for the maximizer is a mapping $\sigma : V_{\max} \to V$ with the requirement that for all $v \in V_{\max}$ $(v, \sigma(v)) \in E$. The set of all such pure stationary strategies for the maximizer is denoted S. A pure stationary strategy for the minimizer is a map $\tau : V_{\min} \to V$ such that for all $v \in V_{\min}$ $(v, \tau(v)) \in E$. The set of all such pure stationary strategies for the minimizer is denoted T. A *pure stationary strategy profile* is a pair $(\sigma, \tau) \in S \times T$.

Theorem 4.2.3 ((**?**), lemma 6)**.** *Let G be a β-stopping simple stochastic game. Then* $\max_{\sigma \in S} \min_{\tau \in T}$

4.3 Shapley's Stochastic Games

Chapter 5

Testing the Algorithms

5.1 Method

Chapter 6

Your next chapter

A dissertation usually contains several chapters.

Chapter 7

Conclusions

7.1 Final Reminder

The body of your dissertation, before the references and any appendices, *must* finish by page 40. The introduction, after preliminary material, should have started on page 1.

You may not change the dissertation format (e.g., reduce the font size, change the margins, or reduce the line spacing from the default single spacing). Be careful if you copy-paste packages into your document preamble from elsewhere. Some LaTeX packages, such as `fullpage` or `savetrees`, change the margins of your document. Do not include them!

Over-length or incorrectly-formatted dissertations will not be accepted and you would have to modify your dissertation and resubmit. You cannot assume we will check your submission before the final deadline and if it requires resubmission after the deadline to conform to the page and style requirements you will be subject to the usual late penalties based on your final submission time.

Appendix A

First appendix

A.1 First section

Any appendices, including any required ethics information, should be included after the references.

Markers do not have to consider appendices. Make sure that your contributions are made clear in the main body of the dissertation (within the page limit).

Appendix B

Participants' information sheet

If you had human participants, include key information that they were given in an appendix, and point to it from the ethics declaration.

Appendix C

Participants' consent form

If you had human participants, include information about how consent was gathered in an appendix, and point to it from the ethics declaration. This information is often a copy of a consent form.