Analysis of Vector Addition Systems

Internship Report

of

Akkapaka Saikiran

Indian Institute of Technology, Bombay

{a.saikiran}@iitb.ac.in

Under the supervision of

Prof. Alain Finkel

LSV, ENS Paris-Saclay, CNRS, Université Paris-Saclay, France

{finkel}@lsv.fr

in the summer of 2020

Contents

1 Introduction

In the summer of 2020, I undertook a virtual research internship at LSV (Laboratoire Spécification et Vérification), ENS Paris-Saclay under Prof Alain Finkel. I worked on Semi-linearity of Projections of Reachability Sets of Vector Addition Systems. The idea of the internship was to explore the work done in this field so far, and the end-goal was to devise an algorithm to construct the aforementioned semi-linear sets, which is very non-trivial since it reduces to the reachability problem. It wasn't met, but I analyzed the reachability sets of Vector Addition Systems by building an understanding of Karp-Miller Graph defined rather informally by Büning et al in [1]. I also explored and elucidated certain properties of the same by writing definitions and proofs in an article, collaborating with Prof Finkel.

The main objective of this report is to provide an overview of the same. In section 2 I describe the events surrounding the internship which contributed to the way things turned out. In section 3 I describe the work and much of this section is drawn from the article I was writing, which is turn is primarily based on [1]. In section 4 I look back and describe the shortcomings faced and the lessons learnt in this virtual research internship.

2 Before the Internship

2.1 The Start

The way I obtained this internship was rather unconventional. In the start of December 2019 I met Prof Akshay, a professor at CSE IITB, to seek guidance for an internship interview, where I'd been asked to read a research paper in his domain. I did not get through that interview so a few days later I met Prof Akshay to seek guidance, yet again. It was then that I was introduced to Prof Alain Finkel, who was visiting IITB at the time. After a chat he asked me to send him my CV as he may have some ideas for a research internship in the summer of 2020.

Fast-forward a few weeks, and I had an internship offer, but not on Vector Addition Systems. The topic then was using Machine Learning in verification of Petri Nets. Coursework of my fourth semester kept me busy for the next two months, but I studied some Machine Learning by sitting through an extra course.

2.2 The Pandemic

In the start of March fears about COVID-19 possibly playing truant to international travel surfaced. Within a couple of weeks there was pandemonium and IITB shut itself down, sending (almost) all of its students home, and lockdowns

were implemented all over India. France was one of the worse hit countries back then so that slowly ruled out the possibility of traveling to ENS Paris-Saclay, something I was really looking forward to. Nevertheless I braced myself for a virtual internship, which was to start when the semester got over. That never happened though, as in the end of March, IITB declared April and May as summer holidays. I got in touch with Prof Finkel and began reading relevant papers to get started with the internship.

2.3 The First Internship Topic

There are two fast but complementary algorithms to find the Minimal Coverabity set for Petri Nets. One is a backward algorithm called QCover [2], while the other is a recently-corrected forward algorithm called MinCov [3]. These algorithms are complementary in the sense that while MinCov is faster on “unsafe" instances, QCover is faster on “safe" instances. It was found that running the algorithms in parallel improved results on benchmarks. So a team of four, including me, assembled with the goal of figuring out how to *learn* which algorithm was likely to do better on a particular input.

Most of April was spent in learning the relevant machine learning and reading the papers related to the two algorithms. This was when the virtual nature of the job was a big bottleneck. Everyone in the team except me was busy with multiple projects and as this was a relatively unexplored field, we couldn’t give enough time to it to get any insight. Moreover, the inadequate background knowledge and the virtual mode of communication hampered my ability to work.

So towards the end of April Prof Finkel suggested it would be a better idea to abandon this topic, and he promised to come up with a more theoretical idea for an internship within a day, one which wouldn’t need collaboration of many people, it would be just me and him. I was disappointed, having spent a month on the topic, but it was a good idea in retrospect. The Machine Learning I learnt in April helped me for a course project I did a couple of months later, and the papers I read provided a solid foundation for many of the concepts involved in my new internship topic.

3 The Internship Work

3.1 Introduction

A Petri Net is a graphical and mathematical tool used in many different science domains, primarily for modeling of concurrent systems. A Vector Addition System (VAS) is a mathematical model equivalent to Petri Nets, which was introduced by

Karp and Miller in [4] and later generalised to Vector Addition System with States (VASS) by Hopcroft and Pansiot in [5].

Informally, a VAS can be described using a starting vector and a set of transition vectors. The initial vector can be seen as values given to multiple counters, and transition vectors can be seen as updates to these counters, with the restriction that the counter values never drop below zero. Given any initial vector with all components non-negative, a sequence of transition vectors can be applied to it, by performing component-wise addition, as long as every intermediate vector has all components non-negative.

3.2 Mathematical Preliminaries

The sets of integers and non-negative integers are denoted as $\mathbb{Z}$ and $\mathbb{N}$. Define $[\,n\,]$ as the set $\{x \in \mathbb{N} \mid 1 \leq x \leq n\}$ for every $n \in \mathbb{N}$. Vectors are denoted in bold and the i-th component of a vector $\mathbf{v}$ is denoted as $\mathbf{v}[i]$. The $\leq$ relation of $\mathbb{N}$ and $\mathbb{Z}$ is extended to $\mathbb{N}^d$ and $\mathbb{Z}^d$ as $\mathbf{u} \leq \mathbf{v}$ iff $\mathbf{u}[i] \leq \mathbf{v}[i]$ for all $i \in [\,d\,]$. Also, $\mathbf{u} < \mathbf{v}$ iff $\mathbf{u} \leq \mathbf{v}$ and $\mathbf{u} \neq \mathbf{v}$.

$\mathbb{N}$ is extended to $\mathbb{N}_\omega$ using a special symbol ω which follows $n < \omega$ and $\omega + n = n + \omega = \omega$ for all $n \in \mathbb{N}$. The intuitive meaning of ω is that it signifies an arbitrarily large quantity. We similarly extend $\mathbb{N}^d$ to $\mathbb{N}_\omega^d$. For a vector $\mathbf{x} \in \mathbb{N}_\omega^d$ define $\Omega(\mathbf{x})$ as $\{i \in [\,d\,] \mid \mathbf{x}[i] = \omega\}$, i.e. the set of components of $\mathbf{x}$ occupied by ω.

We can now formally define a Vector Addition System.

Syntax: A VAS of dimension d is defined as $\mathcal{V} = (\mathbf{x}_0, T)$, where $\mathbf{x}_0 \in \mathbb{N}^d$ is called the *initial marking* and $T = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n\} \subseteq \mathbb{Z}^d$ is a finite set of *transitions* or *transition vectors*.

Semantics: A *transition sequence* $\sigma = \mathbf{u}_1\mathbf{u}_2 \ldots \mathbf{u}_k$ with $\mathbf{u}_i \in T$ is said to be *fireable from* a marking $\mathbf{y}$ if $\mathbf{y} + \sum_{j=1}^{i} \mathbf{u}_j \geq \mathbf{0}$ for all $i \in [\,k\,]$. When σ is fireable from $\mathbf{y}$, we write $\mathbf{y} \xrightarrow{\sigma} \mathbf{z}$ with $\mathbf{z} = \mathbf{y} + \sum_{j=1}^{k} \mathbf{u}_j$ and $\mathbf{z}$ is said to be *reachable* from $\mathbf{y}$. $\boldsymbol{\Delta}(\sigma)$ denotes the vector sum of the transitions in σ. Formally, $\boldsymbol{\Delta}(\sigma) = \sum_{i=1}^{k} \mathbf{u}_i$. The *reachability set* of a VAS $\mathcal{V} = (\mathbf{x}_0, T)$ is thus defined as $\mathcal{R}_\mathcal{V} = \{\mathbf{z} \in \mathbb{N}^d \mid \exists \sigma \in T^* \ \mathbf{x}_0 \xrightarrow{\sigma} \mathbf{z}\}$. Reachability can be shown graphically as follows.

$$\mathbf{y} = \mathbf{y}_0 \xrightarrow{\mathbf{u}_1} \mathbf{y}_1 \xrightarrow{\mathbf{u}_2} \ldots \xrightarrow{\mathbf{u}_3} \mathbf{y}_k = \mathbf{z} \quad \text{such that} \quad \mathbf{y}_i = \mathbf{y} + \boldsymbol{\Delta}(\mathbf{u}_1\mathbf{u}_2 \ldots \mathbf{u}_i)$$

We state without proof a well-known lemma from combinatorics known as Dickson's lemma.

Lemma 1. *Let $\mathbf{x}_1, \mathbf{x}_2, \ldots$ be an infinite sequence of vectors from $\mathbb{N}_\omega^d$. Then there exist two indices i and j with $i < j$ such that $\mathbf{x}_i \leq \mathbf{x}_j$.*

3.3 Relevant Graph Theory

A *graph* G is a pair (P, E) where P is a set of *nodes* (or places) and $E \subseteq P \times P$ is a set of *edges*. If $(p, q) \in E$ we write $p \to_G q$ to denote an edge between p and q, and when G is implicit we simply write $\to$. All graphs we discuss are directed, unless stated otherwise.

A graph is said to be infinite iff P is infinite. A graph may also permit a labeling over its nodes and/or edges. A graph $G' = (P', E')$ is said to be a *subgraph* of a graph $G = (P, E)$ if $P' \subseteq P$ and $E' \subseteq (P' \times P' \cap E)$.

Let G be a graph and p, q two nodes in G.

- A *path* from a node p to a node q of length n is a sequence of nodes $p_0, p_1, \ldots, p_n$ such that $p = p_0$ and $q = p_n$ and $(p_{i-1}, p_i) \in E$ for every $i \in [\, n \,]$.
- A *elementary path* in G is a path $p_0, p_1, \ldots, p_n$ such that all nodes are pairwise distinct.
- A *cycle* in G is a path $p_0, p_1, \ldots, p_n$ such that $p_0 = p_n$, and all other nodes are pairwise distinct.
- We say q is *reachable* from p if there exists a path from p to q, and denote it as $p \xrightarrow{*} q$. We then naturally define $\mathcal{R}(p) = \{\, q \in P \mid p \xrightarrow{*} q \,\}$ as the set of nodes reachable from a node p.
- We say a graph is *rooted* if there is a node $r \in P$ such that every node is reachable from it, i.e. $\mathcal{R}(r) = P$. The node r is called a root of the graph.
- A *rooted tree* is a rooted graph with exactly one root such that the root has no incoming edges, and every other edge has exactly one.
- A rooted tree T is said to be a *spanning tree* of a rooted graph $G = (P, E)$ iff $T = (P, E')$ such that $E' \subseteq E$, i.e. T contains all nodes and some edges of G.
- We can *induce* a rooted spanning tree T from a rooted graph $G = (P, E)$ by performing a breadth-first graph traversal starting at the root and including only edges which lead to nodes not visited yet. There may be more than one rooted spanning trees for a rooted graph.
- We define $\mathcal{S}(p) = \{\, q \in P \mid (p, q) \in E \,\}$ as the *successor* set of the node p, i.e. the set of nodes reachable from p via paths of length 1.
- A graph is said to be *finitely branching* if $\mathcal{S}(p)$ is finite for every $p \in P$.

We are now in a position to state and prove a lemma which is a variant of the well-known König's lemma.

Lemma 2. *Every finitely branching infinite rooted graph contains an infinite elementary path.*

Proof. Let $G = (P, E)$ be a finitely branching infinite rooted graph along with a root $r \in P$. Consider a spanning rooted tree induced from G and call it T. This

can be constructed even though G is infinite because it is finitely branching. We will construct, step by step, an infinite path $p_0, p_1, \ldots, p_n, \ldots$ in T with $p_0 = r$. Observe that $p_i \neq p_j$ for all $i \neq j$ because there are no cycles in T so every path is an elementary path. Any path of T also exists in G as T is but a subgraph of G. Hereafter, $\mathcal{R}$ and $\mathcal{S}$ refer to $\mathcal{R}_T$ and $\mathcal{S}_T$ respectively.

Since G is finitely branching, T is finitely branching, so $\mathcal{S}(r)$ is finite. The set of nodes P is infinite, and $P = \bigcup_{p \in \mathcal{S}(r)} \mathcal{R}(p)$, so there must exist a node $p_1 \in \mathcal{S}(r)$ such that $p_1 \neq r$ and $\mathcal{R}(p_1)$ is infinite because a finite union of finite sets cannot be infinite.

Now suppose (induction hypothesis) that for $k \geq 1$, we have constructed a path $r, p_1, \ldots, p_k$ such that $\mathcal{R}_G(p_k)$ is infinite. Again, by a similar argument there exists some node $p_{k+1} \in \mathcal{S}(p_k)$ such that $\mathcal{R}(p_{k+1})$ is infinite and $p_{k+1} \neq p_k$. We append this to our path. Now by using the axiom of choice, we construct the infinite path $r, p_1, \ldots, p_k, \ldots$. Thus we show that if a finitely branching rooted graph is infinite it has an infinite elementary path.

■

3.4 A Karp-Miller Graph and Related Proofs

We construct a rooted graph $\mathbb{G}$ for a VAS $\mathcal{V} = (\mathbf{x}_0, T)$ which is similar to the coverability tree constructed in [4]. The graph $\mathbb{G}$ is described by a tuple $(P, E, \boldsymbol{\lambda})$ where P is a set of nodes or places, $E \subseteq P \times P$ is a set of edges, and $\boldsymbol{\lambda} : P \cup E \to \mathbb{N}_\omega^n$ is a labeling on the nodes and edges by markings and transitions respectively. By concatenation of edge labels, paths of G can be labeled with transition sequences. The algorithm for construction is described below and in Algorithm 1.

The algorithm initialises $\mathbb{G}$ with no edges and a node r, which is a root, with the label $\mathbf{x}_0$. Then it tries firing transitions from the labels of nodes picked from a set $P^* \subseteq P$ which represents the "nodes to be explored". Let the algorithm pick a node $p \in P^*$ such that $\boldsymbol{\lambda}(p) = \mathbf{x}$ and let $\mathbf{u} \in T$ be fireable from $\mathbf{x}$. If there is no edge $p \xrightarrow{\mathbf{u}}$ then we add a node q and the edge $p \xrightarrow{\mathbf{u}} q$ with $\boldsymbol{\lambda}(q) = \mathbf{x} + \mathbf{u}$.

This procedure would build an infinite rooted tree if not for the following two actions.

1. If we discover a node q by firing $\mathbf{u}$ from p as above, and q has an ancestor q' such that $\boldsymbol{\lambda}(q) = \boldsymbol{\lambda}(q')$, we erase the node q from $\mathbb{G}$ and loop back from p, adding an edge $p \xrightarrow{\mathbf{u}} q'$. This forms a new cycle in the graph $\mathbb{G}$.

2. If q has an ancestor q' such that $\boldsymbol{\lambda}(q') < \boldsymbol{\lambda}(q)$ and $\Omega(\boldsymbol{\lambda}(q')) = \Omega(\boldsymbol{\lambda}(q))$ then we re-label q with $\mathbf{x}$ such that $\mathbf{x}[i] = \omega$ when $\boldsymbol{\lambda}(q')[i] < \boldsymbol{\lambda}(q)[i]$ and $\mathbf{x}[i] = \boldsymbol{\lambda}(q)[i]$ otherwise. If there are multiple nodes which qualify as q', the algorithm chooses one at random.

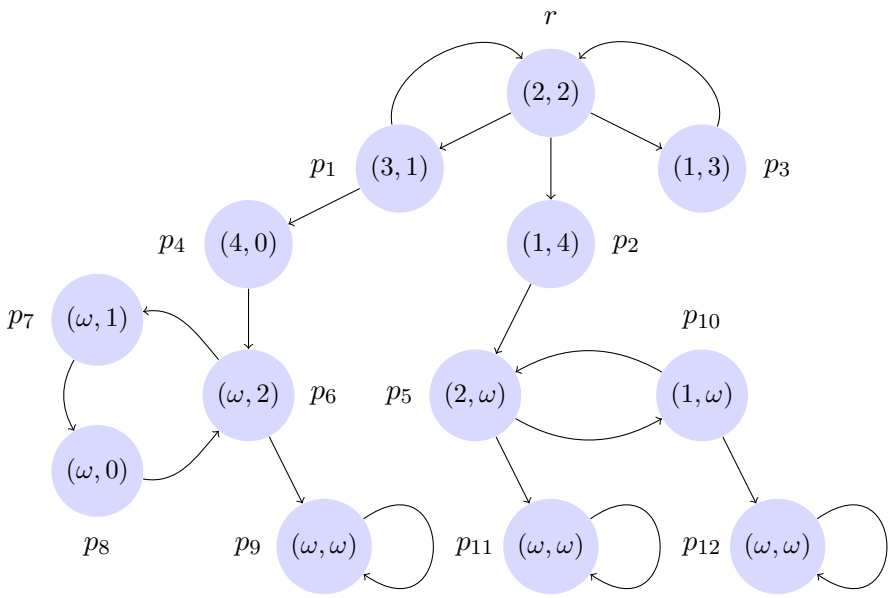

Figure 1: This figure shows a subgraph of $\mathbb{G}$ for a 2-dimensional VAS $\mathcal{V} = (\mathbf{x}_0, \{\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3\})$ where $\mathbf{x}_0 = (2,2)$, $\mathbf{t}_1 = (-1,2)$, $\mathbf{t}_2 = (-1,1)$, $\mathbf{t}_3 = (1,-1)$. Edges have not been labeled with transitions to prevent clutter.

> Such a node q is called an *ω-node*. The set of new components of the marking $\boldsymbol{\lambda}(q))$ is denoted by $\Omega'(\boldsymbol{\lambda}(q))$. We also add a cycle to $\mathbb{G}$ using the label of the path from q' to q. This cycle is called an *ω-cycle*.

In Algorithm 1, NewNode() is self-explanatory function which generates a node which can be added to P, and Path(p, q) is a function which returns the label of the path from p to q in $\mathbb{G}$.

Theorem 1. *Algorithm 1 terminates.*

Proof. Notice that in every action the algorithm adds at least a node or an edge to the graph. Suppose, for the sake of contradiction, that the algorithm ran indefinitely. This would mean it would construct and infinite graph $\mathbb{G}$. Using the description of the algorithm, we now show that $\mathbb{G}$ contains no infinite elementary path. But the graph $\mathbb{G}$ is rooted (with root r) and finite branching (because every node can have at most $|T|$ successors) so we invoke lemma 2 to deduce that $\mathbb{G}$ must be finite.

Observe that in any path of $\mathbb{G}$, once ω occupies some component, it never leaves. Formally, if $p \xrightarrow{*} q$ then $\Omega(\boldsymbol{\lambda}(p)) \subseteq \Omega(\boldsymbol{\lambda}(q))$. So it suffices to show there is no infinite path $p_0, p_1, \ldots, p_n, \ldots$ with $\Omega(\boldsymbol{\lambda}(p_i)) = \Omega(\boldsymbol{\lambda}(p_j))$ for all $i, j \in \mathbb{N}$, because the dimension d of the VAS is finite.

Algorithm 1: A Karp-Miller Graph

Input: A Vector Addition System $\mathcal{V} = (\mathbf{x}_0, \mathbf{T})$
Output: A labeled graph $\mathbb{G} = (P, E, \boldsymbol{\lambda})$

$P \leftarrow \{r\}$; $E \leftarrow \phi$; $P^* \leftarrow \{r\}$; $\boldsymbol{\lambda}(r) \leftarrow \mathbf{x}$;
while $P^* \neq \emptyset$ **do**
  Pick some $p \in P^*$;
  foreach $\mathbf{v} \in \mathbf{V}$ **do**
    if $\mathbf{v} + \boldsymbol{\lambda}(p) \in \mathbb{N}^n_\omega$ **and** $\nexists\, q \in P$ *s.t* $\boldsymbol{\lambda}(p, q) = \mathbf{v}$ **then**
      if $\exists\, q' \in \mathsf{Path}(r, p)$ *s.t* $\boldsymbol{\lambda}(q') = \mathbf{v} + \boldsymbol{\lambda}(p)$ **then**
        // We loop back
        $E = E \cup (p, q')$; $\boldsymbol{\lambda}(p, q') = \mathbf{v}$;
      else
        $q \leftarrow \mathsf{NewNode}()$; $P \leftarrow P \cup \{q\}$;
        $E \leftarrow E \cup \{(p, q)\}$; $\boldsymbol{\lambda}(q) \leftarrow \mathbf{v} + \boldsymbol{\lambda}(p)$;
        $\boldsymbol{\lambda}(p, q) \leftarrow \mathbf{v}$; $P^* \leftarrow P^* \cup q$;
        if $\exists q' \in \mathsf{Path}(r, p)$ *s.t* $\boldsymbol{\lambda}(q') < \mathbf{y}$ **and** $\Omega(\boldsymbol{\lambda}(q')) = \Omega(\mathbf{y})$ **then**
          // The node q is called an ω-node
          foreach $i \in [d]$ **do**
            if $\boldsymbol{\lambda}(q)[i] > \boldsymbol{\lambda}(q')[i]$ **then** $\boldsymbol{\lambda}(q)[i] \leftarrow \omega$;
          end
          Add an ω-cycle at q using $\mathsf{Path}(q', q)$;
        end
      end
    end
  end
end
return $\mathbb{G}$

For the sake of contradiction, suppose $\sigma = p_0, p_1, \ldots, p_n, \ldots$ is an infinite path with $\Omega(\boldsymbol{\lambda}(p_i)) = \Omega(\boldsymbol{\lambda}(p_j))$ for all $i, j \in \mathbb{N}$. Let $|\Omega(\boldsymbol{\lambda}(p_i))| = k$ and $\alpha = \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n, \ldots$ be an infinite sequence of vectors from $\mathbb{N}^{d-k}$ where $\mathbf{x}_i$ is the projection of $\boldsymbol{\lambda}(p_i)$ onto its finite components. Using Lemma 1 we infer the existence of indices $i, j \in \mathbb{N}$ with $i < j$ such that $\mathbf{x}_i \leq \mathbf{x}_j$.

But the two actions performed by the algorithm while trying to apply a transition at a node prevent this. Let q' be some ancestor of q. In the first action the algorithm loops back if $\boldsymbol{\lambda}(q') = \boldsymbol{\lambda}(q)$, and in the second action the algorithm "accelerates" the node q, whenever $\boldsymbol{\lambda}(q') < \boldsymbol{\lambda}(q)$, by re-labeling the responsible components of q with ω. So no infinite elementary path exists in $\mathbf{G}$.

■

Theorem 2. *If* $\mathbf{x} \xrightarrow{\sigma} \mathbf{y}$ *for some* $\sigma = \mathbf{u}_1\mathbf{u}_2\dots\mathbf{u}_n$ *and* $\mathbf{x} = \boldsymbol{\lambda}(p)$ *for some node* $p \in P$*, then there is a unique* σ*-labeled path in* $\mathbb{G}$ *from* p *to a node* $q \in P$ *such that* $\boldsymbol{\lambda}(q) \geq \mathbf{y}$*.*

Proof. Let $\mathbf{x} \xrightarrow{\sigma} \mathbf{y}$ for some $\sigma = \mathbf{u}_1\mathbf{u}_2\dots\mathbf{u}_n$ and $\mathbf{x} = \boldsymbol{\lambda}(p)$ for some node $p \in P$.

Let us write $\mathbf{x} = \mathbf{y}_0 \xrightarrow{\mathbf{u}_1} \mathbf{y}_1 \xrightarrow{\mathbf{u}_2} \dots \xrightarrow{\mathbf{u}_n} \mathbf{y}_n = \mathbf{y}$ in $\mathcal{V}$. We will construct, step-by-step, the σ-labeled path $p = q_0, q_1, \dots, q_n = q$ using induction. Let the base correspond to an empty σ. In this case the lemma is vacuously true, satisfiable using an empty path.

Let us now assume for the sake of induction that for every $j \leq k$, for some $k \geq 0$, if $\mathbf{x} = \mathbf{y}_0 \xrightarrow{\mathbf{u}_1} \dots \xrightarrow{\mathbf{u}_j} \mathbf{y}_j$ and $\mathbf{x} = \boldsymbol{\lambda}(p)$ for some node $p \in P$, then there is a unique path $p = q_0, q_1, \dots, q_j$ with the label $\mathbf{u}_1\mathbf{u}_2\dots\mathbf{u}_j$ such that $\boldsymbol{\lambda}(q_j) \geq \mathbf{y_j}$. We know σ is fireable from x, so $\mathbf{u}_{k+1}$ is fireable from $\mathbf{y}_k$. But $\boldsymbol{\lambda}(q_k) \geq \mathbf{y}_k$, so $\mathbf{u}_{k+1}$ is also fireable from $\boldsymbol{\lambda}(q_k)$. We have the path $q_0, q_1, \dots, q_k$ so far and we will show how to extend it.

If $\boldsymbol{\lambda}(q_k) + \mathbf{u}_{k+1} = \boldsymbol{\lambda}(q_j)$ for some $j \leq k$, then we extend our path with q_{k+1} where $q_{k+1} = q_j$. This creates a cycle in our path, labeled by $\mathbf{u}_{j+1}, \dots, \mathbf{u}_{k+1}$. We deduce that $\mathbf{y}_{k+1} = \mathbf{y}_j$ except perhaps at the ω-components of $\boldsymbol{\lambda}(q_j)$. At its ω-components, $\boldsymbol{\lambda}(q_j) > \mathbf{x}$ for every $x \in \mathbb{N}^d$ and at its non-ω-components, $\boldsymbol{\lambda}(q_j) \geq \mathbf{y}_j$ due to the induction hypothesis. So $\boldsymbol{\lambda}(q_{k+1}) \geq \mathbf{y}_{k+1}$ is maintained.

Else, we find an edge $q_k \xrightarrow{\mathbf{u}_{k+1}} q_{k+1}$ in $\mathbb{G}$. Now there are again two cases. If q_{k+1} is an "accelerated" node, $\boldsymbol{\lambda}(q_{k+1}) > \boldsymbol{\lambda}(q_k) + \mathbf{u}_{k+1}$. If not, $\boldsymbol{\lambda}(q_{k+1}) = \boldsymbol{\lambda}(q_k) + \mathbf{u}_{k+1}$. In either case, $\boldsymbol{\lambda}(q_k) \geq \mathbf{y}_k$ due to the induction hypothesis. By adding $\mathbf{u}_{k+1}$ to both sides of the inequality and using the fact that $\mathbf{y}_{k+1} = \mathbf{y}_k + \mathbf{u}_{k+1}$ we deduce that $\boldsymbol{\lambda}(q_{k+1}) \geq \mathbf{y}_{k+1}$. So we use q_{k+1} to extend our path.

Thus we have shown that every transition sequence which is fireable from the label of some node in $\mathbb{G}$ corresponds to a unique path in $\mathbb{G}$.

■

3.5 Semi-linear sets

If a set $L \subseteq \mathbb{N}^d$ can be represented as $L = \{\mathbf{x}_0 + \sum_{i=1}^{k} \mu_i \mathbf{x}_i \mid \mu_i \in \mathbb{N}\}$ such that $\mathbf{x}_0 \in \mathbb{N}^d$ and $X \subseteq \mathbb{N}^d$, then it is called *linear*. The vector $\mathbf{x}_0$ is the *base* of the linear set and the set $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k\}$ is a finite set of *periods*. The pair $(\mathbf{x}_0, X)$ uniquely determines a linear set and is called the *basis* of the linear set. We may use $L(\mathbf{x}_0, X)$ to denote a linear set with basis $(\mathbf{x}_0, X)$. A set S is called *semi-linear* if it is a finite union of linear sets. Semi-linear sets are closed under union, intersection and complementation [6].

We now prove a result about semi-linear sets which is used to prove semi-linearity of projections of $\mathcal{R}(\mathcal{V})$.

Lemma 3. *Let $S \subseteq \mathbb{N}$ be such that there exist $a, b \in \mathbb{N}$ with $b > 0$ such that for all $s \geq a$ we have $s \in S \implies s + b \in S$. Then S is a semi-linear set.*

Proof. Let us collect all $s \in S$ such that $s \in \{a, a+1, \ldots, a+b-1\}$ in a set C, i.e. $C = \{s \in S \mid a \leq s < a+b\}$. Now for each element $c \in C$ we construct a linear set $\{c + nb \mid n \in \mathbb{N}\}$ which we call $L(c, \{b\})$.

Let us prove that these linear sets are pairwise disjoint. Suppose there exist $c_1, c_2 \in C$ such that $c_1 \neq c_2$ and $L(c_1, \{b\}) \cap L(c_2, \{b\}) \neq \phi$. Then there exist $n_1, n_2 \in \mathbb{N}$ such that $c_1 + n_1 b = c_2 + n_2 b$ and necessarily $n_1 \neq n_2$. This can be rewritten as $c_1 - c_2 = (n_2 - n_1)b$. But since $c_1, c_2 \in C$, and C contains elements of S which are in $[a, a+b)$, we deduce that $|c_1 - c_2| < b$. So it follows that $c_1 - c_2 \neq (n_2 - n_1)b$ which is a contradiction, so the linear sets are pairwise disjoint.

Let us define $S_{\geq a} = \{s \in S \mid s \geq a\}$ and $S_{<a} = \{s \in S \mid s < a\}$. Every $s \in S_{\geq a}$ can be written as $c + nb$ for some $c \in C$ and some $n \in \mathbb{N}$ so it can be represented as in equation 1. $S_{<a}$ is a finite set, so it can be represented as a semi-linear set, as in equation 1.

$$S_{\geq a} = \bigcup_{c \in C} L(c, \{b\}) \qquad S_{<a} = \bigcup_{s \in S_{<a}} L(s, \{0\}) \tag{1}$$

So $S_{\geq a}$ and $S_{<a}$ are semi-linear sets. Therefore $S = S_{\geq a} \cup S_{<a}$ is also a semi linear set, because semi-linear sets are closed under union.

■

4 Reflections and Conclusions

I thought of pursuing a research internship in theoretical computer science as I have liked formal mathematical thinking ever since I was exposed to it. More recently I enjoyed our core course on discrete structures which I took this year. This was my first internship ever, and thus my first time trying research in an academic setting. I was slightly apprehensive about it when I secured it but also pretty excited.

Both me and Prof Finkel were disappointed to end the internship without reaching the goal. Nevertheless, the experience taught me many things. I discovered that research papers are not necessarily written after one has a great idea. The process of research is inseparable from writing articles, which help solidify concepts studied and presents possibilities of coming up with new ideas. Understanding how to pen down my thoughts formally was my biggest learning. I faced a lot of constructive criticism from Prof Finkel while writing my article and it has taught me many lessons which will stay with me and help me for a long time.

The lockdown was a huge setback in many ways, and this being my first internship, I felt the need for more guidance and was periodically lost. I was reading papers without testing of knowledge. There was a goal, but no plan of action or deadlines with presentation. So a point of improvement for me might be to create more short-term deadlines to put a bit of pressure behind the work.

At the same time, working from home has been a novel experience. It forced me to be more independent than I could ever have been and gave me a productive task which I could focus on, and it pacified my worries about COVID and the things which I couldn't control. There were a couple of meetings when I conversed with Prof Finkel using comments on the article and along with being instructive, it was great fun.

During the internship I also got the opportunity to think about where my interests are. I am quite theoretically oriented, but probably this work was a bit too theoretical for my taste. I am being cautious in concluding anything because the lockdown was hardly a normal setting for an internship. But in future projects I would also like to do some implementation work as I think it may be one of my stronger points, and also explore other fields in theoretical as well as applied computer science.

References

[1] Hans Kleine Büning, Theodor Lettmann, and Ernst W. Mayr. Projections of vector addition system reachability sets are semilinear. *Theoretical Computer Science*, 64(3):343 – 350, 1989.

[2] Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. The logical view on continuous petri nets. *ACM Trans. Comput. Logic*, 18(3), August 2017.

[3] Alain Finkel, Serge Haddad, and Igor Khmelnitsky. Minimal Coverability Tree Construction Made Complete and Efficient. In *FoSSaCS 2020 - 23rd International Conference on Foundations of Software Science and Computation Structures*, Dublin, Ireland, April 2020.

[4] Richard M. Karp and Raymond E. Miller. Parallel program schemata. *J. Comput. Syst. Sci.*, 3(2):147–195, May 1969.

[5] John Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8(2):135 – 159, 1979.

[6] Seymour Ginsburg. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, USA, 1966.