PARAMETERIZED COMPLEXITY OF SOME PROBLEMS IN CONCURRENCY AND VERIFICATION

By

M. Praveen

The Institute of Mathematical Sciences, Chennai

A thesis submitted to the
Board of Studies in Computer Science

In partial fulfilment of the requirements
For the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

July 2011

ACKNOWLEDGEMENTS

I wish to thank my advisor Kamal Lodaya for his advice, encouragement and understanding. He taught me concurrency theory and advanced automata theory during my course work. His broad outlook and curiosity for deeper understanding enabled me to work on my thesis involving different areas of theoretical computer science.

I would like to thank R Ramanujam for initiating me into logic and encouraging further studies in it. I took two courses on logic under him and his style of teaching drove me towards using logic to gain useful insights in other areas. He continued helping me with constructive feedback and guidance during my research work.

I would also like to thank Venkatesh Raman for initiating and nurturing my interest in algorithms and parameterized complexity. I took courses on algorithms and parameterized complexity with Venkatesh Raman and his elegant way of teaching enabled me to enjoy working on those subjects.

I would also like to thank Madhavan Mukund from CMI, who has been supportive all through my work and along with Venkatesh Raman and R Ramanujam as members of my doctoral committee provided valuable guidance for my research.

I am grateful to Saket Saurabh, who as a colleague and mentor pointed out research directions and held stimulating discussions, while as a friend, helped tide over some of the not so good times during my PhD. I also thank all staff members of IMSc, from whom I have learnt a lot. I am glad to have had elucidating discussions and received feedback from Stéphane Demri.

I have been fortunate to undertake visits and work in diverse work cultures and research environments. I thank Fedor Fomin and Saket Saurabh for inviting me to visit University of Bergen and Ahmed Bouajjani for inviting me to visit LIAFA laboratory at the University of Paris 7. IMSc sponsored my trip to attend European Summer School on Logic, Language and Information in 2010, which introduced me to several new topics and enabled me to meet leading researchers.

K Narayan Kumar and S P Suresh from CMI have always been appreciative of my work and I have benefitted from discussions I had with them. Many seminars organized by them along with Madhavan Mukund to which I was invited introduced me to new areas of research and gave me the opportunity to interact with scientists from leading universities around the world.

Finally, I would like to thank IMSc for providing excellent facilities for carrying out research work.

Abstract

Formal methods for the analysis of concurrent systems is an active area of research. Many mathematical models like Petri nets, communicating automata, automata with auxiliary storage like counters and stacks, rewrite systems and process algebras have been proposed for modelling concurrent infinite state systems. Efficient algorithms for analysis and the power to express interesting properties of concurrent systems are conflicting goals in these models. Having too much expressiveness results in undecidability, so it is important to get an insight into what kind of restrictions will lead to good analysis algorithms while retaining some expressive power. Restrictions like reversal boundedness in counter automata, disallowing cycles in network of push-down systems etc. lead to decidability in the respective models.

In this thesis, we propose to use the framework of parameterized complexity to study the effect of various restrictions on the complexity of problems related to some models and logics of concurrent systems. Parameterized complexity works by trying to find efficient algorithms for instances of hard problems where one can identify structure that helps in analysis. A numerical parameter is associated with problem instances and algorithms are designed whose time and/or memory requirement is a fast growing function of the parameter, but growing slowly in terms of the size of the instance. On instances where the parameter is small, such algorithms run efficiently. Apart from providing efficient algorithms, parameterized complexity provides a mathematically rigorous way of studying finer structure of the models under analysis.

In the first part of this thesis, we look at the effect of well known graph parameters treewidth and pathwidth on the parameterized complexity of satisfiability of some logics used to specify properties of finite state concurrent systems. This is followed by parameterized complexity of some problems associated with synchronized transition systems and 1-safe Petri nets, which are compactly represented finite state systems. In the second part of the thesis, we look at general Petri nets (which are infinite state) and study the parameterized complexity of coverability, boundedness and extensions of these problems with respect to two parameters.

Contents

1 Introduction **1**

2 Treewidth and Satisfiability of Modal Logics **4**
2.1 Notation and Preliminaries 5
2.2 Treewidth for Modal Logic Formulas 8
2.2.1 On the Relevance of Treewidth for Modal Logic 11
2.3 Reflexive Models 12
2.4 Lower Bounds for Treewidth 13
2.5 Models with Euclidean Property 14
2.6 Transitive Models 17
2.7 Temporal Logics 24
2.8 Remarks and Open Problems 28

3 Synchronized Transition Systems **30**
3.1 Preliminaries 30
3.2 Upper Bound 32
3.3 Lower Bound 32

4 1-safe Petri Nets **35**
4.1 Petri Nets and Problem Definitions 35
4.2 Logics for Specifying Properties 36
4.3 Treewidth, Pathwidth and 1-safe Petri Nets 38
4.3.1 Treewidth, Pathwidth and Graph Pebbling Problems 42
4.4 Benefit Depth and 1-safe Petri Nets 43
4.5 Vertex Cover and Model Checking 1-safe Petri Nets 46
4.6 Open Problems 53

5 Survey of General Petri Nets and Related Concepts **54**
5.1 Petri Nets and Some Properties 54
5.2 Data Words 55
5.2.1 LTL Over Data Words 55
5.2.2 Two Variable Logic on Data Words 56
5.3 Petri Net Languages 58
5.4 Exponential Space Upper Bound for Petri net Coverability and Boundedness 60
5.5 A Unified Approach for Deciding the Existence of Certain Petri Net Paths . 61
5.6 Simulating Exponential Space Turing Machines on Vector Addition Systems with States 62
5.7 Presburger Arithmetic with Monotone Transitive Closure 64
5.8 Branching Vector Addition Systems 66
5.9 Petri Nets and the Theory of Tensor $\star$ 66
5.10 Relevance Logic 67

5.10.1 Implication Conjunction Fusion Fragment of Relevance Logic 67
5.10.2 LR+ Fragment of Relevance Logic 69
5.11 Summary of results 70

6 Petri Nets and Benefit Depth **72**
6.1 System Model 72
6.2 Benefit Depth and Coverability 73
6.3 Logics for Specifying Petri Net Properties 77
6.4 Model Checking Logic of Counting Properties 78
6.4.1 Nested Coverability Properties 78
6.4.2 Boundedness Properties 81

7 Graph Structure and Vertex Cover for Petri Nets **92**
7.1 Vertex Cover for Petri Nets 92
7.2 ParaPspace Algorithm for the Coverability Problem 95
7.3 Model Checking Logic of Counting Properties 98
7.3.1 Nested Coverability Properties 98
7.3.2 Boundedness Properties 99
7.4 Open Problems 107

8 Conclusion **108**

A Brief Introduction to Parameterized Complexity **109**

Chapter 1

Introduction

Parameterized complexity [26] works by trying to find efficient algorithms for instances of hard problems where one can identify structure that helps in analysis. A numerical parameter (usually denoted by k) is associated with problem instances and algorithms are designed whose time and/or memory requirement is a fast growing function of the parameter, but growing slowly in terms of the size of the instance. On instances where the parameter is small, such algorithms run efficiently. For example, with hard problems for which all known algorithms have worst case running time exponential in the size n of the input size, parameterized complexity looks for algorithms with worst case running time $f(k)n^c$, where f is some computable function of the parameter k and c is a constant. Parameterized problems with algorithms of such running time are in the class Fixed Parameter Tractable (FPT), which can be thought of as parameterization of the classical complexity class Polynomial time (PTIME). Appendix A contains a brief introduction to concepts related to FPT and a hierarchy of parameterized complexity classes believed to be intractable. Other classical complexity classes can also be parameterized [36]. There are articles in the literature that consider questions that are essentially parameterized complexity problems related to concurrent systems [87, 45]. We refine such results by considering other parameters and utilizing techniques that have been recently developed in the field of parameterized complexity.

The study of parameterized complexity derived an initial motivation from the study of graph parameters. Many problems that are complete for non-deterministic polynomial time (NP) can be solved in polynomial time on trees and are FPT on graphs that have tree-structured decompositions.

Definition 1.1 (Tree decomposition, treewidth, pathwidth)**.** *A* **tree decomposition** *of a graph $G = (V, E)$ is a pair $(\mathcal{T}, (B_t)_{t \in nodes(\mathcal{T})})$, where $\mathcal{T}$ is a tree and $(B_t)_{t \in nodes(\mathcal{T})}$ is a family of subsets of V such that:*

- *For all $v \in V$, the set $\{t \in nodes(\mathcal{T}) \mid v \in B_t\}$ is nonempty and connected in $\mathcal{T}$.*
- *For every edge $(v_1, v_2) \in E$, there is a $t \in nodes(\mathcal{T})$ such that $v_1, v_2 \in B_t$.*

The width of such a decomposition is the number $\max\{|B_t| \mid t \in nodes(\mathcal{T})\} - 1$. The ***treewidth*** *$tw(G)$ of G is the minimum of the widths of all tree decompositions of G. If the tree $\mathcal{T}$ in the definition of tree decomposition is a path, we get a path decomposition. The* ***pathwidth*** *$pw(G)$ of G is the minimum of the widths of all path decompositions of G.*

Treewidth is a well-studied graph parameter that arises naturally in many contexts. For example, Thorup has shown that flow graphs resulting from structured programs of many languages have small treewidth [93]. Treewidth has also been used as a unifying framework for many decidability results for automata with auxiliary storage [67] and for efficient model checking of First Order logic [2]. Treewidth being such a fundamental and widely occurring

concept, we study its impact on the satisfiability of modal and temporal logics. Modal logics have many applications (reasoning about knowledge [31], programming [81] and hardware verification [84] etc.), in addition to nice computational properties [97, 41]. Many tools have been built for checking satisfiability of modal formulas [53, 77], despite being intractable in the classical sense (PSPACE-complete or NP-complete in most cases). Complexity of modal logic decision problems is well studied [60, 47, 46]. Temporal logic is widely used for formal specification and verification of concurrent systems and extensively researched [15, 98, 68, 59].

In the first part of this thesis, we study the effect of treewidth as a parameter for the satisfiability problems of modal logic, Linear Temporal Logic (LTL) and Computational Tree Logic (CTL). The LTL satisfiability problem is known to be polynomial space (PSPACE)-complete [90] and CTL satisfiability problem is known to be exponential time (EXPTIME)-complete [35, 28]. For modal logic, the satisfiability problem is usually PSPACE-complete or non-deterministic polynomial time (NP)-complete [60], depending on the restrictions imposed on satisfying models. We look for FPT algorithms or for hardness for some parameterized complexity class believed to be intractable (details in Appendix A). It turns out that when there is transitivity or some equivalent concept such as in LTL, CTL and modal satisfiability in transitive models, treewidth does not help. We will see that treewidth does help in the general modal satisfiability problem, thus gathering some evidence that it is transitivity that makes treewidth useless, as far as FPT algorithms are concerned.

Labelled Transition Systems (LTS) are models of sequential systems while extensions like Synchronized LTS and 1-safe Petri nets compactly represent concurrent finite state systems. We continue to study parameterized complexity of problems associated with these models. Most of the problems we consider are PSPACE-complete.

Petri nets, introduced by C. A. Petri [80], are popularly used for modelling concurrent infinite state systems. Using Petri nets to verify various properties of concurrent systems is an ongoing area of research, with abstract theoretical results like [5] and actually constructing tools for C programs like [56]. Reachability is one of the most fundamental problems of Petri nets. Although it is known to be decidable [71, 58], the complexity is not known. Complexity of the reachability problem is known for many subclasses of Petri nets[1], which are the result of various restrictions on Petri nets. Finding such restrictions helps in understanding the structure of Petri nets and in developing techniques. We use parameterized complexity as a mathematically rigorous way of analyzing structural restrictions. With the capability of having an "extended dialog" with hard problems [25], answers to parameterized complexity questions can provide finer understanding of the problems under consideration.

Apart from reachability, coverability and boundedness are some of the most fundamental questions about Petri nets. All three of them are exponential space (EXPSPACE)-hard [66]. Coverability and boundedness are in EXPSPACE [83]. Reachability is known to be decidable [71, 58, 62, 65] but no upper bound is known. Other interesting properties of Petri nets include liveness, deadlock, fairness etc., which especially arise in the context of concurrent reactive systems. Logics have been proposed to uniformly describe such properties of Petri nets [52, 99, 4, 45]. In the second part of this thesis, we study the parameterized complexity of the coverability and boundedness problems. We also study the parameterized complexity of model checking a logic that we have carefully designed to avoid expressing the reachability problem, but is powerful enough to express coverability, boundedness and some extensions.

Let us denote the size of an input instance of a problem by n. If the problem is EXPSPACE-hard, any algorithm will require memory space exponential in n in the worst case. Let us consider some parameter denoted by k. For an EXPSPACE-hard problem such as those mentioned in the previous paragraph, a suitable parameterized complexity theoretic question would be to check if there are algorithms solving the problem using memory space

[1] A survey can be found in [82]

$\mathcal{O}(f(k)poly(n))$. The function $f(k)$ may be any computable function of the parameter k while $poly(n)$ is some polynomial of the input size. Such algorithms are called ParaPspace algorithms. Fundamental complexity theory of such parameterized complexity classes have been studied [36]. We consider two parameters and provide ParaPspace algorithms for coverability, boundedness and model checking the newly designed logic.

This thesis is organized as follows. In Chapter 2, we study the extent to which treewidth can help in obtaining Fixed Parameter Tractable (Fpt) algorithms for satisfiability of various modal logics. In Chapter 3, we give some parameterized complexity results for synchronized transition systems. In Chapter 4, we consider 1-safe Petri nets. Here, the problems under consideration are Pspace-complete and we look for Fpt algorithms (or their absence). We give a brief survey of the literature on concepts and models related to Petri nets and their logics in Chapter 5. In Chapter 6, we introduce a parameter that we call benefit depth, motivated by some refinements in Rackoff's Expspace upper bound [83] for coverability and boundedness. In Chapter 7, we consider another parameter vertex cover and its effect on the complexity of coverability and boundedness. We summarize our results in Chapter 8. Appendix A gives a brief introduction to parameterized complexity and some intuition about why parameter treewidth makes certain problems easier.

Some of the work mentioned above have been published in the following papers.

1. M. Praveen and K. Lodaya. Modelchecking counting properties of 1-safe nets with buffers in parapspace. In *Foundations of Software Technology and Theoretical Computer Science*, volume 4 of *Leibniz International Proceedings in Informatics*, pages 347–358, 2009.

2. M. Praveen. Does treewidth help in modal satisfiability? (extended abstract). In *Mathematical Foundations of Computer Science*, Springer Lecture Notes in Computer Science, volume 6281, pages 580–591, 2010. Full version submitted to journal.

3. M. Praveen. Small vertex cover makes petri net coverability and boundedness easier. In *International Symposium on Parameterized and Exact Computation*, Springer Lecture Notes in Computer Science, volume 6478, pages 216–227, 2010. Full version submitted to journal.

4. M. Praveen and K. Lodaya. Parameterized Complexity Results for 1-safe Petri Nets. In *International Conference on Concurrency Theory*, Springer Lecture Notes in Computer Science, volume 6901, 2011. To appear.

5. M. Praveen. Parameterized Complexity of Analyzing Synchronized Transition Systems. In *Young Researchers Workshop on Concurrency Theory*, 2011. Accepted for presentation.

Chapter 2

Treewidth and Satisfiability of Modal Logics

Treewidth as a parameter has been very successful in obtaining Fixed Parameter Tractable (FPT) algorithms for many classically intractable problems. One such class of problems is constraint satisfaction and closely related problems like satisfiability in propositional logic and the homomorphism problem [18, 88]. There have been recent extensions to quantified constraint satisfaction [13, 78]. In such problems, treewidth is used as a measure of modularity inherent in the given problem instance and algorithms make use of the modularity to increase their efficiency. Understanding the extent to which treewidth can be stretched in such problems is an active area of research [70, 43]. In this chapter, we explore treewidth with respect to satisfiability of modal and temporal logics.

Given a propositional formula in Conjunctive Normal Form (CNF), its *incidence graph* is a graph whose vertices are the clauses and propositional variables. A clause vertex is connected to a variable vertex iff the variable occurs in the clause. We generalize this to modal logic. It is known that any modal logic formula can be effectively converted into a CNF [29, 50]. Given a modal logic formula in CNF, we associate a graph with it. Restricted to propositional CNF formulas (which are modal formulas with modal depth 0), this graph is precisely the incidence graph associated with propositional CNF formulas. We also define CNF fragments of LTL and CTL and associate a graph similar to incidence graphs for formulas in those fragments. We prove that

1. with the treewidth of the graph and the modal depth of the formula as parameters, satisfiability in general models is FPT and so is satisfiability in reflexive models,

2. with the pathwidth and modal depth as parameters, satisfiability in transitive models is W[1]-hard,

3. with the pathwidth and modality depth as parameters, satisfiability of LTL and CTL formulas in CNF is W[1]-hard and

4. with treewidth as the parameter, satisfiability in models that are Euclidean[1] and any combination of reflexive, symmetric and transitive is FPT.

Since modal formulas of modal depth 0 contain all propositional formulas, bounding modal depth alone will not give FPT results (unless PTIME=NP). The main idea behind our FPT results is to construct a relational structure whose domain elements are clauses and literals of a modal formula, with binary relations indicating which literals occur in which clause. Then, satisfiability of the modal formula can be expressed in Monadic Second Order (MSO) logic

[1] A binary relation $\mapsto$ is Euclidean if $\forall x, y, z$, $x \mapsto y$ and $x \mapsto z$ implies $y \mapsto z$.

over the relational structure, enabling us to apply Courcelle's theorem [17]. The lower bound proof involves carefully designing a modal formula of low pathwidth that can implement counting mechanisms using the underlying transitivity. With a suitably defined CNF, we extend the lower bound to satisfiability of LTL and CTL.

Beginning with the seminal paper of Ladner [60], lot of research has been done on the complexity of modal satisfiability. In [46], Halpern considers the effect of bounding different parameters (such as the number of propositional variables, modal depth etc., but not treewidth) on complexity. In [76], Nguyen shows that satisfiability of many modal logics reduce to PTIME under the restriction of Horn fragment and bounded modal depth. In [1], Achilleos et al. consider parameterized complexity of modal satisfiability in general models with the number of propositional variables and other structural aspects (but not treewidth) as parameters. In [2], Adler et al. associate treewidth with First Order (FO) formulae and use it to obtain a FPT algorithm for model checking.

The Complexity of satisfiability of modal logics follow a pattern. In [48], Halpern et al. prove that with the addition of Euclidean property, complexity of (infinitely) many modal logics drop from PSPACE-hard to NP-complete. [49] is another work in this direction. Similar pattern is observed in graded modal logics [57]. With treewidth and modal depth as parameters, our results indicate similar behaviour in the world of parameterized complexity — satisfiability in transitive models is W[1]-hard, while satisfiability in Euclidean and transitive models is FPT, even with treewidth as the only parameter.

2.1 Notation and Preliminaries

We use standard notation about parameterized complexity like FPT algorithms, FPT reductions and W[1]-hardness from [37]. We will also use notation and definitions of relational structures and their tree decompositions from [37]: a *relational vocabulary* τ is a set of relation symbols. Each relation symbol R has an arity $arity(R) \geq 1$. A τ-structure $\mathcal{S}$ consists of a set dom called the *domain* and an interpretation $R^{\mathcal{S}} \subseteq dom^{arity(R)}$ of each relation symbol $R \in \tau$. A *graph* is an $\{E\}$-structure, where E is a binary edge relation. A *tree* is a graph without cycles. If a relational vocabulary τ consists of only unary and binary relation symbols, then any τ-structure can be considered as a graph by treating elements of binary relations as edges of a graph whose vertices are domain elements.

Monadic Second Order (MSO) logic is a powerful logic capable of expressing properties of relational structures. Let $x, y, \dots$ be a set of *first order variables* and $X, Y, \dots$ be a set of *second order variables*. For a given relational vocabulary τ, well formed formulas of MSO logic are those generated by the following syntax:

$$\phi ::= x = y \mid x \in X \mid R(x, y, \dots), R \in \tau \mid \phi \vee \phi \mid \neg\phi \mid \exists x\phi \mid \exists X\phi$$

In the formula $R(x, y, \dots)$ above, the number of elements in the tuple $x, y, \dots$ should be equal to the arity $arity(R)$. In the formula $\exists x\phi$, occurrences of x in ϕ are said to be bound by the quantifier $\exists x$. First order variables not bound by any quantifier are said to be *free variables*. Similar terminology is followed for second order variables. We write $\phi(x, \dots, X, \dots)$ to indicate that $x, \dots$ and $X, \dots$ are free first order and second order variables of ϕ respectively. An *MSO sentence* is a formula without free variables.

Given a MSO formula ϕ, a relational structure $\mathcal{S}$ and an assignment s that assigns an element of the domain of $\mathcal{S}$ to each free first order variable of ϕ and a subset of the domain to each free second order variable, the fact that ϕ is true in $(\mathcal{S}, s)$ (denoted as $\mathcal{S}, s \models \phi$) is defined inductively as follows:

$$\mathcal{S}, s \models x = y \text{ iff } s(x) = s(y)$$

$\mathcal{S}, s \models x \in X$ iff $s(x) \in s(X)$

$\mathcal{S}, s \models R(x, y, \dots)$ iff $(s(x), s(y), \dots) \in R^{\mathcal{S}}$

$\mathcal{S}, s \models \phi_1 \vee \phi_2$ iff $\mathcal{S}, s \models \phi_1$ or $\mathcal{S}, s \models \phi_2$

$\mathcal{S}, s \models \neg\phi$ iff $\mathcal{S}, s \not\models \phi$

$\mathcal{S}, s \models \exists x\phi$ iff there is an assignment s' extending s such that $\mathcal{S}, s' \models \phi$

$\mathcal{S}, s \models \exists X\phi$ iff there is an assignment s' extending s such that $\mathcal{S}, s' \models \phi$

The formula $\forall x\phi$ is an abbreviation for $\neg\exists x\neg\phi$ and $\forall X\phi$ is an abbreviation for $\neg\exists X\neg\phi$. We will extensively use Courcelle's theorem:

Theorem 2.1 (Courcelle's theorem, [16]). *The following problem is* FPT*:*

p-tw-MC(MSO)	
Instance:	*A relational structure $\mathcal{S}$ and a MSO sentence ϕ.*
Parameter:	*The treewidth of $\mathcal{S}$ and the size $\lvert\phi\rvert$ of ϕ.*
Problem:	*Decide whether ϕ is true in $\mathcal{S}$.*

The proof of the above theorem is based on constructing a tree automaton and running it on the tree decomposition of the given graph. The running time of the resulting algorithm is linear in the size of the graph. The size of the tree automaton however can be non-elementary in the size of the MSO sentence, hence the dependence of the running time of the algorithm is in general non-elementary in the size of the MSO sentence. It is known that unless PTIME = NP, such non-elementary dependence can not be avoided [39].

Given a graph, computing a tree decomposition of optimal width is NP-complete, but parameterized by treewidth, it is FPT:

Theorem 2.2 ([7, 8]). *The following two problems are* FPT*:*

p-COMPUTE-TREE-DECOMPOSITION	
Instance:	*A graph G.*
Parameter:	*The treewidth of G.*
Problem:	*Compute optimal tree decomposition of G.*

p-COMPUTE-PATH-DECOMPOSITION	
Instance:	*A graph G.*
Parameter:	*The treewidth of G.*
Problem:	*Compute optimal path decomposition of G.*

We use standard notation for modal logic from [6]: well formed basic modal logic formulae are defined by the grammar $\phi ::= q \in \Phi \mid \bot \mid \neg\phi \mid \phi \vee \psi \mid \Diamond\phi \mid \Box\phi$, where Φ is a set of propositional variables. A *Kripke model* for the basic modal language is a triple $\mathcal{M} = (W, \mapsto, Val)$, where W is a set of worlds, $\mapsto$ is a binary *accessibility* relation on W and $Val : W \times \Phi \to \{\top, \bot\}$ is a valuation function. For $w, v \in W$, if $w \mapsto v$, v is said to be a *successor* of w. The pair $(W, \mapsto)$ is called the *frame* $\mathcal{A}$ underlying $\mathcal{M}$. If $\mapsto$ is reflexive, then $\mathcal{A}$ and $\mathcal{M}$ are said to be a reflexive frame and a reflexive model respectively. Similar nomenclature is followed for other properties of $\mapsto$. The relation $\mapsto$ is Euclidean if for all w_1, w_2, w_3, $w_1 \mapsto w_2$ and $w_1 \mapsto w_3$ implies $w_2 \mapsto w_3$. We denote the fact that a modal formula ϕ is satisfied at a world w in a model $\mathcal{M}$ by $\mathcal{M}, w \models \phi$. For $q \in \Phi$, $\mathcal{M}, w \models q$ iff $Val(w, q) = \top$. Negation $\neg$ and disjunction $\vee$ are treated in the standard way. For any formula ϕ, $\mathcal{M}, w \models \Diamond\phi$ ($\mathcal{M}, w \models \Box\phi$) iff some (all) successor(s) v of w satisfy $\mathcal{M}, v \models \phi$. A modal formula ϕ is *satisfiable* if there is a model $\mathcal{M}$ and a world w in $\mathcal{M}$ such that $\mathcal{M}, w \models \phi$.

Satisfiability in general, reflexive and transitive models are all PSPACE-complete [60], while in equivalence models, it is NP-complete [60].

The modal depth $\mathrm{md}(\phi)$ of a modal formula ϕ is inductively defined as follows. $\mathrm{md}(q) = \mathrm{md}(\bot) = 0$. $\mathrm{md}(\neg\phi) = \mathrm{md}(\phi)$. $\mathrm{md}(\phi \vee \psi) = \max\{\mathrm{md}(\phi), \mathrm{md}(\psi)\}$. $\mathrm{md}(\Diamond\phi) = \mathrm{md}(\Box\phi) = \mathrm{md}(\phi) + 1$. We will use the Conjunctive Normal Form (CNF) for modal logic defined in [50]:

$$\begin{aligned} \mathit{literal} &::= q \mid \neg q \mid \Box \mathit{clause} \mid \Diamond \mathit{CNF} \\ \mathit{clause} &::= \mathit{literal} \mid \mathit{clause} \vee \mathit{clause} \mid \bot \\ \mathit{CNF} &::= \mathit{clause} \mid \mathit{CNF} \wedge \mathit{CNF} \end{aligned}$$

where q ranges over Φ. Any arbitrary modal formula ϕ can be effectively transformed into CNF preserving satisfiability [29]. A *CNF* is a conjunction of clauses and a *clause* is a disjunction of literals. A *literal* is either a propositional variable, a negated propositional variable or a formula of the form $\Box\mathit{clause}$ or $\Diamond\mathit{CNF}$.

Suppose ϕ is a modal formula in CNF. If ϕ is of the form $\mathit{clause}_1 \wedge \mathit{clause}_2 \wedge \cdots \wedge \mathit{clause}_m$, then $\mathit{clause}_1, \mathit{clause}_2, \ldots, \mathit{clause}_m$ and all literals appearing in these clauses are said to be at *level* $\mathrm{md}(\phi)$. If $\Box\mathit{clause}_1$ is a *literal* at some level i, then clause_1 and all literals occurring in clause_1 are said to be at level $i-1$. If $\Diamond\mathit{CNF}$ is a literal at some level i and *CNF* is of the form $\mathit{clause}_1 \wedge \mathit{clause}_2 \wedge \cdots \wedge \mathit{clause}_{m'}$, then $\mathit{clause}_1, \mathit{clause}_2, \cdots, \mathit{clause}_{m'}$ and all literals appearing in these clauses are said to be at level $i-1$. Note that a single propositional variable can occur in the form of a *literal* at different levels.

We will give lower bounds for satisfiability of Linear Temporal Logic (LTL) and Computational Tree Logic (CTL). With a set of atomic propositional variables $\Phi = \{q, \ldots\}$, well formed LTL formulas are those generated by the following syntax.

$$\phi ::= q \in \Phi \mid \neg\, \phi \mid \phi_1 \wedge \phi_2 \mid X\phi \mid \phi_1 U \phi_2$$

These LTL formulas are interpreted on sequences of subsets of Φ. Given a sequence $\pi = \Phi_0\Phi_1\cdots$ of markings, the satisfaction of a LTL formula ϕ at some position i in this sequence is defined as follows.

- $\pi, i \models q$ iff $q \in \Phi_i$.
- $\pi, i \models \neg\phi$ iff $\pi, i \not\models \phi$.
- $\pi, i \models \phi_1 \wedge \phi_2$ iff $\pi, i \models \phi_1$ and $\pi, i \models \phi_2$.
- $\pi, i \models X\phi$ iff there is a position $i+1$ in π and $\pi, i+1 \models \phi$.
- $\pi, i \models \phi_1 U \phi_2$ iff for some $j \geq i$, $\pi, j \models \phi_2$ and for all $i \leq l < j$, $\pi, l \models \phi_1$.

Usual abbreviations are $\top = p \vee \neg p$, $F\phi = \top U \phi$ and $G\phi = \neg F \neg \phi$.

With a set of atomic propositional variables $\Phi = \{q, \ldots\}$, well formed CTL formulas are those generated by the following syntax.

$$\phi ::= q \in \Phi \mid \neg\phi \mid EX\phi \mid AX\phi \mid E[\phi_1 U \phi_2] \mid A[\phi_1 U \phi_2]$$

CTL formulas are interpreted on trees whose nodes are labelled with subsets of Φ. For a tree $\mathcal{T}$ and a node η labelled by the subset Φ_η, satisfaction of a CTL formula is defined as follows.

- $\mathcal{T}, \eta \models q$ iff $q \in \Phi_\eta$.
- $\mathcal{T}, \eta \models \neg\phi$ iff $\mathcal{T}, \eta \not\models \phi$.

- $\mathcal{T}, \eta \models \phi_1 \wedge \phi_2$ iff $\mathcal{T}, \eta \models \phi_1$ and $\mathcal{T}, \eta \models \phi_2$.
- $\mathcal{T}, \eta \models EX\phi$ iff there exists a child η' of η with $\mathcal{T}, \eta' \models \phi$.
- $\mathcal{T}, \eta \models AX\phi$ iff for every child η' of η, $\mathcal{T}, \eta' \models \phi$.
- $\mathcal{T}, \eta \models E[\phi_1 U \phi_2]$ iff for some path $\eta_0\eta_1\eta_2\cdots$ of $\mathcal{T}$ starting from $\eta = \eta_0$, there exists $i \geq 0$ such that $\mathcal{T}, \eta_i \models \phi_2$ and for all j with $0 \leq j < i$, $\mathcal{T}, \eta_j \models \phi_1$.
- $\mathcal{T}, \eta \models A[\phi_1 U \phi_2]$ iff for every path $\eta_0\eta_1\eta_2\cdots$ of $\mathcal{T}$ starting from $\eta = \eta_0$, there exists $i \geq 0$ such that $\mathcal{T}, \eta_i \models \phi_2$ and for all j with $0 \leq j < i$, $\mathcal{T}, \eta_j \models \phi_1$.

Usual abbreviations are $EF\phi = E[\top U \phi]$, $AG\phi = \neg EF\neg\phi$, $AF\phi = A[\top U \phi]$ and $EG\phi = \neg AF\neg\phi$.

2.2 Treewidth for Modal Logic Formulas

In this section, we will associate a relational structure with a modal CNF formula. We show that checking satisfiability of a modal CNF formula is FPT, parameterized by modal depth and the treewidth of the associated relational structure. We begin with an example modal CNF formula.

Consider the modal CNF formula $[\neg q \vee \Box[r \vee \neg s]] \wedge [q \vee \neg r] \wedge [r \vee \Diamond[\neg s]] \wedge [\neg r \vee \Diamond[(t \vee \neg s) \wedge r]]$. Its modal depth is 1 and has 4 clauses at level 1. Figure 2.1 shows a graphical representation of this formula, which is very similar to the formula's syntax tree. The 4 clauses at level 1 are represented by e_1, e_2, e_3 and e_4. e_1 represents the clause $\{\neg q \vee \Box[r \vee \neg s]\}$. Since $\neg q$ occurs as a literal in this clause, there is a dotted arrow from e_1 to q. $\Box[r \vee \neg s]$ (represented by e_9) also occurs as a literal in clause e_1 and hence there is an arrow from e_1 to e_9. e_4 represents the fourth clause at level 1, which contains $\Diamond[(t \vee \neg s) \wedge (r)]$ as a literal. This $\Diamond CNF$ formula is represented by e_{10}. The two clauses $(t \vee \neg s)$ and (r) are represented by e_7 and e_8 respectively and are connected to e_{10} by arrows. The propositional variable r occurs as literal at 2 levels, indicated as Lv_0 and Lv_1.

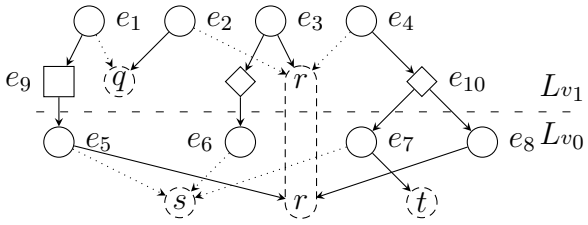


Figure 2.1: Relational structure associated with the modal formula $[\neg q \vee \Box[r \vee \neg s]] \wedge [q \vee \neg r] \wedge [r \vee \Diamond[\neg s]] \wedge [\neg r \vee \Diamond[(t \vee \neg s) \wedge r]]$

Now we will formalize the above example. The intuition behind the following definition is to represent all clauses and literals of a modal CNF formula by the domain elements of a relational structure. Binary relations are used to indicate which literals occur in which clause (and which clauses occur in which literal). Unary relations are used to indicate which elements represent literals and which elements represent clauses. This will enable us to reason about clauses, literals and their dependencies using MSO formulas over the relational structure.

Definition 2.3. *Given a modal CNF formula ϕ, we associate with it a relational structure $\mathcal{S}(\phi)$. It will have one domain element for every clause in ϕ. It will have one domain element*

for every literal of the form $\Box$clause or $\Diamond$CNF in ϕ. It will also have one domain element for every propositional variable used in ϕ.

The relational structure will have two binary relations Oc (occurs) and $\overline{Oc}$ (occurs negatively). The relation $\overline{Oc}$ is the one represented as dotted arrows in Fig. 2.1 — $\overline{Oc}(e_1, e_2)$ iff e_1 represents a clause and e_2 represents a propositional variable occurring negated as a literal in the clause represented by e_1. If e_1 represents a clause, then $Oc(e_1, e_2)$ iff e_2 represents a literal (occurring in the clause represented by e_1) of the form $\Box$clause, $\Diamond$CNF or a non-negated propositional variable. If e_1 represents a literal of the form $\Box$clause, then $Oc(e_1, e_2)$ iff e_2 represents the corresponding clause. If e_1 represents a literal of the form $\Diamond$CNF, then $Oc(e_1, e_2)$ iff e_2 represents a clause in the corresponding CNF. Finally, the following unary relations are present:

Cl	:	*contains all domain elements representing clauses*
Lt	:	*all domain elements representing literals*
$\mathcal{B}_\Box$	:	*all literals of the form $\Box$clause*
$\mathcal{D}_\Diamond$	:	*all literals of the form $\Diamond$CNF*
$(Lv_i)_{0 \leq i \leq md(\phi)}$	:	*all clauses and literals at level i*

For clauses and literals of the form $\Box$*clause* or $\Diamond$*CNF*, there is one domain element for every occurrence of the clause or literal. For example, if the literal $\Diamond(q_1 \wedge q_2)$ occurs in two different positions of a big formula ϕ, the two occurrences will be represented by two different domain elements in $\mathcal{S}(\phi)$. In contrast, different occurrences of a literal that is just a propositional variable or its negation will be represented by the same domain element. In the rest of this chapter, whenever we refer to the treewidth of a modal CNF formula ϕ, we mean the treewidth of $\mathcal{S}(\phi)$.

If e_1 represents a *clause*, $Oc(e_1, e_2)$ means that the clause represented by e_1 can be satisfied by satisfying the literal represented by e_2. Similarly, $\overline{Oc}(e_1, e_2)$ means that the clause represented by e_1 can be satisfied by setting the propositional variable represented by e_2 to false.

If $Cl_0 \subseteq Cl \cap Lv_0$ is a subset of domain elements representing clauses at level 0, let $CNF(Cl_0)$ be the modal CNF formula that is the conjunction of clauses represented by domain elements in Cl_0. We will now see how to check satisfiability of $CNF(\{e_7, e_8\})$ in Fig. 2.1 and describe the generalization of this process given in (2.2) below. We use cl and lt for first order variables intended to represent clauses and literals respectively. First of all, there must be a subset $Tr_0 \subseteq \{r, s, t\} = Lt \cap Lv_0$ that will be set to $\top$, as written in the beginning of (2.2). Then, we must check that this assignment satisfies each clause cl in Cl_0, written as $\forall cl \in Cl_0$ in (2.1). To check that the clause represented by e_7 is satisfied, either a positively occurring literal like t must be set to $\top$ and hence in Tr_0 (written as "$\exists lt \in Tr_0 : Oc(cl, lt)$" in (2.1)) or a negatively occurring literal like s must be set to $\bot$ and hence not in Tr_0 ("$\exists lt \notin Tr_0 : \overline{Oc}(cl, lt)$" in (2.1)). A similar argument applies to e_8 as well.

$$\zeta(Cl, Tr) = \forall cl \in Cl\{[\exists lt \in Tr : Oc(cl, lt)] \vee [\exists lt \notin Tr : \overline{Oc}(cl, lt)]\} \tag{2.1}$$

$$\xi[0](Cl_0) \triangleq \exists Tr_0 \subseteq (Lt \cap Lv_0) : \zeta(Cl_0, Tr_0) \tag{2.2}$$

$$\begin{aligned} \xi[i](Cl_i) \triangleq \exists Tr_i \subseteq (Lt \cap Lv_i) : \zeta(Cl_i, Tr_i) \\ \wedge [B_{\Box i-1} = \{cl' \in (Cl \cap Lv_{i-1}) \mid \exists lt' \in Tr_i \cap \mathcal{B}_\Box, Oc(lt', cl')\} \Rightarrow \\ \forall lt \in Tr_i \cap \mathcal{D}_\Diamond : D_{\Diamond i-1} = \{cl \in (Cl \cap Lv_{i-1}) \mid Oc(lt, cl)\} \Rightarrow \\ \xi[i-1](D \Diamond_{i-1} \cup B_{\Box i-1})] \end{aligned} \tag{2.3}$$

Checking satisfiability at higher levels is slightly more complicated. Suppose $Cl_i \subseteq Cl \cap Lv_i$ is a subset of clauses at level i. We will take $Cl_1 = \{e_1, e_3, e_4\}$ from Fig. 2.1 as an example. If some world w in some Kripke model $\mathcal{M}$ satisfies $CNF(Cl_1)$, there must be some subset Tr_1 of literals at level 1 satisfied at w ("$\exists Tr_i \subseteq (Lt \cap Lv_i)$" in (2.3)). As before,

we check that for every clause represented in Cl_1 ("$\forall cl \in Cl_i$" in (2.1)), there is either a positively occurring literal in Tr_1 ("$\exists lt \in Tr_i : Oc(cl, lt)$" in (2.1)) or a negatively occurring literal not in Tr_1 ("$\exists lt \notin Tr_i : \overline{Oc}(cl, lt)$" in (2.1)). Next, we must check that the literals we have chosen to be satisfied at w (by putting them into Tr_1) can actually be satisfied. Suppose Tr_1 was $\{e_9, q, r, e_{10}\}$. Since e_9 represents a literal of the form $\Box clause$ (with the *clause* represented by domain element e_5), we are committed to satisfy the clause represented by e_5 in any world succeeding w. Let $B\Box_0 = \{e_5\}$ be the set of clauses occurring at level 0 that we have committed to as a result of choosing corresponding $\Box clause$ literals to be in Tr_1 ("$B\Box_{i-1} = \{cl' \in (Cl \cap Lv_{i-1}) \mid \exists lt' \in Tr_i \cap \mathcal{B}\Box, Oc(lt', cl')\}$" in (2.3)). Now, since we have also chosen e_{10} to be in Tr_1 and e_{10} represents a $\Diamond CNF$ formula, there is a demand to create a world w' that succeeds w and satisfies the corresponding *CNF* formula. We have to check that every such demand in Tr_1 can be satisfied ("$\forall lt \in Tr_i \cap \mathcal{D}\Diamond$" in (2.3)) by creating successor worlds. In case of the demand created by e_{10}, $\{e_7, e_8\} = D\Diamond_0$ is the set of clauses in the demanded *CNF* formula ("$D\Diamond_{i-1} = \{cl \in (Cl \cap Lv_{i-1}) \mid Oc(lt, cl)\}$" in (2.3)). Our aim now is to create a successor world w' in which all clauses represented in $D\Diamond_0$ are satisfied. However, w' is a successor world and we have already committed to satisfying all clauses represented in $B\Box_0$ in all successor worlds. Hence, we actually check if the clauses represented in $B\Box_0 \cup D\Diamond_0$ are satisfiable by inductively invoking $\xi[0](D\Diamond_0 \cup B\Box_0)$ ("$\xi[i-1](D\Diamond_{i-1} \cup B\Box_{i-1})$" in (2.3)).

Now, we formalize the above arguments. The part of the formula $[B\Box_{i-1} = \{cl' \in (Cl \cap Lv_{i-1}) \mid \exists lt' \in Tr_i \cap \mathcal{B}\Box, Oc(lt', cl')\} \Rightarrow \cdots]$ in (2.3) can be written in formal MSO syntax as follows:

$$\forall B\Box_{i-1} \quad \forall cl' \quad B\Box_{i-1}(cl') \Leftrightarrow [Cl(cl') \wedge Lv_{i-1}(cl') \wedge \exists lt'\, Tr_i(lt') \wedge \mathcal{B}\Box(lt') \wedge Oc(cl', lt')] \Rightarrow \cdots$$

We will continue to use the slightly informal syntax of (2.2) and (2.3). The following lemma formalizes the meaning of $\zeta(Cl, Tr)$, which will occur repeatedly in many other formulas.

Lemma 2.4. *Let ϕ be a modal CNF formula, Cl be any subset of clauses, Tr be any subset of literals and $\zeta(Cl, Tr) = \forall cl \in Cl\{[\exists lt \in Tr : Oc(cl, lt)] \vee [\exists lt \notin Tr : \overline{Oc}(cl, lt)]\}$. Let $\mathcal{M}$ be some Kripke model and w be a world in it. If $\zeta(Cl, Tr)$ is true in $\mathcal{S}(\phi)$, all literals in Tr are satisfied at w and any propositional variable not in Tr is set to $\perp$ at w, then all clauses in Cl are satisfied at w. If all clauses in Cl are satisfied at w and Tr is the set of literals satisfied at w that are sub-formulas of some clause in Cl, then $\zeta(Cl, Tr)$ is true in $\mathcal{S}(\phi)$.*

Proof. Suppose $\zeta(Cl, Tr)$ is true in $\mathcal{S}(\phi)$ and cl is some clause in Cl. If $[\exists lt \in Tr : Oc(cl, lt)]$ is true in $\mathcal{S}(\phi)$, then some literal in Tr occurs in cl, so cl is satisfied at w (since lt is satisfied at w). If $[\exists lt \notin Tr : \overline{Oc}(cl, lt)]$ is true in $\mathcal{S}(\phi)$, then a propositional variable not in Tr (and hence set to $\perp$ in w) occurs negatively in cl, so cl is satisfied at w.

Suppose Tr is the set of literals satisfied at w that are sub-formulas of some clause in Cl and all clauses in Cl are satisfied at w. Let cl be any clause in Cl. Since cl is satisfied at w, either a positive literal occurring in cl is satisfied at w or a propositional variable occurring negatively in cl is set to $\perp$ in w. In the first case, $[\exists lt \in Tr : Oc(cl, lt)]$ is true in $\mathcal{S}(\phi)$ and in the second case, $[\exists lt \notin Tr : \overline{Oc}(cl, lt)]$ is true in $\mathcal{S}(\phi)$. Therefore, $\zeta(Cl, Tr)$ is true in $\mathcal{S}(\phi)$. □

Lemma 2.5. *The property $\xi[i](Cl_i)$ in (2.3) can be written in a MSO logic formula of size linear in i. If ϕ is any modal formula in CNF and Cl_i is any subset of domain elements representing clauses at level i, then $CNF(Cl_i)$ is satisfiable iff $\xi[i](Cl_i)$ is true in $\mathcal{S}(\phi)$.*

Proof. We will prove that the length $|\xi[i]|$ of $\xi[i]$ is linear in i by induction. Let c be the length of $\xi[i]$ without length of $\xi[i-1]$ counted. As can be seen, $|\xi[0]| \leq c$. Inductively assume that $|\xi[i-1]| \leq ic$. Then, $|\xi[i]| = c + |\xi[i-1]|$. Hence, $|\xi[i]| \leq c + ic = c(i+1)$.

We will now prove the second claim by induction on i.

Base case $i = 0$*:* All literals at level 0 are propositional variables or their negations. Suppose $\xi[0](Cl_0)$ is true in $\mathcal{S}(\phi)$. Lemma 2.4 implies that in a world w, if we set exactly those propositional variables occurring in Tr_0 to $\top$, all clauses in Cl_0 are satisfied at w. If all clauses in Cl_0 are satisfied at some world w, then let Tr_0 be the set of propositional variables set to $\top$ in w that occur as literals in some clause in Cl_0. Lemma 2.4 then implies that $\xi[0](Cl_0)$ is true in $\mathcal{S}(\phi)$.

Induction step: Suppose Cl_i is a subset of domain elements representing clauses occurring at level i and $\xi[i](Cl_i)$ is true in $\mathcal{S}(\phi)$. Consider a Kripke model $\mathcal{M}$ with one world w where the propositional variables occurring in Tr_i are set to $\top$ and others are set to $\bot$. If we can prove that all literals occurring in Tr_i can be satisfied at w, then Lemma 2.4 implies that all clauses in Cl_i are satisfied at w. So let $B_{\Box i-1} = \{cl' \in (Cl \cap Lv_{i-1}) \mid \exists lt' \in Tr_i \cap \mathcal{B}_\Box, Oc(lt', cl')\}$ be the set of clauses that we have to satisfy in all successors of w, since the corresponding $\Box clause$ is in Tr_i. Let $lt \in Tr_i \cap \mathcal{D}_\Diamond$ be a literal of the form $\Diamond CNF$ in Tr_i such that $D\Diamond_{i-1} = \{cl \in (Cl \cap Lv_{i-1}) \mid Oc(lt, cl)\}$ is the set of clauses to be satisfied at a successor of w in order to satisfy lt at w. Since $\xi[i-1](D \Diamond_{i-1} \cup B_{\Box i-1})$ is true in $\mathcal{S}(\phi)$, by induction hypothesis there is a world w' in some Kripke model $\mathcal{M}'$ where all clauses in $D \Diamond_{i-1} \cup B_{\Box i-1}$ are satisfied. Make w' a successor of w to satisfy lt at w. Repeat this for every $lt \in Tr_i \cap \mathcal{D}_\Diamond$ to satisfy all literals in Tr_i at w.

Now we will prove the other direction of the induction step. Suppose Cl_i is a subset of domain elements representing clauses occurring at level i and there is a Kripke model $\mathcal{M}$ and a world w such that $\mathcal{M}, w \models CNF(Cl_i)$. Let Tr_i be the set of literals satisfied at w that are sub-formulas of some clause in Cl_i. Lemma 2.4 implies that $\forall cl \in Cl_i : \left[(\exists lt \in Tr_i : Oc(cl, lt)) \vee \left(\exists lt \notin Tr_i : \overline{Oc}(cl, lt)\right)\right]$ is true in $\mathcal{S}(\phi)$. Suppose $B_{\Box i-1} = \{cl' \in (Cl \cap Lv_{i-1}) \mid \exists lt' \in Tr_i \cap \mathcal{B}_\Box, Oc(lt', cl')\}$ is the set of clauses such that the corresponding $\Box clause$ is in Tr_i. Let $lt \in Tr_i \cap \mathcal{D}_\Diamond$ be a literal of the form $\Diamond CNF$ in Tr_i such that $D\Diamond_{i-1} = \{cl \in (Cl \cap Lv_{i-1}) \mid Oc(lt, cl)\}$ is the set of clauses in the corresponding CNF formula. Since all literals in Tr_i are satisfied at w, there must be a successor w' of w where all clauses in $D \Diamond_{i-1} \cup B_{\Box i-1}$ are satisfied. By induction hypothesis, $\xi[i-1](D \Diamond_{i-1} \cup B_{\Box i-1})$ is true in $\mathcal{S}(\phi)$. Hence, $\xi[i](Cl_i)$ is true in $\mathcal{S}(\phi)$. □

Theorem 2.6. *Given a modal CNF formula ϕ, there is a* FPT *algorithm that checks if ϕ is satisfiable in general models, with treewidth of $\mathcal{S}(\phi)$ and modal depth of ϕ as parameters.*

Proof. Given ϕ, $\mathcal{S}(\phi)$ can be constructed in polynomial time. To check that all clauses of ϕ at level $\mathrm{md}(\phi)$ are satisfiable in some world w of some Kripke model $\mathcal{M}$, we check whether the formula $\exists Cl_{\mathrm{md}(\phi)} \forall cl (Cl_{\mathrm{md}(\phi)}(cl) \Leftrightarrow (Cl(cl) \wedge Lv_{\mathrm{md}(\phi)}(cl))) \wedge \xi[\mathrm{md}(\phi)](Cl_{\mathrm{md}(\phi)})$ is true in $\mathcal{S}(\phi)$. By Lemma 2.5, this is possible iff ϕ is satisfiable and length of the above formula is linear in $\mathrm{md}(\phi)$. An application of Courcelle's theorem will give us the FPT algorithm. □

2.2.1 On the Relevance of Treewidth for Modal Logic

In Theorem 2.6, the role of treewidth is buried inside the proof of Courcelle's theorem. Here, we try to motivate why small treewidth implies more efficient algorithms. Informally, treewidth is a measure of how close a graph is to being a tree. Given a modal logic formula ϕ, the associated structure $\mathcal{S}(\phi)$ is very similar to the syntax tree of ϕ. The structure $\mathcal{S}(\phi)$ is not a tree (i.e., it has cycles) because a single propositional variable may be shared by many clauses of the formula. Thus, if very few variables are shared across clauses, $\mathcal{S}(\phi)$ is very close to a tree, i.e., $\mathcal{S}(\phi)$ will have small treewidth. In the example of Fig. 2.1, the propositional variable r is shared by the clauses represented by domain elements e_2, e_3 and e_4. If we replace r by r' in the clause represented e_4 as in Fig. 2.2, the number of shared variables and cycles will decrease. For example, e_4 was part of many cycles in Fig. 2.1 but

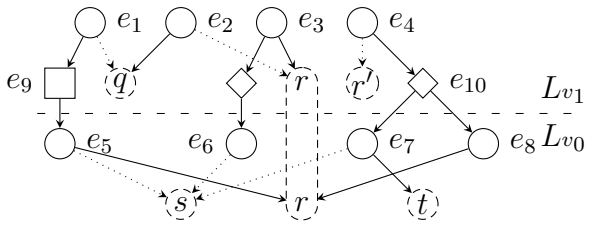


Figure 2.2: Relational structure associated with the modal formula $\{\neg r \vee \Box[r]\} \wedge \{r \vee \Diamond \bot\} \wedge \{r \vee \Diamond[\neg r]\} \wedge \{\neg r \vee \Diamond[(t \vee \neg r) \wedge (r)]\}$

not so in Fig. 2.2. Small treewidth thus implies less sharing of variables across clauses and hence less work for the algorithm.

Treewidth is a very fundamental concept and naturally arises in many contexts, even in industrial applications like software verification [93]. Applications of treewidth related techniques to propositional logic is extensively studied — see [34, Section 1.4] and references therein. Modal logic being a natural and very useful extension of propositional logic, we might expect some benefit by exploring applicability of treewidth related techniques to modal logic.

The set of modal formulas with small treewidth is powerful enough to encode complex formulas. In section 2.4, we prove that there are classes of modal formulas with constant treewidth whose satisfiability is PSPACE-complete. Hence, the restriction of bounded treewidth is not a severe one.

2.3 Reflexive Models

In this section, we extend the basic technique described in section 2.2 to satisfiability in reflexive models. Let $Oc^*(x,y) \triangleq \forall X[X(x) \wedge \forall z \forall u(X(z) \wedge Oc(z,u) \Rightarrow X(u))] \Rightarrow X(y)$ be the reflexive transitive closure of the binary Oc relation.

Let Cl_i be some set of domain elements representing clauses at level at most i. The property $\xi[i](Cl_i)$ defined below checks if there is a reflexive Kripke model $\mathcal{M}$ and a world w in it that satisfies all clauses in Cl_i. Recall the definition of $\zeta(Cl, Tr)$ from (2.1).

$$\xi[0](Cl_0) \triangleq \exists Tr_0 \subseteq (Lt \cap Lv_0) : \zeta(Cl_0, Tr_0) \tag{2.4}$$

$$\begin{aligned}\xi[i](Cl_i) \triangleq \exists Tr_i \subseteq Lt :& \\ & \forall lt \in Tr_i \quad \exists cl \in Cl_i : Oc^*(cl, lt) \\ & \wedge B_{\Box_{i-1}} = \{cl' \in Cl \mid \exists lt' \in Tr_i \cap \mathcal{B}_\Box, Oc(lt', cl')\} \Rightarrow \\ & \zeta(Cl_i \cup B_{\Box_{i-1}}, Tr_i) \\ & \wedge \forall lt \in Tr_i \cap \mathcal{D}\Diamond : D\Diamond_{i-1} = \{cl \in Cl \mid Oc(lt, cl)\} \Rightarrow \\ & \qquad \xi[i-1](D\Diamond_{i-1} \cup B_{\Box_{i-1}})\end{aligned} \tag{2.5}$$

Lemma 2.7. *The property $\xi[i](Cl_i)$ can be written in a MSO logic formula of size linear in i. If ϕ is any modal formula in CNF and Cl_i is any subset of domain elements representing clauses at level at most i, then $CNF(Cl_i)$ is satisfiable in a reflexive model iff $\xi[i](Cl_i)$ is true in $\mathcal{S}(\phi)$.*

Proof. We will prove that the length $|\xi[i]|$ of $\xi[i]$ is linear in i by induction. Let c be the length of $\xi[i]$ without the length of $\xi[i-1]$ counted. As can be seen, $|\xi[0]| \leq c$. Inductively assume that $|\xi[i-1]| \leq ic$. Then, $|\xi[i]| = c + |\xi[i-1]|$. Hence, $|\xi[i]| \leq c + ic = c(i+1)$.

We will now prove the second claim by induction on i.

Base case $i = 0$: Same as the base case in the proof of Lemma 2.5.

Induction step: Suppose Cl_i is a subset of domain elements representing clauses occurring at level at most i and $\xi[i](Cl_i)$ is true in $\mathcal{S}(\phi)$. We will build a reflexive Kripke model $\mathcal{M}$ and prove that it has a world w such that $\mathcal{M}, w \models CNF(Cl_i)$. We will start with a single world w in which the propositional variables occurring in Tr_i are set to $\top$ and others are set to $\bot$. Let $B_{\Box_{i-1}} = \{cl' \in Cl \mid \exists lt' \in Tr_i \cap \mathcal{B}_\Box, Oc(lt', cl')\}$ be the set of clauses such that the corresponding $\Box$*clause* is in Tr_i, so that all clauses in $B_{\Box_{i-1}}$ are to be satisfied at all successors of w, including w itself. The condition $\forall lt \in Tr_i \quad \exists cl \in Cl_i : Oc^*(cl, lt)$ ensures that all literals in Tr_i are at level at most i. Suppose $lt \in Tr_i \cap \mathcal{D}_\Diamond$ is any literal of the form $\Diamond CNF$ in Tr_i such that $D_{\Diamond_{i-1}} = \{cl \in Cl \mid Oc(lt, cl)\}$ is the set of clauses in the corresponding CNF. Since all clauses in $D_{\Diamond_{i-1}} \cup B_{\Box_{i-1}}$ are at level at most $i-1$ and $\xi[i-1](D_{\Diamond_{i-1}} \cup B_{\Box_{i-1}})$ is true in $\mathcal{S}(\phi)$, by induction hypothesis there is a reflexive Kripke model $\mathcal{M}'$ with a world w' in it that satisfies all clauses in $D_{\Diamond_{i-1}} \cup B_{\Box_{i-1}}$. Make w' a successor of w to satisfy lt at w. Repeat this for every $lt \in Tr_i \cap \mathcal{D}_\Diamond$. By induction on the modal depth of any clause $cl \in Cl_i \cup B_{\Box_{i-1}}$, we show that cl is satisfied at w. If $\left(\exists lt \notin Tr_i : \overline{Oc}(cl, lt)\right)$ is true in $\mathcal{S}(\phi)$, a propositional variable occurring negatively in cl is set to $\bot$ at w. If $(\exists lt \in Tr_i : Oc(cl, lt))$ is true in $\mathcal{S}(\phi)$ and lt is a propositional variable, there is a propositional variable occurring in cl set to $\top$ in w. If lt is of the form $\Diamond CNF$, we added some world w' succeeding w to satisfy lt at w. If lt is of the form $\Box$*clause*, then the corresponding *clause* cl' is in $B_{\Box_{i-1}}$ and has modal depth lower than cl. By induction hypothesis on the modal depth of cl', it is satisfied at w. It is also satisfied at all other successors of w by construction.

Now we prove the other direction of the induction step. Suppose Cl_i is a subset of domain elements representing clauses occurring at level at most i and that there is a reflexive Kripke model $\mathcal{M}$ and a world w such that $\mathcal{M}, w \models CNF(Cl_i)$. We prove that $\xi[i](Cl_i)$ is true in $\mathcal{S}(\phi)$. To begin with, we choose Tr_i to be the set of precisely those literals satisfied at w that are sub-formulas of some clause in Cl_i. This will ensure that $\forall lt \in Tr_i \quad \exists cl \in Cl_i : Oc^*(cl, lt)$ is true in $\mathcal{S}(\phi)$. Let $B_{\Box_{i-1}} = \{cl' \in Cl \mid \exists lt' \in Tr_i \cap \mathcal{B}_\Box, Oc(lt', cl')\}$ be the set of clauses such that the corresponding $\Box$*clause* is in Tr_i. The world w satisfies all clauses in Cl_i and since w is its own successor, it also satisfies all clauses in $B_{\Box_{i-1}}$. Therefore, Lemma 2.4 implies that $\forall cl \in Cl_i \cup B_{\Box_{i-1}} : \left[(\exists lt \in Tr_i : Oc(cl, lt)) \vee \left(\exists lt \notin Tr_i : \overline{Oc}(cl, lt)\right)\right]$ is true in $\mathcal{S}(\phi)$.

Let lt be any literal of the form $\Diamond CNF$ in Tr_i and let $D_{\Diamond_{i-1}} = \{cl \in Cl \mid Oc(lt, cl)\}$ be the set of clauses in the corresponding CNF formula. Since w satisfies lt, there must be a successor w' of w that satisfies all clauses in $D_{\Diamond_{i-1}}$ and also all clauses in $B_{\Box_{i-1}}$ since w' is a successor of w. Since all clauses in $D_{\Diamond_{i-1}} \cup B_{\Box_{i-1}}$ are at level at most $i-1$ and w' is a world in a reflexive Kripke model that satisfies all clauses in $D_{\Diamond_{i-1}} \cup B_{\Box_{i-1}}$, we can apply induction hypothesis to conclude that $\xi[i-1](D_{\Diamond_{i-1}} \cup B_{\Box_{i-1}})$ is true in $\mathcal{S}(\phi)$. □

Theorem 2.8. *Given a modal CNF formula ϕ, there is a* FPT *algorithm that checks if ϕ is satisfiable in reflexive models, with the treewidth of $\mathcal{S}(\phi)$ and the modal depth of ϕ as parameters.*

Proof. Given ϕ, $\mathcal{S}(\phi)$ can be constructed in polynomial time. To check that all clauses of ϕ at level $\mathrm{md}(\phi)$ are satisfiable in some world w of some reflexive Kripke model $\mathcal{M}$, we check whether the formula $\exists Cl_{\mathrm{md}(\phi)} \forall cl (Cl_{\mathrm{md}(\phi)}(cl) \Leftrightarrow (Cl(cl) \wedge Lv_{\mathrm{md}(\phi)}(cl))) \wedge \xi[\mathrm{md}(\phi)](Cl_{\mathrm{md}(\phi)})$ is true in $\mathcal{S}(\phi)$. By Lemma 2.7, this is possible iff ϕ is satisfiable in a reflexive model. The length of the above formula is linear in $\mathrm{md}(\phi)$. An application of Courcelle's theorem will give us the FPT algorithm. □

2.4 Lower Bounds for Treewidth

In [46], Halpern proved that even with one propositional variable, modal satisfiability is PSPACE-hard in general models and in reflexive models. This hardness proof involves a

modal formula and here, we will observe that the modal formula is in CNF with constant treewidth. This implies that with treewidth alone as parameter, modal satisfiability is not FPT unless PTIME = PSPACE.

Let $\mathcal{F}$ be a propositional 3-CNF formula with variables $q_1, q_2, \ldots, q_n$. Let $Q_1, Q_2, \ldots, Q_n \in \{\forall, \exists\}$ be Boolean quantifiers so that $Q_1 q_1 Q_2 q_2 \cdots Q_n q_n \mathcal{F}$ is a Quantified Boolean Formula. It is known that checking the truth of such formulas is PSPACE-complete.

Let $\Diamond^i$ ($\Box^i$) denote $\Diamond \cdots \Diamond$ ($\Box \cdots \Box$), with $\Diamond$ ($\Box$) repeated i times respectively. If *clause* is a clause in $\mathcal{F}$ and $q_i, i \geq 2$ occurs as a literal in *clause*, replace q_i with the $\Diamond$*CNF literal* $\Diamond[\Diamond(\neg r \wedge \Diamond^i r) \wedge \Box(r \vee \Box^{i-1} \neg r)]$. Replace occurrences of q_1 with $\Diamond^2(\neg r \wedge \Diamond r)$. Similarly replace occurrences of $\neg q_i, i \geq 2$ and $\neg q_1$ by the $\Box$*clause* literals $\Box[\Box(r \vee \Box^i \neg r) \vee \Diamond(\neg r \wedge \Diamond^{i-1} r)]$ and $\Box^2(r \vee \Box \neg r)$ respectively. Let $clause_1, clause_2, \ldots, clause_m$ be the clauses of $\mathcal{F}$ after the above replacements. Conjunction of the following formulas is equivalent to the formula ψ_A^T defined in [46, Section 3].

$$
\begin{aligned}
init &\triangleq d_0 \wedge \neg d_1 \\
depth &\triangleq \bigwedge_{i=1}^{n+1} \Box^n(\neg d_i \vee d_{i-1}) \\
determined &\triangleq \bigwedge_{i=1}^{n} \Box^n[\neg d_i \vee \neg q_i \vee \Box(\neg d_i \vee q_i)] \wedge \Box^n[\neg d_i \vee q_i \vee \Box(\neg d_i \vee \neg q_i)] \\
branching &\triangleq \bigwedge_{i:Q_{i+1}=\forall} \{\Box^n[\neg d_i \vee d_{i+1} \vee \Diamond(d_{i+1} \wedge \neg d_{i+2} \wedge q_{i+1})] \\
&\qquad \wedge \Box^n[\neg d_i \vee d_{i+1} \vee \Diamond(d_{i+1} \wedge \neg d_{i+2} \wedge \neg q_{i+1})]\} \\
&\wedge \bigwedge_{i:Q_{i+1}=\exists} \Box^n[\neg d_i \vee d_{i+1} \vee \Diamond(d_{i+1} \wedge \neg d_{i+2})] \\
satisfy &\triangleq \bigwedge_{j=1}^{m} \Box^n[\neg d_n \vee clause_j]
\end{aligned}
$$

By replacing $q_1, q_2, \ldots, q_n, d_0, \ldots, d_{n+1}$ by $\Diamond^2(\neg r \wedge \Diamond r), \Diamond[\Diamond(\neg r \wedge \Diamond^2 r) \wedge \Box(r \vee \Box^1 \neg r)], \quad \cdots \quad , \Diamond[\Diamond(\neg r \wedge \Diamond^n r) \wedge \Box(r \vee \Box^{n-1} \neg r)], \Diamond[\Diamond(\neg r \wedge \Diamond^{n+1} r) \wedge \Box(r \vee \Box^n \neg r)], \cdots, \Diamond[\Diamond(\neg r \wedge \Diamond^{2n+2} r) \wedge \Box(r \vee \Box^{2n+1} \neg r)]$ respectively in the above formula, we get a modal CNF formula ψ_{CNF}^T that is satisfiable in a reflexive Kripke model iff $Q_1 q_1 Q_2 q_2 \cdots Q_n q_n \mathcal{F}$ is true [46, Section 3]. Note that r is the only propositional variable used in ψ_{CNF}^T. Hence, if the domain element representing r is removed from $\mathcal{S}(\psi_{CNF}^T)$, the remaining structure is a tree, which has a decomposition with each bag containing at most 2 elements. Adding the element representing r to all bags will give a tree decomposition of $\mathcal{S}(\psi_{CNF}^T)$ of width 2. Therefore, satisfiability of modal CNF formulas of constant treewidth in reflexive models is PSPACE-hard. Hence, unless PTIME = PSPACE, modal satisfiability in reflexive models is not FPT with treewidth alone as parameter. It is proved in [46, Section 3] that a simpler version of ψ_A^T (called ψ_A^K) will work for satisfiability in general models. It is routine to check that using the same procedure as above, a modal CNF formula ψ_{CNF}^K with $\mathcal{S}(\psi_{CNF}^K)$ having constant treewidth can be constructed such that ψ_{CNF}^K is satisfiable iff $Q_1 q_1 Q_2 q_2 \cdots Q_n q_n \mathcal{F}$ is true. Hence, unless PTIME = PSPACE, modal satisfiability in general models is not FPT with treewidth alone as parameter.

2.5 Models with Euclidean Property

In this section, we will investigate the parameterized complexity of satisfiability in Euclidean models. The main observation leading to the FPT algorithm is the fact that if a modal

formula is satisfied in a Euclidean model, then it is satisfied in a rather simple model. As proved in [57], if a modal formula is satisfied at some world w_0 in some Euclidean model $\mathcal{M}$, then it is satisfied in a model whose underlying frame is of the form $(W \cup \{w_0\}, \mapsto)$ where $W \times W \subseteq \mapsto$. The frame looks as illustrated in Fig. 2.3. The worlds w_1, w_2, w_3 serve to satisfy

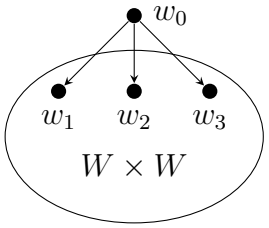


Figure 2.3: Illustration of an Euclidean frame

some $\Diamond CNF$ formulas at w.

We will drop all unary relations $(Lv_i)_{0 \leq i \leq \mathrm{md}(\phi)}$. Instead, we will have one unary relation Pv containing all domain elements representing propositional variables. This will not change the treewidth of $\mathcal{S}(\phi)$. Let $B_\Box$ be a set clauses and Tr be a set of literals of the form $\Diamond CNF$. The following formula checks if there exists a Kripke model with a frame of the form $W \times W$ as shown in Fig. 2.3.

$$\begin{aligned}
\chi(B_\Box, Tr) \triangleq \exists Tr_0 \subseteq Lt \setminus Pv : \\
& B_{\Box 0} = \{cl \in Cl \mid \exists lt \in (Tr_0 \cap \mathcal{B}_\Box) \wedge Oc(lt, cl)\} \Rightarrow \\
& \forall lt \in Tr : \exists Pv_0 \subseteq Pv : D\diamond_0 = \{cl \in Cl \mid Oc(lt, cl)\} \Rightarrow \\
& \quad \forall cl \in D \diamond_0 \cup B_\Box \cup B_{\Box 0} : \\
& \quad [(\exists lt \in Tr_0 \cup Pv_0 : Oc(cl, lt)) \vee (\exists lt \notin Pv_0 : \overline{Oc}(cl, lt))] \\
& \wedge \forall lt \in Tr_0 \cap \mathcal{D}\diamond : \exists Pv_1 \subseteq Pv : D\diamond_1 = \{cl \in Cl \mid Oc(lt, cl)\} \Rightarrow \\
& \quad \forall cl \in D \diamond_1 \cup B_{\Box 0} : \\
& \quad [(\exists lt \in Tr_0 \cup Pv_1 : Oc(cl, lt)) \vee (\exists lt \notin Pv_1 : \overline{Oc}(cl, lt))]
\end{aligned} \tag{2.6}$$

Lemma 2.9. *Suppose ϕ is a modal CNF formula, $B_\Box$ is a set of clauses and Tr is a set of literals of the from $\Diamond CNF$. Then $\chi(B_\Box, Tr)$ is true in $\mathcal{S}(\phi)$ iff there is a Kripke model where all worlds are accessible from one another such that for each $\Diamond CNF$ literal in Tr, there is a world satisfying the corresponding CNF formula and all clauses in $B_\Box$.*

Proof. Suppose $\chi(B_\Box, Tr)$ is true in $\mathcal{S}(\phi)$. Let $\mathcal{M}$ be a Kripke model with one world w_i for each literal $lt_i \in Tr$, where the propositional variables in the corresponding Pv_0 (as witnessed by the truth of $\chi(B_\Box, Tr)$ in $\mathcal{S}(\phi)$) are set to $\top$ and others are set to $\bot$. Add one world w_j for each literal $lt_j \in Tr_0 \cap \mathcal{D}\diamond$ of the form $\Diamond CNF$ in Tr_0, where the propositional variables in the corresponding Pv_1 are set to $\top$ and others are set to $\bot$. By induction on the modal depth of any clause cl, we prove the following claim:

claim: If cl is in $D \diamond_1 \cup B_{\Box 0}$ where $D\diamond_1 = \{cl' \in Cl \mid Oc(lt_j, cl')\}$ for some literal $lt_j \in Tr_0 \cap \mathcal{D}\diamond$, then cl is satisfied at the world w_j. If cl is in $D \diamond_0 \cup B_\Box \cup B_{\Box 0}$ where $D\diamond_0 = \{cl' \in Cl \mid Oc(lt_i, cl')\}$ for some lt_i in Tr, then cl is satisfied in the world w_i.

Let cl be any clause as in the claim. If $\exists lt \in Tr_0$ is true and lt is of the form $\Diamond CNF$, then the clauses in the corresponding CNF have modal depth less than cl and satisfied is some world w_j by induction hypothesis. If lt is of the form $\Box clause$, then the corresponding clause belongs to $B_{\Box 0}$ with modal depth less than cl, so it is satisfied in all worlds by induction hypothesis. Otherwise, cl is satisfied by the valuation of propositional variables in the corresponding world. This proves that $\mathcal{M}$ meets the requirements of the lemma.

For the other direction, suppose there is a Kripke model $\mathcal{M}$ as specified in the lemma. Let Tr_0 be the set of literals of modal depth at least 1 that are satisfied at some world in $\mathcal{M}$. Let $B_{\Box 0} = \{cl \in Cl \mid \exists lt \in (Tr_0 \cap \mathcal{B}_\Box) \wedge Oc(lt, cl)\}$ be the set of clauses such that the corresponding $\Box clause$ is in Tr_0. All clauses in $B_{\Box 0}$ are satisfied in all worlds of $\mathcal{M}$. For any literal $lt \in Tr$, there is some world w in $\mathcal{M}$ satisfying the corresponding CNF formula and all clauses in $B_\Box$. Let $D_{\Diamond 0} = \{cl \in Cl \mid Oc(lt, cl)\}$ be the set of clauses in the corresponding CNF formula and Pv_0 be the set of propositional variables set to $\top$ in w. Since all clauses in $D \Diamond_0 \cup B_\Box \cup B_{\Box 0}$ are satisfied at w, $[(\exists lt \in Tr_0 \cup Pv_0 : Oc(cl, lt)) \vee (\exists lt \notin Pv_0 : \overline{Oc}(cl, lt))]$ is true in $\mathcal{S}(\phi)$. Finally, let $lt \in Tr_0 \cap \mathcal{D}_\Diamond$ be a literal of the form $\Diamond CNF$ in Tr_0. There must be a world w' in $\mathcal{M}$ satisfying the corresponding CNF formula. Let $D_{\Diamond 1} = \{cl \in Cl \mid Oc(lt, cl)\}$ be the set of clauses in the corresponding CNF formula and let Pv_1 be the set of propositional variables set to $\top$ in w'. Since all clauses of $D \Diamond_1 \cup B_{\Box 0}$ are satisfied at w', $[(\exists lt \in Tr_0 \cup Pv_1 : Oc(cl, lt)) \vee (\exists lt \notin Pv_1 : \overline{Oc}(cl, lt))]$ is true in $\mathcal{S}(\phi)$. $\Box$

The following formula makes use of $\chi(B_\Box, Tr)$ to check if a set of clauses Cl_0 is satisfiable in an Euclidean model. Recall the definition of $\zeta(Cl, Tr)$ from Lemma 2.4.

$$\begin{aligned} \chi'(Cl_0) \triangleq \exists Tr_0 \subseteq Lt : \zeta(Cl_0, Tr_0) \\ \wedge \chi(\{cl \in Cl \mid \exists lt \in Tr_0 \cap \mathcal{B}_\Box, Oc(lt, cl)\}, Tr_0 \cap \mathcal{D}_\Diamond) \end{aligned} \tag{2.7}$$

Lemma 2.10. *Let Cl_0 be a set of clauses occurring in a modal CNF formula ϕ. $CNF(Cl_0)$ is satisfiable at a world w_0 in an Euclidean model $\mathcal{M}$ in which w_0 is not its own successor iff $\chi'(Cl_0)$ is true in $\mathcal{S}(\phi)$.*

Proof. Suppose $\chi'(Cl_0)$ is true in $\mathcal{S}(\phi)$. Let $\mathcal{M}$ be a Kripke model with one world w_0 where the propositional variables in Tr_0 are set to $\top$ and others are set to $\bot$. Let $\{cl \in Cl \mid \exists lt \in Tr_0 \cap \mathcal{B}_\Box, Oc(lt, cl)\}$ be the set of clauses such that the corresponding $\Box clause$ is in Tr_0. By Lemma 2.9, there is a Kripke model $\mathcal{M}'$ with all worlds accessible from one another such that for each literal of the form $\Diamond CNF$ in $Tr_0 \cap \mathcal{D}_\Diamond$, there is a world satisfying the corresponding CNF formula and all clauses in $\{cl \in Cl \mid \exists lt \in Tr_0 \cap \mathcal{B}_\Box, Oc(lt, cl)\}$. Make all such worlds successors of w_0 so that $\mathcal{M}$ is an Euclidean model where all literals in Tr_0 are satisfied at w_0. Lemma 2.4 implies that all clauses in Cl_0 are satisfied at w_0.

For the other direction, suppose there is a Euclidean model $\mathcal{M}$ as specified in the lemma. Let Tr_0 be the set of literals satisfied at w_0 that occur as sub-formulas of some clause in Cl_0. Lemma 2.4 implies that $\zeta(Cl_0, Tr_0)$ is true in $\mathcal{S}(\phi)$. Let W be the set of worlds in $\mathcal{M}$ other than w_0 reachable from w_0. Since $\mathcal{M}$ is Euclidean, all worlds in W are accessible from one another. For each $\Diamond CNF$ literal in Tr_0, one of the worlds in W accessible from w_0 satisfies the corresponding CNF formula and all clauses in $\{cl \in Cl \mid \exists lt \in Tr_0 \cap \mathcal{B}_\Box, Oc(lt, cl)\}$. Therefore, Lemma 2.9 implies that $\chi'(\{cl \in Cl \mid \exists lt \in Tr_0 \cap \mathcal{B}_\Box, Oc(lt, cl)\}, Tr_0 \cap \mathcal{D}_\Diamond)$ is true in $\mathcal{S}(\phi)$. $\Box$

Theorem 2.11. *Let ϕ be a modal CNF formula. With treewidth of $\mathcal{S}(\phi)$ as parameter, there is a* FPT *algorithm for checking whether ϕ is satisfiable in a Kripke model that satisfies Euclidean property and any combination of reflexivity, symmetry and transitivity.*

Proof. Let Cl_0 be the set of clauses in ϕ at the top level. To $\mathcal{S}(\phi)$, add a new literal element lt of the form $\Diamond CNF$, and add Oc edges from this new lt element to all clauses Cl_0. This will increase the treewidth of $\mathcal{S}(\phi)$ by only a constant. If ϕ is satisfied at a world w_0 that is its own successor, then Euclidean property will force all worlds to be accessible from one another. Therefore, by Lemma 2.9 and Lemma 2.10, checking satisfiability of ϕ in an Euclidean model is equivalent to checking the truth of $\chi'(Cl_0) \vee \chi(\emptyset, \{lt\})$ in the modified $\mathcal{S}(\phi)$. Addition of reflexivity or symmetry again forces all worlds to be accessible from one another (except in

the case where there is only one world, which can be easily handled by a small MSO formula), so checking satisfiability of ϕ in a Euclidean model that is also reflexive and/or symmetric is equivalent to checking the truth of $\chi(\emptyset, \{lt\})$ in the modified $\mathcal{S}(\phi)$.

If the model is required to be Euclidean and transitive, then, referring to Fig. 2.3, all worlds in W would be accessible from w_0. If a literal $\Box clause$ has to be satisfied at w_0, all worlds in W have to satisfy the corresponding *clause*. This can be easily handled by modifying $\chi(B_\Box, Tr)$ as follows:

$$\begin{aligned}\chi(B_\Box, Tr) \triangleq\ & \exists Tr_0 \subseteq Lt \setminus Pv : \\ & B_{\Box 0} = \{cl \in Cl \mid \exists lt \in (Tr_0 \cap \mathcal{B}_\Box) \wedge Oc(lt, cl)\} \Rightarrow \\ & \forall lt \in Tr \cup (Tr_0 \cap \mathcal{D}_\Diamond) : \exists Pv_0 \subseteq Pv : \\ & \qquad D_{\Diamond 0} = \{cl \in Cl \mid Oc(lt, cl)\} \Rightarrow \forall cl \in D \diamond_0 \cup B_\Box \cup B_{\Box 0} : \\ & \qquad [(\exists lt \in Tr_0 \cup Pv_0 : Oc(cl, lt)) \vee (\exists lt \notin Pv_0 : \overline{Oc}(cl, lt))]\end{aligned} \tag{2.8}$$

All the MSO formulas used above are of constant length. Hence, an application of Courcelle's theorem gives the desired result. □

The Euclidean property is very strong in the sense that it makes the complexity of infinitely many modal logics drop from PSPACE-hard to NP-complete [48]. One might hope for extending the results of this section to any modal logic whose frames is a subset of Euclidean frames. The results in [48] use semantic characterizations while our MSO formulas can only reason about syntax of modal logic formulas. Even though there is a close relation between the syntax and semantics of modal logic of Euclidean frames (which have been used to obtain the results of this section), it seems difficult to exploit this relation to obtain FPT algorithms for arbitrary extensions of modal logic of Euclidean frames. It remains to be seen if other tools from the theory of MSO logic on graphs can be used to achieve this.

2.6 Transitive Models

In transitive models, formulae with small modal depth can check properties of all worlds reachable from a given world. This makes it difficult to obtain FPT algorithms for satisfiability in transitive models. Similar to hierarchies of intractable problems like the polynomial hierarchy in classical complexity theory, there are hierarchies of intractable parameterized problems. The parameterized complexity class W[1] is in one such hierarchy. Parameterized problems that are W[1]-hard are believed not to have FPT algorithms. More details can be found in [37, 26].

To formally prove W[1]-hardness, we will give a parameterized reduction from the Partitioned Weighted Satisfiability (p-PW-SAT) problem to satisfiability of modal CNF formulas in transitive models. The primal graph of a propositional CNF formula consists of one vertex for each propositional variable, and an edge between two variables iff they occur together in a clause. For $e \in \mathbb{N}$, let $[e]$ denote the set $\{1, 2, \ldots, e\}$. The p-PW-SAT problem is defined as follows:

p-PW-SAT	
Instance:	A propositional CNF formula $\mathcal{F}$ over variables $\Phi = \{q_1, \ldots, q_n\}$, a partition $part : \Phi \to [e]$ of the variables and a target function $tg : [e] \to \mathbb{N}$.
Parameter:	Pathwidth of the primal graph of $\mathcal{F}$ and the number of parts e.
Problem:	Decide if there is a satisfying assignment that sets exactly $tg(\rho)$ variables to $\top$ in each part ρ.

Lemma 2.12. *The p-*Pw-Sat *problem is* W[1]*-hard when parameterized by the number of parts e and the pathwidth of the primal graph.*

Proof. We will give a parameterized reduction from the Number List Coloring Problem (Nlcp). An instance of Nlcp is a graph $G = (V, E)$, a set of colors S_v for each vertex $v \in V$ and a target function $tg : \cup_{v \in V} S_v \to \mathbb{N}$. We need to check if G can be properly colored (every adjacent pair of vertices get different colors) such that every vertex v is colored from its set S_v and there are exactly $tg(\ell)$ vertices colored with ℓ for every $\ell \in \cup_{v \in V} S_v$. In [32], it is proved that even for graphs of pathwidth 2, Nlcp is W[1]-hard when parameterized by the total number of colors in $\cup_{v \in V} S_v$.

Given an instance of Nlcp with a graph of pathwidth 2, we associate with it an instance of p-Pw-Sat with the set of propositional variables $\{q_v^\ell \mid v \in V, \ell \in S_v\}$. Every color $\ell \in \cup_{v \in V} S_v$ is a partition of the set of propositional variables and contains the variables $\{q_v^\ell \mid S_v \supseteq \{\ell\}\}$. The target function is the same as the target function of the Nlcp instance. The CNF formula is the conjunction of the following formulae:

$$atLeast \triangleq \bigwedge_{v \in V} \left(\bigvee_{\ell \in S_v} q_v^\ell \right)$$

$$atMost \triangleq \bigwedge_{v \in V} \bigwedge_{\ell \neq \ell' \in S_v} \left(\neg q_v^\ell \vee \neg q_v^{\ell'} \right)$$

$$proper \triangleq \bigwedge_{(v,u) \in E} \bigwedge_{\ell \in S_v \cap S_u} \left(\neg q_v^\ell \vee \neg q_u^\ell \right)$$

Suppose the given Nlcp instance is a Yes instance. In the associated p-Pw-Sat instance, set q_v^ℓ to $\top$ iff the vertex v receives color ℓ in the witnessing coloring. Since every vertex gets a color from its set, the formula *atLeast* above is satisfied. Since every vertex gets at most one color, the formula $atMost$ is satisfied. If (v, u) is any edge in the graph, then since v and u get different colors in the witnessing coloring, the formula *proper* above is also satisfied. Since target function of the p-Pw-Sat instance is same as the target function of the Nlcp instance, the target function of p-Pw-Sat is also satisfied.

On the other hand, suppose that the instance of p-Pw-Sat is a Yes instance. Color a vertex v with the color ℓ iff the propositional variable q_v^ℓ is set to $\top$ in the witnessing satisfying assignment. The formula *atLeast* ensures that every vertex gets at least one color from its set, while the formula $atMost$ ensures that every vertex gets at most one color. If (v, u) is an edge in G and ℓ is a common color between S_v and S_u, then the formula *proper* above ensures that at least one of the vertices v, u do not get the color ℓ. Hence, the coloring given to the graph G is proper. Again since the target function of the p-Pw-Sat instance is same as the target function of the Nlcp instance, the target function of Nlcp is also satisfied.

Now, it is left to prove that the parameters of the p-Pw-Sat instance is bounded by some functions of the parameters of the Nlcp instance. The first parameter of the p-Pw-Sat instance is the number of partitions, which is same as the total number of colors in the Nlcp instance (and later is a parameter of the Nlcp instance). The second parameter is the pathwidth of the primal graph of the CNF formula. Consider any path decomposition of width 2 of the graph G in the Nlcp instance. For every bag B and every vertex v in the bag, replace v by the set $\{q_v^\ell \mid \ell \in S_v\}$. We claim that the resulting decomposition is a path decomposition of the primal graph of the CNF formula in the p-Pw-Sat instance. It is sufficient to prove that for every clause in the CNF formula, there is a bag containing all propositional variables occurring as literals in that clause. For any clause in the formula *atLeast* or $atMost$ associated with a vertex v, any bag that contained the vertex v before

replacement will meet the above criteria. For a clause in the formula *proper* associated with an edge (v, u), any bag that contained the vertices v and u before replacement will suffice. In the new path decomposition, number of elements in any bag is at most 3 times the total number of colors in the NLCP instance. Hence, the pathwidth of the primal graph of the CNF formula in the p-PW-SAT instance is also bounded by a function of the parameters of the NLCP instance. □

Theorem 2.13. *With pathwidth and modal depth as parameters, modal satisfiability in transitive models is* W[1]*-hard.*

The rest of this section is devoted to a proof of the above theorem, which is by a parameterized reduction from p-PW-SAT to satisfiability of modal CNF formulae in transitive models. From Definition 1.1, it is clear that pathwidth is at least as large as treewidth. Hence, any problem that is W[1]-hard when parameterized by pathwidth is also W[1]-hard parameterized by treewidth. Given an instance $(\mathcal{F}, part : \Phi \to [e], tg : [e] \to \mathbb{N})$ of p-PW-SAT problem with the pathwidth of the primal graph of $\mathcal{F}$ being pw, we construct a modal CNF formula $\phi_{\mathcal{F}}$ of modal depth 2 in FPT time such that the pathwidth (and hence the treewidth) of $\mathcal{S}(\phi_{\mathcal{F}})$ is bounded by a function of pw and e and p-PW-SAT is a YES instance iff $\phi_{\mathcal{F}}$ is satisfiable in a transitive model. Suppose the propositional variables used in $\mathcal{F}$ are $q_1, q_2, \ldots, q_n$. The idea is that if $\phi_{\mathcal{F}}$ is satisfied at some world w_0 in some transitive model $\mathcal{M}$, then $\mathcal{M}, w_0 \models \mathcal{F}$. To check that the required targets of the number of variables set to true in each partition are met, $\phi_{\mathcal{F}}$ will force the existence of worlds $w_1, w_2, \ldots, w_n$ arranged as $w_0 \mapsto w_1 \mapsto w_2 \mapsto \cdots \mapsto w_n$. In the formula $\phi_{\mathcal{F}}$, we will maintain a counter for each partition of the propositional variables. At each world w_i, if q_i is true, we will force the counter corresponding to $part(q_i)$ to increment. At the world w_n, the counters will have the number of variables set to $\top$ in each partition. We will then verify in the formula $\phi_{\mathcal{F}}$ that these counts meet the given target. Such counting tricks are standard in complexity theoretic arguments of modal logics, for example [6, Section 6.8]. The challenge here is to implement the counting in a modal formula of small pathwidth.

In a p-PW-SAT instance containing n propositional variables and e partitions, we will denote the number of variables in partition ρ by $n[\rho]$. We first construct an optimal path decomposition of the primal graph of $\mathcal{F}$ in FPT time. We will name the variables occurring in the first bag as $q_1, \ldots, q_i$. We will name the variables newly introduced in the second bag as $q_{i+1}, \ldots, q_{i'}$ and so on. In the rest of the construction, we will use this same ordering $q_1, \ldots, q_n$ of the propositional variables. This will be important to maintain the pathwidth of the resulting modal formula low. The modal CNF formula $\phi_{\mathcal{F}}$ will use all the propositional variables $q_1, \ldots, q_n$ used by $\mathcal{F}$ and also use the following additional variables:

- $t_{\uparrow 1}, \ldots, t_{\uparrow e}, f_{\uparrow 1}, \ldots, f_{\uparrow e}$: partition indicators.
- For each partition ρ, $tr^0_\rho, \ldots, tr^{n[\rho]}_\rho, fl^0_\rho, \ldots, fl^{n[\rho]}_\rho$: counters to count the number of variables set to $\top$ and $\bot$ in partition ρ.
- $d_0, \ldots, d_{n+1}$: depth indicators.

The modal CNF formula $\phi_{\mathcal{F}}$ is the conjunction of the formulae described below. For clarity, we have used the implication symbol $\Rightarrow$ but it can be easily converted to CNF. Also for notational convenience, we will use $part(i)$ instead of $part(q_i)$. $\Phi(\rho)$ is the set of variables among $\{q_1, \ldots, q_n\}$ in partition ρ. The formula *determined* ensures that all successors of w_0 preserve the assignment of $q_1, \ldots, q_n$. The formula *depth* ensures that for all i, $d_i \wedge \neg d_{i+1}$ holds in the world w_i.

In w_{i-1}, if q_i is set to $\top$, we want to indicate that in w_i, the counter for partition $part(i)$ should be incremented. We will indicate this in the formula *setCounter* by setting the

variable $t_{\uparrow part(i)}$ to $\top$. Similar indication is done for the counter keeping track of variables set to $\bot$ in partition ρ.

$$determined \triangleq \bigwedge_{i=1}^{n} q_i \Rightarrow \Box q_i \wedge \bigwedge_{i=1}^{n} \neg q_i \Rightarrow \Box \neg q_i$$

$$depth \triangleq \Diamond(d_1 \wedge \neg d_2) \wedge \bigwedge_{i=1}^{n-1} \Box\left[(d_i \wedge \neg d_{i+1}) \Rightarrow \Diamond(d_{i+1} \wedge \neg d_{i+2})\right]$$

$$setCounter \triangleq (q_1 \Rightarrow t_{\uparrow part(1)}) \wedge (\neg q_1 \Rightarrow f_{\uparrow part(1)}) \wedge \bigwedge_{i=2}^{n} \Box\left\{[d_{i-1} \wedge \neg d_i] \Rightarrow [(q_i \Rightarrow t_{\uparrow part(i)}) \wedge (\neg q_i \Rightarrow f_{\uparrow part(i)})]\right\}$$

$$incCounter \triangleq (t_{\uparrow part(1)} \Rightarrow \Box tr^1_{part(1)}) \wedge (f_{\uparrow part(1)} \Rightarrow \Box fl^1_{part(1)}) \wedge \bigwedge_{\rho=1}^{e} \bigwedge_{j=0}^{n[\rho]-1} \Box[t_{\uparrow\rho} \Rightarrow (tr^j_\rho \Rightarrow \Box tr^{j+1}_\rho)] \wedge \Box[f_{\uparrow\rho} \Rightarrow (fl^j_\rho \Rightarrow \Box fl^{j+1}_\rho)]$$

$$targetMet \triangleq \bigwedge_{\rho=1}^{e} \Box[d_n \Rightarrow (tr^{tg(\rho)}_\rho \wedge \neg tr^{tg(\rho)+1}_\rho)] \wedge \bigwedge_{\rho=1}^{e} \Box[d_n \Rightarrow (fl^{n[\rho]-tg(\rho)}_\rho \wedge \neg fl^{n[\rho]-tg(\rho)+1}_\rho)]$$

Variables $tr^0_\rho, \ldots, tr^{n[\rho]}_\rho$ implement the counter keeping track of variables set to $\top$ in partition ρ. If j variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to $\top$, then we want tr^j_ρ to be set to $\top$ in w_i. To maintain this, in w_{i-1}, if it is indicated that a counter is to be incremented (by setting $t_{\uparrow\rho}$ to $\top$), we will force all successors of w_{i-1} to increment the tr_ρ counter in the formula $incCounter$. Finally, we check that at w_n, all the targets are met in the formula $targetMet$.

The modal CNF formula $\phi_{\mathcal{F}}$ we need is the conjunction of $\mathcal{F}$, the formulae defined above and the miscellaneous formulae below (which ensure that counters are initiated properly and are monotonically non-decreasing).

$$determined' \triangleq \bigwedge_{\rho=1}^{e} tr^0_\rho \Rightarrow \Box tr^0_\rho \wedge \bigwedge_{\rho=1}^{e} fl^0_\rho \Rightarrow \Box fl^0_\rho$$

$$countInit \triangleq d_0 \wedge \neg d_1 \wedge \bigwedge_{\rho=1}^{e} (\neg tr^1_\rho \wedge \neg fl^1_\rho \wedge tr^0_\rho \wedge fl^0_\rho)$$

$$depth' \triangleq \bigwedge_{\rho=1}^{e} \bigwedge_{j=0}^{n[\rho]} \left[\Box(tr^j_\rho \Rightarrow \Box tr^j_\rho) \wedge \Box(fl^j_\rho \Rightarrow \Box fl^j_\rho)\right]$$

$$countMonotone \triangleq \bigwedge_{i=1}^{n} \Box(d_i \Rightarrow d_{i-1}) \wedge \bigwedge_{\rho=1}^{e} \bigwedge_{j=2}^{n[\rho]} \left[\Box(tr^j_\rho \Rightarrow tr^{j-1}_\rho) \wedge \Box(fl^j_\rho \Rightarrow fl^{j-1}_\rho)\right]$$

Lemma 2.14. *If a p-*Pw-Sat *instance is a* Yes *instance, then the modal formula constructed above is satisfied in a transitive Kripke model.*

Proof. We will construct a transitive Kripke model using the satisfying assignment f that satisfies $\mathcal{F}$ while meeting the given target. The model $\mathcal{M}$ consists of worlds $w_0, w_1, \ldots, w_n$ arranged as $w_0 \mapsto w_1 \mapsto w_2 \mapsto \cdots \mapsto w_n$. In all worlds, q_i is set to $f(q_i)$ for all i, thus ensuring that $\mathcal{M}, w_0 \models \mathcal{F} \wedge determined$. In w_i, $\{d_0, \ldots, d_i\}$ are set to $\top$ and $\{d_{i+1}, \ldots, d_{n+1}\}$ are set

to $\bot$ for all i between 0 and n, thus ensuring that $\mathcal{M}, w_0 \models \mathit{depth} \wedge d_0 \wedge \neg d_1$, the last two clauses coming from the formula *countInit*. It also ensures that $\mathcal{M}, w_0 \models \bigwedge_{i=1}^{n} \Box(d_i \Rightarrow d_{i-1})$, which is part of *countMonotone*. We will set tr_ρ^0 and fl_ρ^0 to $\top$ in all worlds and tr_ρ^1 and fl_ρ^1 to $\bot$ in w_0 for all partitions ρ, thus ensuring $\mathcal{M}, w_0 \models \mathit{countInit} \wedge \mathit{determined}'$. At w_{i-1}, we will set $t_{\uparrow part(i)}$ to q_i's value in the same world and $f_{\uparrow part(i)}$ to $\neg q_i$'s value. This will ensure that $\mathcal{M}, w_0 \models \mathit{setCounter}$.

At w_i, for any partition ρ, if j variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to $\top$, then we will set $\{tr_\rho^0, \ldots, tr_\rho^j\}$ to $\top$ and $\{tr_\rho^{j+1}, \ldots, tr_\rho^{n[\rho]}\}$ to $\bot$. If j' variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to $\bot$, we will set $\{\mathit{fl}_\rho^0, \ldots, \mathit{fl}_\rho^{j'}\}$ to true and $\{\mathit{fl}_\rho^{j'+1}, \ldots, \mathit{fl}_\rho^{n[\rho]}\}$ to $\bot$. For any $\rho \neq \mathit{part}(i+1)$, we will set $t_{\uparrow\rho}$ and $f_{\uparrow\rho}$ to $\bot$ at w_i. These will ensure that $\mathcal{M}, w_0 \models \mathit{incCounter} \wedge \mathit{depth}' \wedge \mathit{countMonotone}$.

Combined with the above settings of all propositional variables in $\mathcal{M}$, it is easy to check that the fact that f meets the target for each partition implies $\mathcal{M}, w_0 \models \mathit{targetMet}$. $\Box$

Lemma 2.15. *Suppose the modal CNF formula $\phi_{\mathcal{F}}$ constructed above is satisfied at some world w_0 of some transitive Kripke model $\mathcal{M}$. Then $\mathcal{M}$ contains distinct worlds $w_1, \ldots, w_n$ such that for each i between 1 and n, w_i is a successor of w_{i-1}. Moreover, $\{d_0, \ldots, d_i\}$ are set to $\top$ and $\{d_{i+1}, \ldots, d_{n+1}\}$ are set to $\bot$ in w_i. For any partition ρ, if j variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to $\top$ in w_0, then $\{tr_\rho^0, \ldots, tr_\rho^j\}$ are all set to $\top$ in w_i. If j' variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to $\bot$ in w_0, then $\{\mathit{fl}_\rho^0, \ldots, \mathit{fl}_\rho^{j'}\}$ are all set to $\top$ in w_i.*

Proof. We will first prove the existence of worlds $w_1, \ldots, w_i$ by induction on i.

Base case $i = 1$: Since $\mathcal{M}, w_0 \models \mathit{depth}$, there must be a successor w_1 of w_0 that satisfies $d_1 \wedge \neg d_2$. Since $\mathcal{M}, w_0 \models \mathit{countInit}$, w_0 satisfies $d_0 \wedge \neg d_1$ and hence w_1 can not be same as w_0. Since $\mathcal{M}, w_0 \models \Box(d_3 \Rightarrow d_2)$ (part of *countMonotone*) and w_1 is a successor of w_0, we get $\mathcal{M}, w_1 \models d_3 \Rightarrow d_2$. Since d_2 is set to $\bot$ in w_1, this means that d_3 is also set to $\bot$ in w_1. Similar reasoning can be used to prove that all of $\{d_2, \ldots, d_{n+1}\}$ are set to $\bot$ in w_1. The fact that $\mathcal{M}, w_1 \models d_1 \Rightarrow d_0$ means that d_0 is set to $\top$ in w_1 (since d_1 is set to $\top$ in w_1).

Induction step: Assume that worlds $w_1, \ldots, w_i$ exist in $\mathcal{M}$ with the stated properties. Hence, w_i satisfies $d_i \wedge \neg d_{i+1}$. Since w_0 satisfies *depth* and w_i is a successor of w_0 (by transitivity), there must be a successor w_{i+1} of w_i that satisfies $d_{i+1} \wedge \neg d_{i+2}$. Since all worlds $w_0, \ldots, w_i$ satisfy $\neg d_{i+1}$, w_{i+1} is distinct from all of them. The fact that w_{i+1} satisfies $d_{i'} \Rightarrow d_{i'-1}$ for all i' (these formulae are part of *countMonotone* formula satisfied by w_0) can be used to show that all of $d_0, \ldots, d_{i+1}$ are set to $\top$ in w_{i+1} and all of $d_{i+2}, \ldots, d_{n+1}$ are set to $\bot$ in w_{i+1}.

We will now prove the second claim of the lemma, which is about values of $\{tr_\rho^0, \ldots, tr_\rho^j\}$ in w_i. We will first prove that tr_ρ^j is set to $\top$ by induction on i.

Base case $i = 1$: If q_1 is not in part ρ, there is nothing to prove ($tr_{part(1)}^0$ is set to $\top$ in all worlds). If q_1 is in part ρ and q_1 is set to $\bot$, there is nothing to prove. If q_1 is in part ρ and q_1 is set to $\top$, then since w_0 satisfies *setCounter*, we get $\mathcal{M}, w_0 \models q_1 \Rightarrow t_{\uparrow part(1)}$. Since, q_1 is set to $\top$ and $\mathit{part}(1) = \rho$, we get that $t_{\uparrow\rho}$ is set to $\top$ in w_0. Since w_0 satisfies *incCounter*, we get $\mathcal{M}, w_0 \models t_{\uparrow\rho} \Rightarrow \Box tr_\rho^1$ and hence $\mathcal{M}, w_0 \models \Box tr_\rho^1$. Since w_1 is a successor of w_0, we conclude that in w_1, tr_ρ^1 is set to $\top$.

Induction step: Case 1: q_i is not in part ρ and none of the variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to to $\top$ in w_i. In this case, there is nothing to prove.

Case 2: q_i is not in part ρ and some $1 \leq j < i$ variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to $\top$. By the induction hypothesis, tr_ρ^j is set to $\top$ in w_{i-1}. Now $\mathcal{M}, w_0 \models \mathit{depth}'$. Hence $\mathcal{M}, w_0 \models \Box(tr_\rho^j \Rightarrow \Box tr_\rho^j)$, and hence $\mathcal{M}, w_{i-1} \models tr_\rho^j \Rightarrow \Box tr_\rho^j$ (since w_{i-1} is a successor of w_0), and hence $\mathcal{M}, w_{i-1} \models \Box tr_\rho^j$ (since tr_ρ^j is set to $\top$ in w_{i-1}), and hence $\mathcal{M}, w_i \models tr_\rho^j$ (since w_i is a successor of w_{i-1}).

Case 3: q_i is in part ρ and q_i is set to $\bot$. If none of the variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to $\top$, then the argument is similar to case 1. If some $1 \leq j < i$ variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to $\top$, then the argument is similar to case 2.

Case 4: q_i is in part ρ and q_i is set to $\top$. We know that w_{i-1} satisfies $d_{i-1} \wedge \neg d_i$. Since w_0 satisfies *setCounter*, we have $\mathcal{M}, w_0 \models \Box\left\{[d_{i-1} \wedge \neg d_i] \Rightarrow [q_i \Rightarrow t_{\uparrow_{part(i)}}]\right\}$, and hence $\mathcal{M}, w_{i-1} \models [d_{i-1} \wedge \neg d_i] \Rightarrow [q_i \Rightarrow t_{\uparrow_{part(i)}}]$ (since w_{i-1} is a successor of w_0), and hence $\mathcal{M}, w_{i-1} \models q_i \Rightarrow t_{\uparrow_\rho}$ (since $\mathcal{M}, w_{i-1} \models d_{i-1} \wedge \neg d_i$), and hence $\mathcal{M}, w_{i-1} \models t_{\uparrow_\rho}$ (since $\mathcal{M}, w_{i-1} \models q_i$). Since w_0 satisfies *incCounter* and w_{i-1} is a successor of w_0, we get $\mathcal{M}, w_{i-1} \models t_{\uparrow_\rho} \Rightarrow (tr_\rho^{j-1} \Rightarrow \Box tr_\rho^j)$. We have already seen that $t_{\uparrow_\rho}$ is set to $\top$ in w_{i-1} and tr_ρ^{j-1} is set to $\top$ in w_{i-1} by the induction hypothesis (j is at least 1 since q_i is in part ρ and is set to $\top$). Hence, we get $\mathcal{M}, w_{i-1} \models \Box tr_\rho^j$. Since w_i is a successor of w_{i-1}, we conclude that tr_ρ^j is set to $\top$ in w_i.

Now, since w_0 satisfies $\Box(tr_\rho^j \Rightarrow tr_\rho^{j-1})$ (this is part of *countMonotone*) and w_i is a successor of w_0, we get $\mathcal{M}, w_i \models tr_\rho^j \Rightarrow tr_\rho^{j-1}$. Since tr_ρ^j is set to $\top$ in w_i, it follows that tr_ρ^{j-1} is also set to $\top$ in w_i. Similarly, $tr_\rho^0, \ldots, tr_\rho^j$ are all set to $\top$ in w_i.

The proof for values of $\{fl_\rho^0, \ldots, fl_\rho^{j'}\}$ is symmetric to the proof of values of $\{tr_\rho^0, \ldots, tr_\rho^j\}$. □

Theorem 2.16. *If $\phi_\mathcal{F}$ constructed above is satisfied in a transitive model, then the* p-Pw-Sat *instance is a* Yes *instance.*

Proof. Suppose $\phi_\mathcal{F}$ is satisfied in some world w_0 of a transitive model. Since $\mathcal{F}$ is part of $\phi_\mathcal{F}$, the assignment to $\{q_1, \ldots, q_n\}$ induced by w_0 satisfies $\mathcal{F}$. We claim that this assignment also meets the targets. If not, we will derive a contradiction. For some partition ρ, suppose there are more than $tg(\rho)$ variables set to $\top$. Then by Lemma 2.15, $tr_\rho^{tg(\rho)+1}$ will be set to $\top$ in w_n, contradicting the fact that w_0 satisfies *targetMet*. For some partition ρ, if there are less than $tg(\rho)$ variables set to $\top$, then there will be more than $n[\rho] - tg(\rho)$ variables set to $\bot$. By Lemma 2.15, $fl_\rho^{n[\rho]-tg(\rho)+1}$ will be set to $\top$ in w_n, again contradicting the fact that w_0 satisfies *targetMet*. □

Given an instance of p-Pw-Sat problem, the formula $\phi_\mathcal{F}$ described above can be constructed in Fpt time. To complete the proof of Theorem 2.13, we will prove that the pathwidth of $\phi_\mathcal{F}$ is bounded by some function of e and pw. $\phi_\mathcal{F}$ has been carefully constructed to keep pathwidth low.

Lemma 2.17. *The pathwidth of $\mathcal{S}(\phi_\mathcal{F})$ is at most $4pw + 2e + 5$.*

Proof. Given an optimal path decomposition of the primal graph of $\mathcal{F}$, depth counters can be added to the bags without increasing their size much since the order of depth counters is same as the order of $q_1, \ldots, q_n$. There are only $2e$ partition indicators $t_{\uparrow_1}, \ldots, t_{\uparrow_e}, f_{\uparrow_1}, \ldots, f_{\uparrow_e}$, so they can also be added to the bags without increasing their size very much. However, the set of $2n$ partition counters (of the form tr_ρ^j or fl_ρ^j) has to be added carefully to maintain the size of the bags. Formulas of $\phi_\mathcal{F}$ have been carefully designed to enable this. The key observation is that the only "link" between $q_1, \ldots, q_n$ and partition counters are partition indicators and there are only $2e$ of them. The following proof relies on this observation.

Consider an optimal path decomposition of the primal graph of $\mathcal{F}$ with each bag containing at most pw elements. Ensure that for all i with $1 \leq i < n$, there is a bag containing both q_i and q_{i+1} or there is a bag with q_i such that the next bag contains q_{i+1} (call this the *continuity* property). If this is not the case for some i, consider the last bag B containing q_i and the first bag B' containing q_{i+1}. No bag that is between B and B' will introduce any new variable (if it did, that new variable would have been q_{i+1} according to our order). Hence, all the bags in between B and B' are subsets of B. Hence, they can all be removed and B'

can become the bag immediately after B. The resulting decomposition is still a path decomposition of the primal graph of $\mathcal{F}$ with each bag containing at most pw elements. Moreover, the order of variables $q_1, \ldots, q_n$ does not change due to the change we have made in the path decomposition. This new decomposition has a bag containing q_i such that the next bag contains q_{i+1}. Now, we can repeat the above process until we get a path decomposition with the continuity property.

For any i with $1 \leq i \leq n$, let B_i be a bag containing the propositional variable q_i. We will expand this path decomposition by adding variables used in $\phi_{\mathcal{F}}$ such that for every *clause* that appears in $\phi_{\mathcal{F}}$, there is a bag that contains all propositional variables appearing in that clause. Each of these expanded bags will have at most $4pw + 2e$ elements. We will then show how to expand this into a path decomposition of $\mathcal{S}(\phi_{\mathcal{F}})$, by adding at most 6 elements to each bag (creating duplicate copies of existing bags if required). This will prove that the pathwidth of $\mathcal{S}(\phi_{\mathcal{F}})$ is at most $4pw + 2e + 5$.

First, in each bag B and each element q_i in it, add d_{i-1}, d_i and d_{i+1}. Note that due to continuity property of the decomposition we started with, the expanded decomposition still retains the property that all bags containing an element forms a connected component, even after adding depth counters $d_0, \ldots, d_{n+1}$. Next, add $t_{\uparrow 1}, \ldots, t_{\uparrow e}, f_{\uparrow 1}, \ldots, f_{\uparrow e}$ to all the bags. We will refer to the bag containing q_i, d_{i-1}, d_i and d_{i+1} as B_i. Now, we have a decomposition with each bag containing at most $4pw + 2e$ elements, and the last bag contains d_n. To this bag, we will append $2e$ paths serially. For $1 \leq \rho \leq e$, $(2\rho - 1)^{\text{th}}$ path will be as follows: $\{d_n, t_{\uparrow 1}, \ldots, t_{\uparrow e}, f_{\uparrow 1}, \ldots, f_{\uparrow e}, tr_\rho^0, tr_\rho^1\} - \{d_n, t_{\uparrow 1}, \ldots, t_{\uparrow e}, f_{\uparrow 1}, \ldots, f_{\uparrow e}, tr_\rho^1, tr_\rho^2\} - \cdots - \{d_n, t_{\uparrow 1}, \ldots, t_{\uparrow e}, f_{\uparrow 1}, \ldots, f_{\uparrow e}, tr_\rho^{n[\rho]-1}, tr_\rho^{n[\rho]}\}$. We will refer to these bags as $B_\rho^1, \ldots, B_\rho^{n[\rho]}$. $2\rho^{\text{th}}$ path is similar, with fl_ρ variables replacing tr_ρ variables. We will refer to these bags in $2\rho^{\text{th}}$ path as $B_\rho^{1'}, \ldots, B_\rho^{n[\rho]'}$. Each of these new bags has at most $2e + 3$ elements, and the whole decomposition still retains the property that for any element, the set of bags containing that element forms a connected component.

Now we will show how to expand the above decomposition into a path decomposition of $\mathcal{S}(\phi_{\mathcal{F}})$. We have to add clauses and literals occurring in $\phi_{\mathcal{F}}$ and ensure that for any pair of elements $Oc(e_1, e_2)$ or $\overline{Oc}(e_1, e_2)$, there is a bag containing both e_1 and e_2. To achieve this, we may have to "augment" an existing bag with new elements. If B_i is a bag in the path decomposition $\cdots - B_i - \ldots$, augmenting B_i with elements e_1 and e_2 means that we add another bag $\cdots - B_i' - B_i - \ldots$ with B_i' containing all elements of B_i and in addition containing e_1 and e_2. If we ensure that these new elements introduced during augmentation is never added to any other bag in the decomposition, augmentation will not violate the path decomposition's property that for any element, the set of bags containing that element forms a connected component. Now, we will go through each sub-formula of $\phi_{\mathcal{F}}$ and prove that all its clauses, literals and Oc pairs are already represented in the path decomposition we have constructed above or that the decomposition can be augmented to represent them.

- Clauses in $\mathcal{F}$: For each clause in $\mathcal{F}$, the propositional variables in that clause form a clique in the primal graph of $\mathcal{F}$. Hence, there is a bag B in the new decomposition that contains all propositional variables occurring in that clause. Augment B with a new domain element representing the clause.

- *determined*: Here, the clauses are of the form $\neg q_i \vee \Box q_i$ and $q_i \vee \Box \neg q_i$. Augment the bag B_i containing q_i with 3 domain elements, one for the clause $\neg q_i \vee \Box q_i$ itself, one for the literal $\Box q_i$ and one for the clause in this literal that contains q_i as its only literal. Perform similar augmentation for the clause $q_i \vee \Box \neg q_i$.

- *depth*: For $\Diamond(d_1 \wedge \neg d_2)$, augment the bag B_1 containing d_1 and d_2 with 4 domain elements representing literals and clauses of $\Diamond(d_1 \wedge \neg d_2)$. Augment the bag B_{i+1} con-

taining d_i, d_{i+1} and d_{i+2} with 6 elements representing literals and clauses of $\Box[\neg d_i \vee d_{i+1} \vee \Diamond(d_{i+1} \wedge \neg d_{i+2})]$.

- *setCounter*: Augment the bag B_1 containing q_1 and $t_{\uparrow part(1)}$ with one element representing the clause $\neg q_1 \vee t_{\uparrow part(1)}$. Do a similar augmentation for the clause $q_1 \vee f_{\uparrow part(1)}$. $\Box(q \wedge r)$ is equivalent to $\Box q \wedge \Box r$. Hence, the latter part of *setCounter* can be split into clauses $\Box(\neg d_{i-1} \vee d_i \vee \neg q_i \vee t_{\uparrow part(i)})$ and $\Box(\neg d_{i-1} \vee d_i \vee q_i \vee f_{\uparrow part(i)})$. Augment the bag B_i containing $d_{i-1}, d_i, q_i, t_{\uparrow part(i)}$ and $f_{\uparrow part(i)}$ with 6 elements representing clauses and literals of these two clauses.

- *incCounter*: Augment the bag $B^1_{part(1)}$ containing $t_{\uparrow part(1)}$ and $tr^1_{part(1)}$ with 3 elements representing clauses and literals of $(\neg t_{\uparrow part(1)} \vee \Box tr^1_{part(1)})$. Similarly augment the bag $B^{1'}_{part(1)}$ for $(\neg f_{\uparrow part(1)} \vee \Box fl^1_{part(1)})$. Augment the bag B^{j+1}_ρ containing $t_{\uparrow \rho}, tr^j_\rho$ and tr^{j+1}_ρ with 6 elements representing literals and clauses of $\Box(\neg t_{\uparrow \rho} \vee \neg tr^j_\rho \vee \Box tr^{j+1}_\rho)$. Similarly augment $B^{j+1'}_\rho$ for $\Box(\neg f_{\uparrow \rho} \vee \neg fl^j_\rho \vee \Box fl^{j+1}_\rho)$.

- *targetMet*: Augment the bag $B^{tg(\rho)+1}_\rho$ containing $d_n, tr^{tg(\rho)}_\rho$ and $tr^{tg(\rho)+1}_\rho$ with 6 elements for the literals and clauses in $\Box(\neg d_n \vee tr^{tg(\rho)}_\rho)$ and $\Box(\neg d_n \vee \neg tr^{tg(\rho)+1}_\rho)$. Similarly augment $B^{n[\rho]-tg(\rho)+1'}_\rho$ for $\Box(\neg d_n \vee fl^{n[\rho]-tg(\rho)}_\rho)$ and $\Box(\neg d_n \vee \neg fl^{n[\rho]-tg(\rho)+1}_\rho)$

- *determined'*: Augment the bag B^1_ρ containing tr^0_ρ with 3 elements representing literals and clauses of $\neg tr^0_\rho \vee \Box tr^0_\rho$. Similarly augment $B^{1'}_\rho$ for $\neg fl^0_\rho \vee \Box fl^0_\rho$.

- *countInit*: Augment the bag B_1 containing d_0 and d_1 with 2 elements representing the clauses in $d_0 \wedge \neg d_1$. Augment the bag B^1_ρ containing tr^0_ρ and tr^1_ρ with 2 elements representing the clauses in $\neg tr^1_\rho \wedge tr^0_\rho$. Similarly augment $B^{1'}_\rho$ for $\neg fl^1_\rho \wedge fl^0_\rho$.

- *depth'*: Augment the bag B^j_ρ containing tr^j_ρ with 6 elements representing literals and clauses of $\Box(\neg tr^j_\rho \vee \Box tr^j_\rho)$. Similarly augment $B^{j'}_\rho$ for $\Box(\neg fl^j_\rho \vee \Box fl^j_\rho)$.

- *countMonotone*: Augment the bag B_i containing d_i and d_{i-1} with 3 elements representing literals and clauses of $\Box(\neg d_i \vee d_{i-1})$. Augment the bag B^j_ρ containing tr^j_ρ and tr^{j-1}_ρ with 3 elements representing literals and clauses of $\Box(\neg tr^j_\rho \vee tr^{j-1}_\rho)$. Similarly augment $B^{j'}_\rho$ for $\Box(\neg fl^j_\rho \vee fl^{j-1}_\rho)$.

□

In the absence of transitivity, the above reduction would require a formula of modal depth that depends on n (and hence it would no longer be a parameterized reduction). The above hardness proof will however go through for any class of transitive frames that has paths of unbounded length of the form $w_1 \mapsto w_2 \mapsto \cdots \mapsto w_n$ without any reverse paths. See [91] for some context on such classes of transitive frames of unbounded depth.

2.7 Temporal Logics

In this section, we prove W[1]-hardness of satisfiability of a CNF fragment of LTL and CTL given below, with treewidth and modality depth as parameters. The proof is very similar to the one in the previous section. We give a parameterized reduction from p-Pw-Sat to satisfiability of LTL/CTL formulas. The formulas required are obtained by the modal formulas given in the previous section by replacing $\Diamond$ with X and $\Box$ with G in case of LTL and $\Diamond$ with EX and $\Box$ with AG in case of CTL. We will consider the following fragment of

LTL and associate treewidth to a LTL formula of this form in the same way we associated treewidth to modal logic formulas in CNF.

$$\begin{aligned} \mathit{literal} &::= q \mid \neg q \mid G\mathit{clause} \mid X\,\mathit{CNF} \\ \mathit{clause} &::= \mathit{literal} \mid \mathit{clause} \vee \mathit{clause} \mid \bot \\ \mathit{CNF} &::= \mathit{clause} \mid \mathit{CNF} \wedge \mathit{CNF} \end{aligned}$$

For LTL formulas of the above Conjunctive Normal Form, we associate modality depth in the same way we associate modal depth to modal logic formulas, replacing $\Diamond$ with X and $\Box$ with G.

We will also consider the following fragment of CTL and associate treewidth to a CTL formula of this form in the same way we associated treewidth to modal formulas in CNF.

$$\begin{aligned} \mathit{literal} &::= q \mid \neg q \mid AG\mathit{clause} \mid EX\,\mathit{CNF} \\ \mathit{clause} &::= \mathit{literal} \mid \mathit{clause} \vee \mathit{clause} \mid \bot \\ \mathit{CNF} &::= \mathit{clause} \mid \mathit{CNF} \wedge \mathit{CNF} \end{aligned}$$

For CTL formulas of the above Conjunctive Normal Form, we associate modality depth in the same way we associate modal depth to modal logic formulas, replacing $\Diamond$ with EX and $\Box$ with AG.

Theorem 2.18. *With pathwidth and modality depth as parameters, satisfiability of the CNF fragment of LTL/CTL given above is* W[1]*-hard.*

The rest of this section is devoted to a proof of the above theorem. Given an instance $(\mathcal{F}, \mathit{part} : \Phi \to [e], tg : [e] \to \mathbb{N})$ of p-Pw-Sat problem, compute an optimal path decomposition of the primal graph of $\mathcal{F}$ and order the propositional variables as $q_1, \ldots, q_n$ as described in the beginning of section 2.6. Consider LTL formulas in (2.9) and the CTL formulas in (2.10).

Lemma 2.19. *If a p-*Pw-Sat *instance is a* Yes *instance, then the LTL formula that is the conjunction of $\mathcal{F}$ and formulas in* (2.9) *is satisfiable, and so is the CTL formula that is the conjunction of $\mathcal{F}$ and formulas in* (2.10)*.*

Proof. It is routine to verify that the model $w_0 \mapsto w_1 \mapsto \cdots w_n$ constructed in Lemma 2.14 satisfies the LTL and CTL formulas. □

$$
\begin{aligned}
determined &\triangleq \bigwedge_{i=1}^{n} q_i \Rightarrow Gq_i \wedge \bigwedge_{i=1}^{n} \neg q_i \Rightarrow G\neg q_i \\
depth &\triangleq X(d_1 \wedge \neg d_2) \wedge \bigwedge_{i=1}^{n-1} G\left[(d_i \wedge \neg d_{i+1}) \ \Rightarrow X(d_{i+1} \wedge \neg d_{i+2})\right] \\
setCounter &\triangleq (q_1 \Rightarrow t_{\uparrow part(1)}) \wedge (\neg q_1 \Rightarrow f_{\uparrow part(1)}) \\
&\quad \wedge \bigwedge_{i=2}^{n} G\left\{[d_{i-1} \wedge \neg d_i] \Rightarrow [(q_i \Rightarrow t_{\uparrow part(i)}) \wedge (\neg q_i \Rightarrow f_{\uparrow part(i)})]\right\} \\
incCounter &\triangleq (t_{\uparrow part(1)} \Rightarrow Gtr^1_{part(1)}) \wedge (f_{\uparrow part(1)} \Rightarrow Gfl^1_{part(1)}) \\
&\quad \wedge \bigwedge_{\rho=1}^{e} \bigwedge_{j=0}^{n[\rho]-1} G[t_{\uparrow\rho} \Rightarrow (tr^j_\rho \Rightarrow Gtr^{j+1}_\rho)] \wedge G[f_{\uparrow\rho} \Rightarrow (fl^j_\rho \Rightarrow Gfl^{j+1}_\rho)] \\
targetMet &\triangleq \bigwedge_{\rho=1}^{e} G[d_n \Rightarrow (tr^{tg(\rho)}_\rho \wedge \neg tr^{tg(\rho)+1}_\rho)] \qquad (2.9) \\
&\quad \wedge \bigwedge_{\rho=1}^{e} G[d_n \Rightarrow (fl^{n[\rho]-tg(\rho)}_\rho \wedge \neg fl^{n[\rho]-tg(\rho)+1}_\rho)] \\
determined' &\triangleq \bigwedge_{\rho=1}^{e} tr^0_\rho \Rightarrow Gtr^0_\rho \wedge \bigwedge_{\rho=1}^{e} fl^0_\rho \Rightarrow Gfl^0_\rho \\
countInit &\triangleq d_0 \wedge \neg d_1 \wedge \bigwedge_{\rho=1}^{e} (\neg tr^1_\rho \wedge \neg fl^1_\rho \wedge tr^0_\rho \wedge fl^0_\rho) \\
depth' &\triangleq \bigwedge_{\rho=1}^{e} \bigwedge_{j=0}^{n[\rho]} \left[G(tr^j_\rho \Rightarrow Gtr^j_\rho) \wedge G(fl^j_\rho \Rightarrow Gfl^j_\rho)\right] \\
countMonotone &\triangleq \bigwedge_{i=1}^{n} G(d_i \Rightarrow d_{i-1}) \wedge \bigwedge_{\rho=1}^{e} \bigwedge_{j=2}^{n[\rho]} \left[G(tr^j_\rho \Rightarrow tr^{j-1}_\rho) \wedge G(fl^j_\rho \Rightarrow fl^{j-1}_\rho)\right]
\end{aligned}
$$

Lemma 2.20. *Suppose $\mathcal{F}$ and the conjunction of the formulas in* (2.9) *or* (2.10) *is satisfied at position* 0 *of some model π (a linear sequence in case of LTL, tree in case of CTL). Then the model contains positions $1, \ldots, n$ such that for each i between* 1 *and n, position i is a successor of $i-1$. Moreover, $\{d_0, \ldots, d_i\}$ are set to $\top$ and $\{d_{i+1}, \ldots, d_{n+1}\}$ are set to $\bot$ in position i. For any partition ρ, if j variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to $\top$ in position* 0*, then $\{tr^0_\rho, \ldots, tr^j_\rho\}$ are all set to $\top$ in position i. If j' variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to $\bot$ in position* 0*, then $\{fl^0_\rho, \ldots, fl^{j'}_\rho\}$ are all set to $\top$ in position i.*

$$determined \triangleq \bigwedge_{i=1}^{n} q_i \Rightarrow AGq_i \wedge \bigwedge_{i=1}^{n} \neg q_i \Rightarrow AG\neg q_i$$

$$depth \triangleq EX(d_1 \wedge \neg d_2) \wedge \bigwedge_{i=1}^{n-1} AG\left[(d_i \wedge \neg d_{i+1}) \Rightarrow EX(d_{i+1} \wedge \neg d_{i+2})\right]$$

$$\begin{aligned} setCounter \triangleq\ & (q_1 \Rightarrow t_{\uparrow part(1)}) \wedge (\neg q_1 \Rightarrow f_{\uparrow part(1)}) \\ & \wedge \bigwedge_{i=2}^{n} AG\left\{[d_{i-1} \wedge \neg d_i] \Rightarrow [(q_i \Rightarrow t_{\uparrow part(i)}) \wedge (\neg q_i \Rightarrow f_{\uparrow part(i)})]\right\} \end{aligned}$$

$$\begin{aligned} incCounter \triangleq\ & (t_{\uparrow part(1)} \Rightarrow AGtr^1_{part(1)}) \wedge (f_{\uparrow part(1)} \Rightarrow AGfl^1_{part(1)}) \\ & \wedge \bigwedge_{\rho=1}^{e} \bigwedge_{j=0}^{n[\rho]-1} AG[t_{\uparrow \rho} \Rightarrow (tr^j_\rho \Rightarrow AGtr^{j+1}_\rho)] \wedge AG[f_{\uparrow \rho} \Rightarrow (fl^j_\rho \Rightarrow AGfl^{j+1}_\rho)] \end{aligned}$$

$$\begin{aligned} targetMet \triangleq\ & \bigwedge_{\rho=1}^{e} AG[d_n \Rightarrow (tr^{tg(\rho)}_\rho \wedge \neg tr^{tg(\rho)+1}_\rho)] \\ & \wedge \bigwedge_{\rho=1}^{e} AG[d_n \Rightarrow (fl^{n[\rho]-tg(\rho)}_\rho \wedge \neg fl^{n[\rho]-tg(\rho)+1}_\rho)] \end{aligned} \tag{2.10}$$

$$determined' \triangleq \bigwedge_{\rho=1}^{e} tr^0_\rho \Rightarrow AGtr^0_\rho \wedge \bigwedge_{\rho=1}^{e} fl^0_\rho \Rightarrow AGfl^0_\rho$$

$$countInit \triangleq d_0 \wedge \neg d_1 \wedge \bigwedge_{\rho=1}^{e} (\neg tr^1_\rho \wedge \neg fl^1_\rho \wedge tr^0_\rho \wedge fl^0_\rho)$$

$$depth' \triangleq \bigwedge_{\rho=1}^{e} \bigwedge_{j=0}^{n[\rho]} \left[AG(tr^j_\rho \Rightarrow AGtr^j_\rho) \wedge AG(fl^j_\rho \Rightarrow AGfl^j_\rho)\right]$$

$$countMonotone \triangleq \bigwedge_{i=1}^{n} AG(d_i \Rightarrow d_{i-1}) \wedge \bigwedge_{\rho=1}^{e} \bigwedge_{j=2}^{n[\rho]} \left[AG(tr^j_\rho \Rightarrow tr^{j-1}_\rho) \wedge AG(fl^j_\rho \Rightarrow fl^{j-1}_\rho)\right]$$

Proof. We will first prove the existence of positions $1, \ldots, i$ by induction on i.

Base case $i = 1$: Since $\pi, 0 \models depth$, there must be a successor 1 of 0 that satisfies $d_1 \wedge \neg d_2$. Since $\pi, 0 \models G(d_3 \Rightarrow d_2)$ or $\pi, 0 \models AG(d_3 \Rightarrow d_2)$ (part of *countMonotone*) and 1 is a successor of 0, we get $\pi, 1 \models d_3 \Rightarrow d_2$. Since d_2 is set to $\bot$ in 1, this means that d_3 is also set to $\bot$ in 1. Similar reasoning can be used to prove that all of $\{d_2, \ldots, d_{n+1}\}$ are set to $\bot$ in 1. The fact that $\pi, 1 \models d_1 \Rightarrow d_0$ means that d_0 is set to $\top$ in 1 (since d_1 is set to $\top$ in 1).

Induction step: Assume that positions $1, \ldots, i$ exist in π with the stated properties. Hence, position i satisfies $d_i \wedge \neg d_{i+1}$. Since position 0 satisfies *depth* and i occurs after 0, there must be a successor $i + 1$ of i that satisfies $d_{i+1} \wedge \neg d_{i+2}$. The fact that position $i + 1$ satisfies $d_{i'} \Rightarrow d_{i'-1}$ for all i' (these formulae are part of *countMonotone* formula satisfied by position 0) can be used to show that all of $d_0, \ldots, d_{i+1}$ are set to $\top$ and all of $d_{i+2}, \ldots, d_{n+1}$ are set to $\bot$ in position $i + 1$.

We will now prove the second claim of the lemma, which is about values of $\{tr^0_\rho, \ldots, tr^j_\rho\}$ in position i. We will first prove that tr^j_ρ is set to $\top$ by induction on i.

Base case $i = 1$: If q_1 is not in part ρ, there is nothing to prove ($tr^0_{part(1)}$ is set to $\top$ in all positions). If q_1 is in part ρ and q_1 is set to $\bot$, there is nothing to prove. If q_1 is in part ρ and q_1 is set to $\top$, then since position 0 satisfies *setCounter*, we get $\pi, 0 \models q_1 \Rightarrow t_{\uparrow part(1)}$. Since,

q_1 is set to $\top$ and $part(1) = \rho$, we get that $t_{\uparrow\rho}$ is set to $\top$ in position 0. Since position 0 satisfies *incCounter*, we get $\pi, 0 \models t_{\uparrow\rho} \Rightarrow Gtr^1_\rho$ or $\pi, 0 \models t_{\uparrow\rho} \Rightarrow AGtr^1_\rho$ and hence $\pi, 0 \models Gtr^1_\rho$ or $\pi, 0 \models AGtr^1_\rho$. Since position 1 occurs after 0, we conclude that in position 1, tr^1_ρ is set to $\top$.

Induction step: Case 1: q_i is not in part ρ and none of the variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to to $\top$ in i. In this case, there is nothing to prove.

Case 2: q_i is not in part ρ and some $1 \leq j < i$ variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to $\top$. By the induction hypothesis, tr^j_ρ is set to $\top$ in position $i-1$. Now $\pi, 0 \models depth'$. Hence $\pi, 0 \models G(tr^j_\rho \Rightarrow Gtr^j_\rho)$ or $\pi, 0 \models AG(tr^j_\rho \Rightarrow AGtr^j_\rho)$, and hence $\pi, i-1 \models tr^j_\rho \Rightarrow Gtr^j_\rho$ or $\pi, i-1 \models tr^j_\rho \Rightarrow AGtr^j_\rho$(since position $i-1$ occurs after 0), and hence $\pi, i-1 \models Gtr^j_\rho$ or $\pi, i-1 \models AGtr^j_\rho$(since tr^j_ρ is set to $\top$ in position $i-1$), and hence $\pi, i \models tr^j_\rho$ (since position i occurs after $i-1$).

Case 3: q_i is in part ρ and q_i is set to $\bot$. If none of the variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to $\top$, then the argument is similar to case 1. If some $1 \leq j < i$ variables in $\Phi(\rho) \cap \{q_1, \ldots, q_i\}$ are set to $\top$, then the argument is similar to case 2.

Case 4: q_i is in part ρ and q_i is set to $\top$. We know that position $i-1$ satisfies $d_{i-1} \wedge \neg d_i$. Since position 0 satisfies *setCounter*, we have $\pi, 0 \models G\left\{[d_{i-1} \wedge \neg d_i] \Rightarrow [q_i \Rightarrow t_{\uparrow part(i)}]\right\}$ or $\pi, 0 \models AG\left\{[d_{i-1} \wedge \neg d_i] \Rightarrow [q_i \Rightarrow t_{\uparrow part(i)}]\right\}$, and hence $\pi, i-1 \models [d_{i-1} \wedge \neg d_i] \Rightarrow [q_i \Rightarrow t_{\uparrow part(i)}]$ (since position $i-1$ occurs after 0), and hence $\pi, i-1 \models q_i \Rightarrow t_{\uparrow\rho}$ (since $\pi, i-1 \models d_{i-1} \wedge \neg d_i$), and hence $\pi, i-1 \models t_{\uparrow\rho}$ (since $\pi, i-1 \models q_i$). Since position 0 satisfies *incCounter* and position $i-1$ occurs after 0, we get $\pi, i-1 \models t_{\uparrow\rho} \Rightarrow (tr^{j-1}_\rho \Rightarrow Gtr^j_\rho)$ or $\pi, i-1 \models t_{\uparrow\rho} \Rightarrow (tr^{j-1}_\rho \Rightarrow AGtr^j_\rho)$. We have already seen that $t_{\uparrow\rho}$ is set to $\top$ in position $i-1$ and tr^{j-1}_ρ is set to $\top$ in position $i-1$ by the induction hypothesis (j is at least 1 since q_i is in part ρ and is set to $\top$). Hence, we get $\pi, i-1 \models Gtr^j_\rho$ or $\pi, i-1 \models AGtr^j_\rho$. Since position i occurs after $i-1$, we conclude that tr^j_ρ is set to $\top$ in i.

Now, since position 0 satisfies $G(tr^j_\rho \Rightarrow tr^{j-1}_\rho)$ or $AG(tr^j_\rho \Rightarrow tr^{j-1}_\rho)$ (this is part of *countMonotone*) and position i is a successor of 0, we get $\pi, i \models tr^j_\rho \Rightarrow tr^{j-1}_\rho$. Since tr^j_ρ is set to $\top$ in position i, it follows that tr^{j-1}_ρ is also set to $\top$ in position i. Similarly, $tr^0_\rho, \ldots, tr^j_\rho$ are all set to $\top$ in position i.

The proof for values of $\{fl^0_\rho, \ldots, fl^{j'}_\rho\}$ is symmetric to the proof for values of $\{tr^0_\rho, \ldots, tr^j_\rho\}$. □

It is routine to verify that the proofs of Theorem 2.16 and Lemma 2.17 also prove Theorem 2.21 and Lemma 2.22 below respectively. This completes the proof of Theorem 2.18.

Theorem 2.21. *If the LTL or CTL formula obtained by conjoining $\mathcal{F}$ with the formulas in* (2.9) *or* (2.10) *is satisfiable, then the* p-PW-SAT *instance is a* YES *instance.*

Lemma 2.22. *The pathwidth of the LTL or CTL formula obtained by conjoining $\mathcal{F}$ with the formulas in* (2.9) *or* (2.10) *is at most* $4pw + 2e + 5$.

2.8 Remarks and Open Problems

By expressing satisfiability of modal formulae as a MSO property, we obtained a FPT algorithm for modal satisfiability in general models with treewidth and modal depth as parameters. The resulting algorithm has running time linear in the size of the modal formula by Courcelle's theorem. Due to the dependence of the constructed MSO sentence on modal depth, the FPT algorithm obtained in section 2.2 has a running time with a tower of 2's whose height is $\mathcal{O}(\mathrm{md}(\phi))$. It remains to be seen if this can be improved or there is a lower bound that prevents any improvement. Lower bounds in similar but different contexts have

been proved in [1, 78]. The parameterized complexity of modal satisfiability in symmetric models is open.

In modal formulas, different occurrences of the same sub-formula are considered as separate syntactic objects, increasing the overall size of the formula if same sub-formula is repeated many times. Another direction is to look at satisfiability of modal circuits, where there is only one syntactic object for a sub-formula and different occurrences of that sub-formula are indicated by referring to the same syntactic object. There are variations of incidence graphs, such as primal/dual graphs [88], whose effect on complexity of satisfiability may also be considered.

Chapter 3

Synchronized Transition Systems

Labelled Transition Systems (LTSs) are popularly used to model sequential systems. Synchronized transition systems are generalizations where we can model many sequential systems interacting with one another through synchronization. With the number of systems participating in such synchronization as a parameter, parameterized complexity of many problems like reachability and model checking are studied in [21]. All the problems studied are shown to be complete for some intractable parameterized complexity class. In this chapter, we will introduce a stronger parameter and give parameterized complexity results with a combination of the new and old parameters.

3.1 Preliminaries

A *labelled transition system* (LTS) over some alphabet $\Sigma = \{a, b, \dots\}$ is a tuple $\mathscr{A} = (Q, \Sigma, \rightarrow)$ where $Q = \{s, u, \dots\}$ is the set of local states and $\rightarrow \subseteq Q \times \Sigma \times Q$ is the set of local transitions. We assume the standard notation $s \xrightarrow{a} u$, $s \xrightarrow{w} u$ ($w \in \Sigma^*$), $s \xrightarrow{*} u$, $s \xrightarrow{+} u$ etc. For $W \subseteq \Sigma^*$, $s \xrightarrow{W} u$ means that for some $w \in W$, $s \xrightarrow{w} u$. The size of a finite LTS $\mathscr{A}$ is $|\mathscr{A}| = |Q| + |\Sigma| + |\rightarrow|$. *Product systems* are products of individual LTSs. Assuming $\mathscr{A}_i = (Q_i, \Sigma, \rightarrow_i)$ for $i = 1, \dots, k$, which are called *processes*, the product $\mathscr{A}_1 \times \cdots \times \mathscr{A}_k$ denotes a LTS $(Q, \Sigma, \rightarrow)$ where $Q = \prod_{i=1}^{k} Q_i$ is the set of global states and the set of global transitions $\rightarrow \subseteq Q \times \Sigma \times Q$ is defined as follows. The processes synchronize on common actions: $\langle s_1, \dots, s_k \rangle \xrightarrow{a} \langle u_1, \dots, u_k \rangle$ iff $s_i \xrightarrow{a} u_i$ for every $i = 1, \dots, k$. A global state $\overline{s} = \langle s_1, \dots, s_k \rangle$ of $\mathscr{A}_1 \times \cdots \times \mathscr{A}_k$ corresponds to each process $\mathscr{A}_i$ being in the local state s_i for every $i = 1, \dots, k$.

Definition 3.1. *Given an initial and a final global state of the product system, the reachability problem is to check if starting from the initial global state, the product system can reach the final global state through a sequence of transitions.*

It is known that for 1-safe Petri nets (which are a form of product systems), reachability and other problems like model checking LTL and μ-calculus are PSPACE-complete [30]. In [21], the parameterized complexity of these problems for product systems are studied with the number of processes k as parameter. Following are the main results from [21], where the number of processes k is the only parameter by default.

1. Reachability and its refinements repeated reachability, local reachability and fair reachability are AW[SAT]-hard.

2. Linear Temporal Logic and Computational Tree Logic model checking are AW[SAT]-hard with both the size of the formula and the number of processes k as parameters.

3. Hennessy-Milner Logic model checking is AW[1]-complete with both the size of the formula and the number of processes k as parameters.

4. μ-calculus model checking is XP-complete with both the size of the formula and the number of processes k as parameters.

5. Bismimulation is XP-complete.

It is shown in [21] that most of the above classification remain intact under variations like replacing global synchronization with binary synchronization (where only two processes can participate in a global transition) and bounding the size of the alphabet Σ by a constant. With such robust hardness results, we are forced to define stronger parameters to obtain FPT results.

Following is the definition of a stronger parameter with which we show FPT results. The intuition behind this definition is to restrict a particular behaviour of product systems that is common in most of the hardness proofs mentioned in the previous paragraph. Consider a product system consisting of two processes $\mathscr{A}_1$ and $\mathscr{A}_2$. Let $\{s_1, \ldots, s_{n1}\}$ and $\{u_1, \ldots, u_{n2}\}$ be the set of local states of $\mathscr{A}_1$ and $\mathscr{A}_2$ respectively. The number of global states would be very large if every global state (s_i, u_j) is reachable in the product system. It is reasonable to assume that in product systems modelling human designed systems, a reachable global state (s_i, u_j) would imply some connection between s_i and u_j and that not every pair of local states will have such a connection. The following definition formalizes this.

Definition 3.2. *For two LTSs $\mathscr{A}_1, \mathscr{A}_2$ with initial local states s_{in}, u_{in} and a set of strings $W \subseteq \Sigma^*$, suppose s is a local state of $\mathscr{A}_1$ such that $\forall w \in W, \quad s_{in} \xrightarrow{w} s$. Define* **synchronous W-neighbours** *of $s \in Q_1$ to be $Nbr[W](s) = \{u \in Q_2 \mid u_{in} \xrightarrow{W} u\}$. Let $m[W, \mathscr{A}_1, \mathscr{A}_2] = \max\{|Nbr[W](s)| \mid s \in Q_1\}$. This means that there are $m[W, \mathscr{A}_1, \mathscr{A}_2]$ global states all of which have the same local state in $\mathscr{A}_1$ but have distinct local states in $\mathscr{A}_2$. To generalize this, let $\mathscr{A}_1, \ldots, \mathscr{A}_k$ be LTSs and define $m[i, j] = \max\{m[W, \mathscr{A}_i, \mathscr{A}_j] \mid W \subseteq \Sigma^*\}$, for $i \neq j \in \{1, \ldots, k\}$. Let $m[i] = \max\{m[i, j] \mid j \neq i\}$ and $m = \min\{m[i] \mid 1 \leq i \leq k\}$. The number m is called the* **synchronous neighbourhood size** *of $\mathscr{A}_1 \times \cdots \times \mathscr{A}_k$.*

Figure 3.1 shows an example of a synchronized transition system with three processes. Local states are shown as small circles and local transitions as directed edges with labels alongside. Consider the local state s of $\mathscr{A}_1$ and the set of strings W represented by the

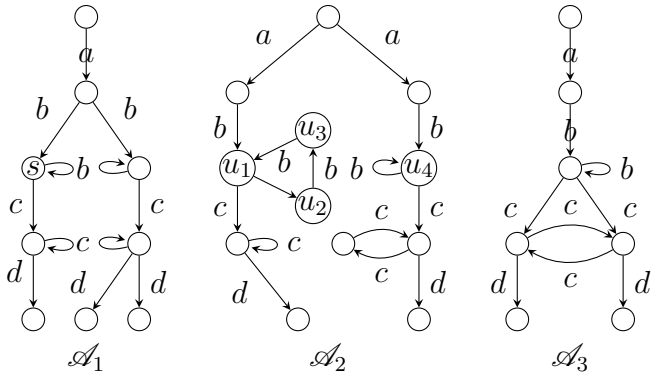


Figure 3.1: An example system with small synchronous neighbourhood size

regular expression abb^*. Considering $\mathscr{A}_1$ and $\mathscr{A}_2$, the synchronous W-neighbours of s is given by $Nbr[W](s) = \{u_1, u_2, u_3, u_4\}$. It can be seen that $m[2] = 3$ as per Definition 3.2.

In the following section, we give FPT upper bounds when both synchronous neighbourhood size and number of processes are parameters. In section 3.3, we show W[1]-hardness when synchronous neighbourhood size is the only parameter.

3.2 Upper Bound

Lemma 3.3. *For a product system $\mathscr{A}_1 \times \cdots \times \mathscr{A}_k$ with synchronous neighbourhood size m, the number of reachable global states is at most nm^{k-1}, where n is the maximum number of local states in any one LTS.*

Proof. The required result is obvious for $k = 1$ since $nm^{1-1} = n$. Suppose $k > 1$ and the number of reachable global states is greater than or equal to $nm^{k-1} + 1$. In such a case, we claim that $m[i] > m$ for every $i = 1, \ldots, k$, a contradiction.

For any $i = 1, \ldots, k$, there are at most n local states in $\mathscr{A}_i$. For each local state s of $\mathscr{A}_i$, let Q_s be the set of $k-1$ tuples that form a reachable global state when combined with s. There is at least one local state s in $\mathscr{A}_i$ such that $|Q_s| \geq m^{k-1} + 1$ (otherwise, total number of reachable global states will be at most nm^{k-1}). Let s be such a local state. For each $j \neq i$, let Q_s^j be the set of local states in $\mathscr{A}_j$ that appear in the j^{th} process of some tuple in Q_s. There is at least one j such that $|Q_s^j| \geq m + 1$ (otherwise, $|Q_s| \leq m^{k-1}$). Let $Q_s^j = \{u_1, \ldots, u_{m+1}, \ldots\}$. Now, $(s, u_1), \ldots, (s, u_{m+1})$ are each part of some reachable global state. Let w_l be the sequence of transitions that enables the product system to reach a global state with $\mathscr{A}_i$ in local state s and $\mathscr{A}_j$ in local state u_l. Now, with the set of strings $\{w_1, \ldots, w_{m+1}\}$, $\mathscr{A}_i$ reaches one local state s while for the same set of strings, $\mathscr{A}_j$ reaches $m + 1$ different local states $\{u_1, \ldots, u_{m+1}\}$. Hence, $m[i, j] > m$. □

Theorem 3.4. *With the number of processes k and synchronous neighbourhood size m as parameters, reachability in product systems is* FPT.

Proof. With both the number of processes m and synchronous neighbourhood size k as parameters, the upper bound of Lemma 3.3 allows us to construct the whole state space of the composed system in FPT time, hence reachability is FPT. □

3.3 Lower Bound

With the result of the previous section, a natural question would be whether reachability is FPT when parameterized by the synchronized neighbourhood size alone. In this section, we answer this question in the negative, by a parameterized reduction from the Parameterized Weighted 2-CNF$^-$ Satisfiability (p-W2CNF$^-$-SAT). An instance of p-W2CNF$^-$-SAT consists of a 2-CNF formula $\mathcal{F}$ over the propositional variables $q_1, \ldots, q_n$ and a number k such that each literal in every clause is a negated propositional variable. The problem is to check if there is an assignment satisfying $\mathcal{F}$ while setting exactly k variables to $\top$. This problem is known to be W[1]-hard [37, Theorem 6.28].

Theorem 3.5. *With the synchronous neighbourhood size m as parameter, reachability in product systems is* W[1]*-hard.*

Proof. Given an instance of p-W2CNF$^-$-SAT, construct a product system consisting of the following LTSs. The LTS shown in Fig. 3.2 is a controller ensuring that exactly k variables are set to $\top$. For q_1, q_n and all q_i for each i between 2 and $n-1$, the LTSs shown in Fig. 3.3, Fig. 3.5 and Fig. 3.4 are constructed respectively. If there are m clauses $C_1, \ldots, C_m$ in $\mathcal{F}$, then for each j between 1 and $m-1$, the LTS shown in Fig. 3.6 is constructed. Finally, the

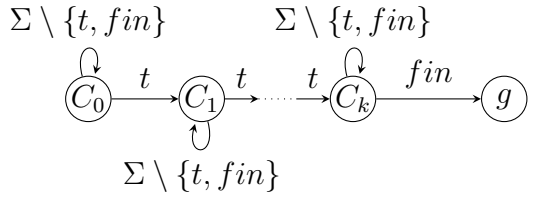


Figure 3.2: Controller

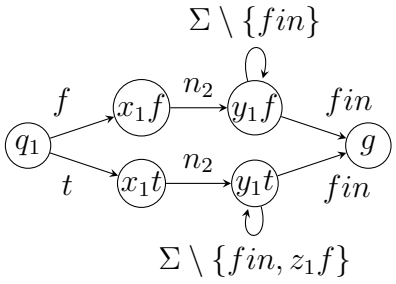


Figure 3.3: Assignment for q_1

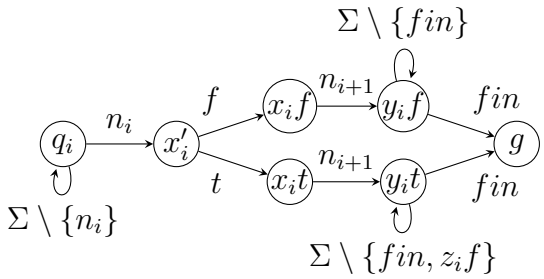


Figure 3.4: Assignment for q_i

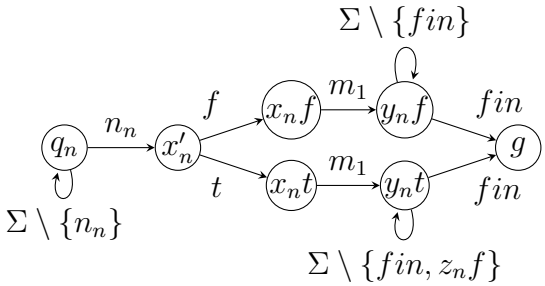


Figure 3.5: Assignment for q_n

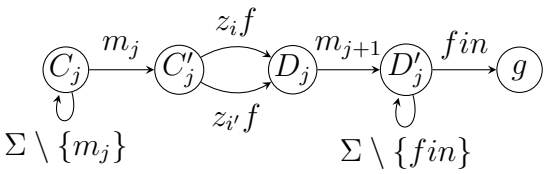


Figure 3.6: Checking satisfaction of C_j

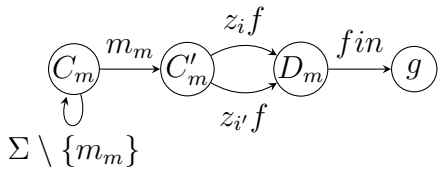


Figure 3.7: Checking satisfaction of C_m

LTS shown in Fig. 3.7 is also constructed. In Fig. 3.6 (resp. Fig. 3.7), it is assumed that the clause C_j (resp. C_m) is $\neg q_i \vee \neg q_{i'}$.

In the initial global state, the controller LTS in Fig. 3.2 is in the state C_0. LTSs in Fig. 3.3, Fig. 3.4 and Fig. 3.5 are in states q_1, q_i and q_n respectively. LTSs in Fig. 3.6 and Fig. 3.7 are in states C_j and C_m respectively. In the final global state to be reached, all LTSs are in the state g.

Suppose that the given p-W2CNF$^-$-SAT instance is satisfiable. For each q_i, if the satisfying assignment assigns $\bot$, take the transition labelled f from x'_i to $x_i f$ in the LTS of Fig. 3.4. If the satisfying assignment assigns $\top$, take the transition labelled t from x'_i to $x_i t$, taking the controller LTS one step nearer to C_k in the process. Since there are exactly k variables set to $\top$ in the satisfying assignment, the controller LTS will reach C_k, k LTSs reach $y_i t$ and the remaining LTSs among those in Fig. 3.3, Fig. 3.4 and Fig. 3.5 reach $y_i f$. Since all clauses are satisfied, LTSs in Fig. 3.6 and Fig. 3.7 can reach the state D'_j. Now, all LTSs can take the transition labelled fin to reach the state g.

On the other hand, suppose the final global state can be reached from the initial global state in the product system constructed above. Consider the assignment that assigns $\top$ to q_i if the LTS in Fig. 3.3 takes the transition labelled t from x'_i to $x_i t$ and assigns $\bot$ otherwise. Since the controller LTS in Fig. 3.2 needs exactly k transitions labelled t to reach g, the assignment constructed above sets k variables to $\top$ and all others to $\bot$. Since all LTSs in Fig. 3.6 and Fig. 3.7 reach g and the clause C_j has to be satisfied to reach D_j from C'_j, the constructed assignment satisfies all clauses.

Except the controller LTS of Fig. 3.2, all other LTSs have a constant number of states. Hence, the synchronous neighbourhood size of the product system constructed above is a constant. Therefore, with synchronous neighbourhood size as parameter, reachability in product systems is W[1]-hard. □

Chapter 4

1-safe Petri Nets

1-safe Petri nets are powerful models that can compactly represent finite state systems. They can model concurrent behaviour such as causality, independence, synchronization etc. Complexity of various problems of 1-safe Petri nets are known [14, 30], most of them PSPACE-complete. In this chapter, we show either a FPT algorithm or W[1]-hardness for various problems with respect to the parameters treewidth, benefit depth and vertex cover number. Treewidth is a graph theoretic concept widely used as a parameter to get tractable algorithms for many hard problems [16, 3]. Apart from throwing light on the effect of treewidth on 1-safe Petri nets, we also get as a corollary a proof of a conjecture about a graph pebbling problem made by Downey, Fellows and Stege [27, section 5].

It turns out that most of the problems for 1-safe Petri nets are W[1]-hard with treewidth as parameter. Therefore, we turn to the stronger parameter vertex cover number. This helps in understanding the role of different parameters in the complexity of our problems and the techniques may be useful in other contexts as well. We show that LTL model checking is FPT with the size of the LTL formula and vertex cover number as parameters, by combining the well known automata theoretic method with a powerful result about feasibility of Integer Linear Programming (ILP) parameterized by the number of variables [64, 54, 38]. Automata theoretic methods are extensively studied and arise in many other contexts [97, 45, 59].

The SIGNED DIGRAPH PEBBLING problem considered in [27] can simulate 1-safe Petri nets and it is shown that with treewidth and the length of the firing sequence as parameters, this problem is FPT. Downey, Fellows and Stege conjectured that with treewidth alone as parameter, the problem is W[1]-hard [27], which we prove here.

4.1 Petri Nets and Problem Definitions

Let $\mathbb{Z}$ be the set of integers and $\mathbb{N}$ the set of natural numbers. We first define general Petri nets and then impose some restrictions to get 1-safe Petri nets. A Petri net is a 4-tuple $\mathcal{N} = (P, T, Pre, Post)$, where P is a set of places, T is a set of transitions and Pre and $Post$ are the incidence functions: $Pre : P \times T \to [0 \ldots W]$ (arcs going from places to transitions) and $Post : P \times T \to [0 \ldots W]$ (arcs going from transitions to places), where $W \geq 1$ is the maximum arc weight. A place p is an *input* (*output*) place of a transition t if $Pre(p,t) = 1$ ($Post(p,t) = 1$) respectively. Conventionally, ${}^\bullet t$ ($t^\bullet$) denote the set of input (output) places of a transition t. In diagrams, places will be represented by circles and transitions by thick bars. Arcs are represented by weighted directed edges between places and transitions.

A function $M : P \to \mathbb{N}$ is called a *marking*. A marking can be thought of as a configuration of the Petri net, with every place p having $M(p)$ tokens. Given a Petri net $\mathcal{N}$ with a marking M and a transition t such that for every place p, $M(p) \geq Pre(p,t)$, the transition t is said to be *enabled* at M and can be *fired*. After firing, the new marking M' (denoted as $M \stackrel{t}{\Longrightarrow} M'$)

is given by $M'(p) = M(p) - Pre(p,t) + Post(p,t)$ for every place p. A place p is an *input* (*output*) place of a transition t if $Pre(p,t) \geq 1$ ($Post(p,t) \geq 1$) respectively. We can think of firing a transition t resulting in $Pre(p,t)$ tokens being deducted from every input place p and $Post(p',t)$ tokens being added to every output place p'. A sequence of transitions $\sigma = t_1 t_2 \cdots t_r$ (called *firing sequence*) is said to be enabled at a marking M if there are markings $M_1, \ldots, M_r$ such that $M \stackrel{t_1}{\Longrightarrow} M_1 \stackrel{t_2}{\Longrightarrow} \cdots \stackrel{t_r}{\Longrightarrow} M_r$. Markings $M, M_1, \ldots, M_r$ are called *intermediate markings* and the sequence of markings $MM_1 \cdots M_r$ is called a *run*. The fact that firing σ at M results in M_r is denoted by $M \stackrel{\sigma}{\Longrightarrow} M_r$ and we say that M_r is reachable from M. The set $\mathcal{R}(\mathcal{N}, M_0)$ is the set of markings reachable from the initial marking M_0.

We assume that a Petri net is presented as two matrices for *Pre* and *Post*. We will assume that a Petri net $\mathcal{N}$ has m places, n transitions and that W is the maximum of the range of *Pre* and *Post*. We define the size of the Petri net to be $|\mathcal{N}| = 2mn \log W + m \log |M_0|$ bits, where $|M_0|$ is the maximum of the range of the initial marking M_0.

Definition 4.1 (Reachability, Coverability and Boundedness)**.** *Given a Petri net with an initial marking M_0 and a target marking M_{cov}, the* Coverability *problem is to determine if there is a firing sequence σ such that $M_0 \stackrel{\sigma}{\Longrightarrow} M'$ and for every place p, $M'(p) \geq M_{cov}(p)$ (this is denoted as $M' \geq M_{cov}$). If we replace $M' \geq M_{cov}$ by $M' = M_{cov}$, we get the* reachability *problem. The boundedness problem is to determine if there is a number $c \in \mathbb{N}$ such that for every firing sequence σ enabled at M_0 with $M_0 \stackrel{\sigma}{\Longrightarrow} M$, $M(p) \leq c$ for every place p.*

Let $\mathcal{N}$ be a Petri net with an initial marking M_0 with $M_0(p) \in \{0,1\}$ for every place p. For every firing sequence σ enabled at M_0 with $M_0 \stackrel{\sigma}{\Longrightarrow} M$, if we have $M(p) \in \{0,1\}$ for every place p, then the Petri net $\mathcal{N}$ with initial marking M_0 is said to be a 1-safe Petri net. Reachability, coverability and model checking some temporal logics on 1-safe Petri nets are complete for PSPACE.

4.2 Logics for Specifying Properties

We will use Linear Temporal Logic (LTL) for specifying properties of 1-safe Petri nets. Syntax of LTL is the same as the one defined in section 2.1, except that instead of a set Φ of atomic propositional variables, we will use the set of places of a Petri net. For a Petri net with set of places P, LTL formulas are formed by the following syntax. We follow the notation in [30].

$$\phi ::= p \in P \mid \neg\, \phi \mid \phi_1 \wedge \phi_2 \mid X\phi \mid \phi_1 U \phi_2$$

These LTL formulas are interpreted on runs of a 1-safe Petri net. Given a run $\pi = M_0 M_1 \cdots$, the satisfaction of a LTL formula ϕ at some position i in this sequence is defined as follows.

- $\pi, i \models p$ iff $M_i(p) = 1$.
- $\pi, i \models \neg\phi$ iff $\pi, i \not\models \phi$.
- $\pi, i \models \phi_1 \wedge \phi_2$ iff $\pi, i \models \phi_1$ and $\pi, i \models \phi_2$.
- $\pi, i \models X\phi$ iff there is a position $i+1$ in π and $\pi, i+1 \models \phi$.
- $\pi, i \models \phi_1 U \phi_2$ iff for some $j \geq i$, $\pi, j \models \phi_2$ and for all $i \leq l < j$, $\pi, l \models \phi_1$.

Usual abbreviations are $\top = p \vee \neg p$, $F\phi = \top U \phi$ and $G\phi = \neg F \neg \phi$. Let $\mathcal{N}$ be a 1-safe Petri net with initial marking M_0. A firing sequence σ enabled at M_0 is said to be *maximal* if σ is infinite or it is finite, $M_0 \stackrel{\sigma}{\Longrightarrow} M$ and no transition is enabled at M. A 1-safe net $\mathcal{N}$ with initial marking M_0 is a model of an LTL formula ϕ iff every run π that can be

obtained by firing any maximal sequence of transitions from M_0 (we will call them **maximal runs**) satisfies $\pi, 0 \models \phi$. Much more expressive is the Monadic Second Order (MSO) logic of Büchi [12], interpreted on a maximal run $M_0 M_1 \cdots$, with $\pi, s \models p(x)$ iff $M_{s(x)}(p) = 1$ under an assignment s to the variables, which denote positions of the run. First- and (monadic) second-order quantifiers and Boolean operations are available as usual. The satisfaction of a MSO formula by a run is independent of places that do not occur in the MSO formula:

Proposition 4.2. *Let $\mathcal{N}$ be a 1-safe net and ϕ be an MSO formula. Let $P_\phi \subseteq P$ be the subset of places that occur in ϕ. Let $\pi = M_0 M_1 \cdots$ and $\pi' = M'_0 M'_1 \cdots$ be two finite or infinite runs of $\mathcal{N}$ such that for all positions i of π and for all $p \in P_\phi$, $M_i(p) = M'_i(p)$. For any assignment s, we have $\pi, s \models \phi$ iff $\pi', s \models \phi$.*

Proof. By a straightforward induction on the structure of ϕ. □

Some interesting properties of Petri nets like liveness (a transition is live if it can always occur again) cannot be expressed in LTL. Another such example is specifying that the initial marking is a home marking, which means that the initial marking can be reached from any reachable marking. Computational Tree Logic (CTL) introduces quantification over set of computations starting from a marking to enable expressing such properties. As in LTL, CTL syntax is the same as given in section 2.1 with the set of places P replacing the set of atomic propositional variables Φ. Following is the CTL syntax for 1-safe Petri nets, again using the notation of [30].

$$\phi ::= p \in P \mid \neg\phi \mid \phi \wedge \phi \mid EX\phi \mid AX\phi \mid E[\phi_1 U \phi_2] \mid A[\phi_1 U \phi_2]$$

CTL formulas are interpreted on computation trees, which are possibly infinite trees labelled with a set of atomic propositions. In case of 1-safe Petri nets, the set of atomic propositions is the set of places. The root of the tree is labelled with the initial marking. Every node of the tree labelled by some marking M will have as its successors, the set of nodes labelled by markings that are reachable from M by firing a transition. For a tree $\mathcal{T}$ and a node η labelled by the marking M_η, satisfaction of a CTL formula is defined as follows.

- $\mathcal{T}, \eta \models p$ iff $M_\eta(p) = 1$.
- $\mathcal{T}, \eta \models \neg\phi$ iff $\mathcal{T}, \eta \not\models \phi$.
- $\mathcal{T}, \eta \models \phi_1 \wedge \phi_2$ iff $\mathcal{T}, \eta \models \phi_1$ and $\mathcal{T}, \eta \models \phi_2$.
- $\mathcal{T}, \eta \models EX\phi$ iff there exists a child η' of η with $\mathcal{T}, \eta' \models \phi$.
- $\mathcal{T}, \eta \models AX\phi$ iff for every child η' of η, $\mathcal{T}, \eta' \models \phi$.
- $\mathcal{T}, \eta \models E[\phi_1 U \phi_2]$ iff for some path $\eta_0 \eta_1 \eta_2 \cdots$ of $\mathcal{T}$ starting from $\eta = \eta_0$, there exists $i \geq 0$ such that $\mathcal{T}, \eta_i \models \phi_2$ and for all j with $0 \leq j < i$, $\mathcal{T}, \eta_j \models \phi_1$.
- $\mathcal{T}, \eta \models A[\phi_1 U \phi_2]$ iff for every path $\eta_0 \eta_1 \eta_2 \cdots$ of $\mathcal{T}$ starting from $\eta = \eta_0$, there exists $i \geq 0$ such that $\mathcal{T}, \eta_i \models \phi_2$ and for all j with $0 \leq j < i$, $\mathcal{T}, \eta_j \models \phi_1$.

Usual abbreviations are $EF\phi = E[\top U \phi]$, $AG\phi = \neg EF \neg\phi$, $AF\phi = A[\top U \phi]$ and $EG\phi = \neg AF \neg\phi$. Reachability, coverability and LTL and CTL model checking for 1-safe Petri nets are all PSPACE-complete [30].

4.3 Treewidth, Pathwidth and 1-safe Petri Nets

Given a Petri net $\mathcal{N}$, we associate with it an undirected **flow graph** $G(\mathcal{N}) = (P, E)$ where $(p_1, p_2) \in E$ iff for some transition t, $Pre(p_1, t) + Post(p_1, t) \geq 1$ and $Pre(p_2, t) + Post(p_2, t) \geq 1$. If a place p is both an input and an output place of some transition, the vertex corresponding to p has a self loop in $G(\mathcal{N})$. Informally, if $P_1 = t^\bullet \cup {}^\bullet t$ is the set of input and output places of a transition t, then P_1 induces a clique in the flow graph. With a similar notion of a flow graph for product systems that we saw in Chapter 3, a global transition would have induced a clique intersecting each process. Presence of cliques increases the treewidth and pathwidth of graphs. In 1-safe Petri nets, such induced cliques are localized to set of places interacting with a transition.

In this section, we will prove W[1]-hardness for problems associated with 1-safe nets, with treewidth of the flow graph of the Petri net as parameter. For this, we give a parameterized reduction from the p-PW-SAT problem. Recall from Lemma 2.12 that when parameterized by the number of parts e and the pathwidth of the primal graph, p-PW-SAT is W[1]-hard.

Now we will demonstrate a parameterized reduction from p-PW-SAT to reachability in 1-safe Petri nets, with the pathwidth of the flow graph of the Petri net as parameter. Given an instance of p-PW-SAT, let $q_1, \ldots, q_n$ be the variables used. Construct an optimal path decomposition of the primal graph of the CNF formula in the given p-PW-SAT instance (this can be done in FPT time [8]). For every clause in the CNF formula, the primal graph contains a clique formed by all variables occurring in that clause. There will be at least one bag in the path decomposition of the primal graph that contains all the vertices in this clique. Order the bags of the path decomposition from left to right and call the clause whose clique appears first C_1, the clause whose clique appears second as C_2 and so on. If more than one such clique appears for the first time in the same bag, order the corresponding clauses arbitrarily. Let $C_1, \ldots, C_m$ be the clauses ordered in this way. We will call this the **path decomposition ordering** of clauses, and use it to prove that the pathwidth of the flow graph of the constructed 1-safe Petri net is low (Lemma 4.4). For a partition ρ between 1 and e, we let $n[\rho]$ be the number of variables in ρ. We will construct a 1-safe Petri net with the following places.

1. For every propositional variable q_i used in the given p-PW-SAT instance, places $q_i, x_i, \overline{x}_i$.
2. For every partition ρ between 1 and e, places $t_{\uparrow\rho}, f_{\uparrow\rho}, tr^0_\rho, \ldots, tr^{tg(\rho)}_\rho$ and $fl^0_\rho, \ldots, fl^{n[\rho]-tg(\rho)}_\rho$.
3. For each clause C_j, a place C_j. An additional place C_{m+1}.
4. Places s (for serializing certain operations in the Petri net) and g (for checking some global conditions).

The construction of the Petri net is illustrated in the following diagrams, divided into modules for understandability. The notation $part(i)$ stands for the partition to which q_i belongs. For each propositional variable q_i used in the given p-PW-SAT instance, part of the net shown in Fig. 4.1 is constructed. Intuitively, the truth assignment of q_i is determined by firing t_i or f_i. The token in $x_i/\overline{x}_i$ is used to check satisfaction of clauses later. The token in $t_{\uparrow part(i)}/f_{\uparrow part(i)}$ is used to count the number of variables set to $\top/\bot$ in each part, with the part of the net in Fig. 4.2. To transfer a token from tr^0_ρ to tr^1_ρ, we need to have a token in $t_{\uparrow\rho}$. To transfer a token from tr^0_ρ to $tr^{tg(\rho)}_\rho$, we will need $tg(\rho)$ tokens in $t_{\uparrow\rho}$. Similarly for $f_{\uparrow\rho}$.

For each clause C_j between 1 and m, the part of the net shown in Fig. 4.3 is constructed. In Fig. 4.3, it is assumed that $C_j = q_1 \vee \overline{q_2} \vee q_3$. For each clause, the corresponding part of the net has to be constructed similarly according to the literals occurring in the clause. Intuitively, a token can be moved from place C_j to C_{j+1} iff the clause C_j is satisfied by the truth assignment determined by the firings of t_i/f_i for each i between 1 and n.

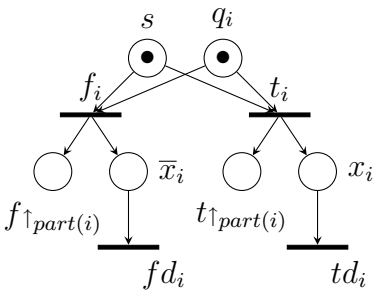


Figure 4.1: Part of the net for each variable q_i

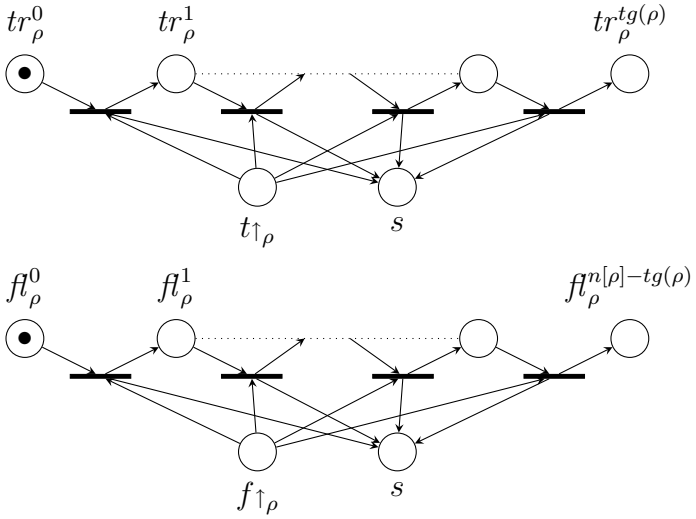


Figure 4.2: Part of the net for each part ρ between 1 and e

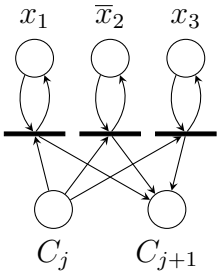


Figure 4.3: Part of the net for each clause C_j

Finally, the part of the net in Fig. 4.4 checks that the given target has been met for all parts. The initial marking of the constructed net consists of 1 token each in the places

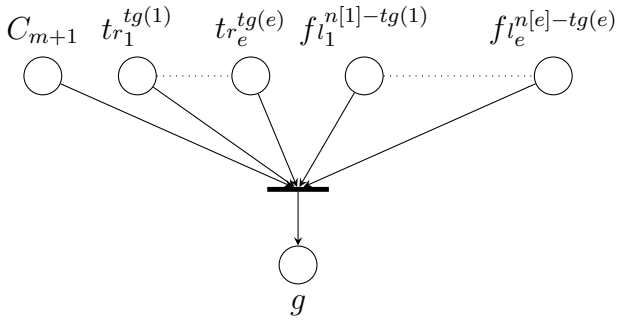


Figure 4.4: Part of the net to check that target has been met

$q_1, \ldots, q_n, s, tr_1^0, \ldots, tr_e^0, fl_1^0, \ldots, fl_e^0$ and C_1, with 0 tokens in all other places. The final marking to be reached consists of 1 token each in the places s, g and 0 tokens in all other places.

Lemma 4.3. *Given a* p-PW-SAT *instance, the Petri net described above can be constructed in* FPT *time. The constructed Petri net is* 1*-safe. The given instance of* p-PW-SAT *is a* YES *instance iff in the constructed* 1*-safe net, the required final marking can be reached from the given initial marking.*

Proof. The only non-trivial process in the construction of the Petri net is computing an optimal path decomposition of the primal graph of the CNF formula in the given p-PW-SAT instance. This can be done in FPT time and the rest of the construction can be done in polynomial time.

To see that the constructed net is 1-safe, observe that in the given initial marking, the only transitions enabled are $t_1, f_1, \ldots, t_n, f_n$. Only one of these transitions can be fired since they all take away the only token available in the place s. Once some transition t_i/f_i fires, the only other transition enabled is tc_i/fc_i respectively (transitions td_i/fd_i will also be enabled, but since they never add any token to any place, firing those will never violate 1-safeness). On firing one of these, the only transition enabled is the transition in Fig. 4.2 that can shift a token from $tr^0_{part(i)}$ to $tr^1_{part(i)}$. On firing this transition, the situation is similar to the one at the initial marking. If some clause C_j is satisfied by the truth assignment determined by the firing of t_i/f_i transitions, some transition in Fig. 4.3 may be enabled. However, such a transition may only shift a token from C_j to C_{j+1} and will not violate 1-safeness. Finally, the transition in Fig. 4.4 is enabled at most once since all its input places can get at most one token at most once.

Suppose the given instance of p-PW-SAT is a YES instance. Starting with $i = 1$, repeat the following firing sequence for each i between 1 and n. If q_i is $\top$ in the witnessing satisfying assignment, fire t_i else fire f_i. Then use the token thus put into $t_{\uparrow part(i)}/f_{\uparrow part(i)}$ respectively to shift a token one place to the right in Fig. 4.2 and put a token back in the place s. Continue with the next i. Since the witnessing assignment meets the target in each partition, we will have one token each in the places $tr_1^{tg(1)}, \ldots, tr_e^{tg(e)}, fl_1^{n[1]-tg(1)}, \ldots, fl_e^{n[e]-tg(e)}$. In addition, there will be a token in $x_i/\overline{x}_i$ iff the witnessing assignment set q_i to $\top/\bot$ respectively. Since this witnessing assignment satisfies all the clauses of the CNF formula, we can move the initial token in C_1 to C_{m+1} using the transitions in Fig. 4.3. Now, the transition in Fig. 4.4 can be fired to get a token into the place g. Now, the only tokens left are those in the places s and g, and those in $x_i/\overline{x}_i$. We can remove the tokens in $x_i/\overline{x}_i$ by firing td_i/fd_i to reach the final marking.

Suppose the required final marking is reachable in the constructed Petri net. Since a token has to be added to the place g to reach the final marking and the transition in Fig. 4.4 is the only transition that can add tokens to g, all input places of that transition must receive a token. The only way to get a token in places $tr_\rho^{tg(\rho)}$ is to shift the initial token in the place tr_ρ^0 $tg(\rho)$ times. This requires exactly $tg(\rho)$ tokens in the place $t_{\uparrow\rho}$. A similar argument holds for getting a token in $fl_\rho^{n[\rho]-tg(\rho)}$. Since the only way to add a token to $t_{\uparrow\rho}/f_{\uparrow\rho}$ is to fire transitions t_i/f_i (such that $part(i) = \rho$), the only way to get a token each in $tr_1^{tg(1)}, \ldots, tr_e^{tg(e)}, fl_1^{n[1]-tg(1)}, \ldots, fl_e^{n[e]-tg(e)}$ is to fire either t_i or f_i for each i between 1 and n. Consider any firing sequence reaching the required final marking. Consider the truth assignment to $q_1, \ldots, q_n$ that assigns $\top$ to exactly those variables q_i such that t_i was fired in the firing sequence. This truth assignment meets the target for each part since this firing sequence adds one token each to the places $tr_1^{tg(1)}, \ldots, tr_e^{tg(e)}, fl_1^{n[1]-tg(1)}, \ldots, fl_e^{n[e]-tg(e)}$. To reach the final marking, a token is also required at the place C_{m+1}. The only way to get this token is to shift the initial token in C_1 to C_{m+1} through the transitions in Fig. 4.3. This means that every clause is satisfied by the truth assignment we constructed. □

It is now left to prove that the pathwidth of the flow graph of the constructed 1-safe net is a function of the parameters of the given p-Pw-Sat instance.

Lemma 4.4. *Suppose a given instance of* p-Pw-Sat *has a CNF formula whose primal graph has pathwidth* pw *and* e *parts. Then, the flow graph of the* 1*-safe net constructed as described above has pathwidth at most* $3pw + 4e + 7$.

Proof. We show a path decomposition of the flow graph of the net. Call the set of places $\{s, g, C_{m+1}, t_{\uparrow 1}, \ldots, t_{\uparrow e}, f_{\uparrow 1}, \ldots, f_{\uparrow e}, tr_1^{tg(1)}, \ldots, tr_e^{tg(e)}, fl_1^{n[1]-tg(1)}, \ldots, fl_e^{n[e]-tg(e)}\}$ as P_1. Consider an optimal path decomposition of the primal graph of the CNF formula. In every bag, replace every occurrence of each q_i by the set $\{q_i, x_i, \overline{x}_i\} \cup P_1$.

Let $C_1, \ldots, C_m$ be the clauses in the path decomposition order as explained in the beginning of this sub-section. We will first show that places representing clauses can be added to the bags of the above decomposition without increasing their size much, while maintaining the invariant that all bags containing any one place are connected in the decomposition. We will do this by *augmenting* existing bags with new elements: if B is any bag in the decomposition and p is an element not in B, augmenting B with p means creating a new bag B' immediately to the left of B containing p in addition to the elements in B. We will call the new bag B' thus created an augmented bag. Perform the following operation in increasing order for each j between 1 and m: if B is the first non-augmented bag from left to contain all literals of the clause C_j, augment B with C_j.

There will be m new bags created due to the above augmentation steps. Due to the path decomposition ordering of $C_1, \ldots, C_m$, the augmented bag containing C_{j+1} occurs to the right of the augmented bag containing C_j for each j, $1 \leq j < m$. There might be some non-augmented bags between the augmented bags containing C_j and C_{j+1}. If so, add C_j to such non-augmented bags. Now, to every bag, if it contains C_j for some j between 1 and m, add C_{j+1}. It is routine to verify the following properties of the sequence we have with the bags modified as above.

- Each bag has at most $3pw + 4e + 8$ elements.
- The set of bags containing any one element forms a contiguous sub-sequence.
- Every vertex and edge in any subgraph induced by the parts of the net in Fig. 4.1, Fig. 4.3, and Fig. 4.4 is contained in some bag.

To account for the subgraph induced by the parts of the net in Fig. 4.2, we append the following sequence of bags for each ρ between 1 and e:

$$(\{tr_\rho^0, tr_\rho^1\} \cup P_1) - (\{tr_\rho^1, tr_\rho^2\} \cup P_1) - \cdots - (\{tr_\rho^{tg(\rho)-1}, tr_\rho^{tg(\rho)}\} \cup P_1) -$$
$$(\{fl_\rho^0, fl_\rho^1\} \cup P_1) - (\{fl_\rho^1, fl_\rho^2\} \cup P_1) - \cdots - (\{fl_\rho^{n[\rho]-tg(\rho)-1}, fl_\rho^{n[\rho]-tg(\rho)}\} \cup P_1)$$

The resulting sequence of bags is a path decomposition of the flow graph of the Petri net, whose width is at most $3pw + 4e + 7$. □

Lemma 4.3 and Lemma 4.4 together give a parameterized reduction from p-Pw-Sat to reachability in 1-safe nets with the pathwidth of the flow graph of the net as parameter. We thus get the following theorem.

Theorem 4.5. *With the pathwidth of the flow graph of a* 1*-safe Petri net as parameter, reachability is* W[1]*-hard.*

In the above reduction, it is enough to check if in the constructed 1-safe net, we can reach a marking that has a token at the place g. This can be expressed as a coverability problem, CTL model checking and also as complement of LTL/MSO model checking problems with formulas of constant size. Hence, we get the following theorem.

Theorem 4.6. *With the pathwidth of the flow graph of a* 1*-safe Petri net as parameter, coverability, CTL and the complement of LTL/MSO model checking problems are* W[1]*-hard, even when formula sizes are constant.*

4.3.1 Treewidth, Pathwidth and Graph Pebbling Problems

The techniques used in the above lower bound proof can be easily translated to some graph pebbling problems considered in [27]. As conjectured in [27, section 5], we prove that Signed Digraph Pebbling parameterized by treewidth is W[1]-hard. In [27], this problem is referred to as Signed Digraph Pebbling I.

Definition 4.7 (Signed Digraph Pebbling,[27])**.** Instance: *A red/blue bipartite digraph $D = (V, A)$ for which the vertex set V is partitioned $V = R \cup B$ into two partitions, and also the arc set A is partitioned into two partitions $A = A^+ \cup A^-$.*
Question: *Starting from the* start state *where there are no pebbles on any of the red vertices, is it possible to reach the* finish state *where there are pebbles on all of the red vertices, by a series of moves of the following form?* A legal move: *If b is a blue vertex such that for all u such that $(u, b) \in A^+$, u is pebbled, and for all u such that $(u, b) \in A^-$, u is not pebbled (in which case we say that b is* enabled*), then the set of vertices u such that $(b, u) \in A^+$ are reset by making them all pebbled, and the set of all vertices u such that $(b, u) \in A^-$ are reset by making them all unpebbled.*

Corollary 4.8. *Parameterized by pathwidth,* Signed Digraph Pebbling *is* W[1]*-hard.*

Proof. We will reduce p-Pw-Sat to Signed Digraph Pebbling. First reduce the given p-Pw-Sat instance to an 1-safe net as shown above. From this 1-safe net, construct an instance of Signed Digraph Pebbling as follows. Let the set of all places form the set of red vertices R and the set of all transitions form the set of blue vertices B. The arcs of the Signed Digraph Pebbling instance are as follows.

1. If $Pre(p, t) = 1$ in the 1-safe net, draw an A^+ arc from p to t in the Signed Digraph Pebbling instance.

2. If $Pre(p,t) = 1$ and $Post(p,t) = 0$, draw an A^- arc from t to p.

3. If $Pre(p,t) = 0$ and $Post(p,t) = 1$, draw an A^+ arc from t to p.

Suppose that in the 1-safe net, $M_1 \stackrel{t}{\Longrightarrow} M_2$. It is clear that the constructed Signed Digraph Pebbling instance in the state where precisely those red vertices are pebbled that have a token in M_1 enables the blue vertex t, and can move to the state where precisely those red vertices are pebbled that have a token in M_2. Add a special blue vertex b_1 with A^+ arcs from b_1 to $q_1, q_2, \ldots, q_n, tr_1^0, \ldots, tr_e^0, fl_1^0, \ldots, fl_e^0, C_1$ and s. Add A^- arcs from all red vertices to b_1. In the start state where there no pebbles at all, b_1 is the only blue vertex enabled. The blue vertex b_1 is enabled only in the start state. Upon performing the legal move using b_1 from the start state, we will reach a state in which precisely those red vertices are pebbled that have a token in the initial marking of the 1-safe net. From this state, there is at least one pebbled red vertex in any reachable state, so b_1 is never enabled again. From this state, we can reach a state with the red vertex g pebbled iff the given p-Pw-Sat instance is a Yes instance. Add another special blue vertex b_2 with an A^+ arc from the red vertex g to b_2. Add A^+ arcs from b_2 to all red vertices. All blue vertices except b_1 and b_2 unpebble at least one red vertex. Hence, the only way to reach the finish state (where all red vertices must be pebbled) from the start state is to enable b_2. The only way to enable b_2 is to reach a state where the red vertex g is pebbled. Hence, the constructed Signed Digraph Pebbling instance is a Yes instance iff the given p-Pw-Sat instance is a Yes instance.

To complete the reduction, it only remains to show that the pathwidth of the Signed Digraph Pebbling instance is bounded by the pathwidth of the flow graph the intermediate 1-safe net. Consider an optimal path decomposition of this flow graph. For every transition t, the set of all input and output places of t forms a clique in the flow graph. Hence, there will be at least one bag B in the path decomposition containing all these places. Create an extra bag B' adjacent to B containing all elements of B and also the blue vertex corresponding to t. After doing this for each transition, add the vertices b_1 and b_2 to all bags. The resulting decomposition is a path decomposition of the Signed Digraph Pebbling instance. Its width is at most 3 more than the pathwidth of the graph flow the 1-safe net. □

4.4 Benefit Depth and 1-safe Petri Nets

In this section, we will introduce a parameter called benefit depth and show that benefit depth as a parameter is not helpful for 1-safe Petri nets, by showing W[1]-hardness results. In Chapter 6, we give a positive result using benefit depth as a parameter for general Petri nets.

Definition 4.9. *Given a place p, the set of places $Ben(p) \subseteq P$ and the set of transitions $T_{ben}(p) \subseteq T$ benefited by p are the smallest sets that satisfy the following properties:*

1. *$p \in Ben(p)$.*

2. *If some $p' \in Ben(p)$ and there is a transition t with $Pre(p',t) \geq 1$, then $t \in T_{ben}(p)$.*

3. *If some transition $t \in T_{ben}(p)$ and there is a place p'' such that $Post(p'',t) \geq 1$, then $p'' \in Ben(p)$.*

The benefit depth of a 1-safe Petri net is defined as $\max_{p \in P}\{|Ben(p)|\}$.

Intuitively, places in $Ben(p)$ are those that can get a token from p, may be through a sequence of many transitions. Benefit depth can be thought of as a generalization of the out-degree in directed graphs.

To show that with benefit depth as parameter, reachability in 1-safe nets is W[1]-hard, we will show a parameterized reduction from the constraint satisfaction problem (Csp). The parameterized Csp we are interested in is defined as follows:

p-Csp	
Instance:	A set of variables $\{q_1, \ldots, q_n\}$, domain $\{1, \ldots, dom\}$ and a set of constraints $\mathcal{C}$. Each constraint $C \in \mathcal{C}$ is a pair (S, R), where S, the *constraint scope*, is a non-empty sequence of variables from $\{q_1, \ldots, q_n\}$, and R, the *constraint relation*, is a relation over $\{1, \ldots, dom\}$, whose arity matches the length of S.
Parameter:	The size of the domain dom and the maximum number deg of constraints in the scopes of which any one variable can occur.
Problem:	Decide if there is an assignment $s : \{q_1, \ldots, q_n\} \to \{1, \ldots, dom\}$ such that for each constraint $C = (q_i, \ldots, q_{i'}, R) \in \mathcal{C}$, $(s(q_i), \ldots, s(q_{i'})) \in R$.

This parameterized version of Csp is W[1]-hard [88, Corollary 2]. Given an instance of Csp with domain size dom, degree deg, n variables and m constraints, we construct a 1-safe net with the following places.

1. For every variable q_i, a place q_i.
2. For every constraint C_j where j is between 1 and m, a place C_j.
3. For every i between 1 and n, for every domain element d between 1 and dom, for every constraint C_j in which q_i appears, the place $q[i]^d_j$.
4. One place g for checking that all constraints are satisfied.

We assume without loss of generality that every variable occurs in at least one constraint. Construction of the 1-safe net is illustrated in the following diagrams. For every variable q_i and domain value d (between 1 and dom), part of the net shown in Fig. 4.5 is constructed. Intuitively, the transition t^d_i is fired to assign domain value d to q_i. In Fig. 4.5,

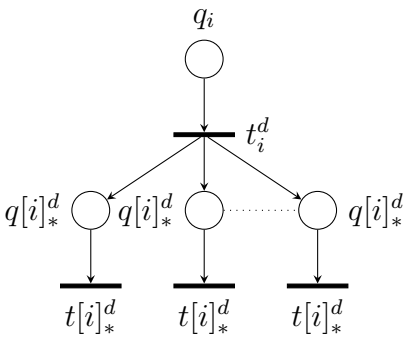


Figure 4.5: Part of the net for every variable q_i and domain value d

the set of places labelled by $q[i]^d_*$ should be understood to stand for the set of places $\{q[i]^d_j \mid q_i \text{ occurs in constraint } C_j\}$. The set of transitions labelled $t[i]^d_*$ should be similarly understood. For every constraint C_j and every admissible tuple of domain values for C_j, part of the net shown in Fig. 4.6 is constructed. In Fig. 4.6, it is assumed that the constraint C_j consists of variables q_1, q_2 and q_3 and that $(3, 5, 6)$ is an admissible tuple for this constraint. Finally, the part of the net in Fig. 4.7 verifies that all constraints are satisfied. The initial marking has 1 token each in each of the places $q_1, \ldots, q_n$ and 0 tokens in all other places. The final marking to be reached is 1 token at the place g and 0 tokens in all other places.

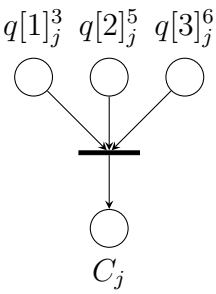


Figure 4.6: Part of the net for every constraint C_j and every admissible tuple

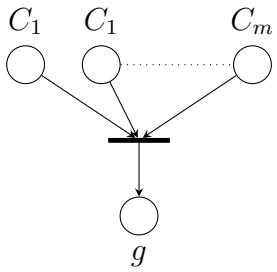


Figure 4.7: Part of the net to check that all constraints are satisfied

Lemma 4.10. *Given a* Csp *instance of domain size dom and degree deg, the benefit depth of the* 1*-safe net constructed above is at most* $2 + deg(dom + 1)$*. The given* Csp *instance is satisfiable iff the required final marking is reachable from the initial marking in the constructed* 1*-safe net.*

Proof. Maximum number of places are benefited by some place in $\{q_1, \ldots, q_n\}$. Any place q_i can benefit itself, the place g, all places $\{q[i]^1_1, \ldots, q[i]^{dom}_{deg}\}$ and at most deg places among $\{C_1, \ldots, C_m\}$. This adds up to at most $2 + deg(dom + 1)$.

Suppose the given Csp instance is satisfiable. For each variable q_i, if d is the domain value assigned to q_i by the satisfying assignment, fire the transition t^d_i shown in Fig. 4.5. Since the satisfying assignment satisfies all the constraints, the transitions shown in Fig. 4.6 can be fired to get a token into each of the places $C_1, \ldots, C_m$. Then the transition shown in Fig. 4.7 can be fired to get a token in the place g. Any tokens remaining in places $q[i]^d_*$ can be removed by firing transitions $t[i]^d_*$ shown in Fig. 4.5. Now, the token in the place g is the only token in the entire net and this is the final marking required to be reached.

Suppose the required final marking is reachable in the constructed 1-safe net. Consider any firing sequence reaching the required final marking. Since the final marking needs a token in the place g and the only transition that can add token to g is the one shown in Fig. 4.7, the firing sequence fires this transition. For this transition to be enabled, a token needs to be present in each of the places $C_1, \ldots, C_m$. These tokens can only be added by firing transitions shown in Fig. 4.6. To fire these transitions, tokens needs to be present in the places $q[i]^d_*$. To generate these tokens, the firing sequence would have to fire some transition t^d_i for each i between 1 and n. Consider the assignment that assigns domain value d to q_i iff the firing sequence fired t^d_i. By construction of the Petri net, this assignment satisfies all the constraints. □

Since the 1-safe net described above can be constructed in time polynomial in the size of the given Csp instance, Lemma 4.10 shows that this reduction is a parameterized reduction

from CSP (with *dom* and *deg* as parameters) to reachability in 1-safe nets (with benefit depth as the parameter). We thus get the following theorem.

Theorem 4.11. *With benefit depth as the parameter, reachability in* 1*-safe Petri nets is* W[1]*-hard.*

Again in the above reduction, it is enough to check if in the constructed 1-safe net, we can reach a marking that has a token at the place g. This can be expressed as a coverability problem, CTL model checking and also as the complement of LTL/MSO model checking problem with a formula of constant size. Hence, we get the following theorem.

Theorem 4.12. *With benefit depth as the parameter, coverability, CTL and the complement of LTL/MSO model checking problems are* W[1]*-hard, even when formula sizes are constant.*

4.5 Vertex Cover and Model Checking 1-safe Petri Nets

A **vertex cover** $VC \subseteq V$ of a graph $G = (V, E)$ is a subset of vertices such that for every edge in E, at least one of its vertices is in VC. The **vertex cover number** of a graph is the size of a smallest vertex cover. In this section, we will show that with the vertex cover number of the flow graph of the given 1-safe Petri net and size of the given LTL/MSO formula as parameters, checking whether the given net is a model of the given formula is FPT. With vertex cover number as the only parameter, we cannot hope to get this kind of tractability:

Proposition 4.13. *Model checking LTL (and hence MSO) formulas on* 1*-safe Petri nets whose flow graph has constant vertex cover number is* CO-NP*-hard. Model checking CTL formulas on* 1*-safe Petri nets whose flow graph has constant vertex cover number is* CO-NP*-hard.*

Proof. We give a reduction from the propositional logic validity problem. Let $\mathcal{F}$ be a propositional formula over variables $q_1, \ldots, q_n$. Consider the 1-safe net $\mathcal{N}_\mathcal{F}$ shown in Fig. 4.8. The

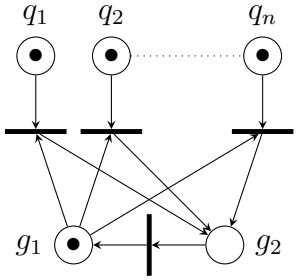


Figure 4.8: The net $\mathcal{N}_\mathcal{F}$ associated with a propositional formula $\mathcal{F}$

initial marking consists of 0 tokens in g_2 and 1 token each in all other places. The flow graph of $\mathcal{N}_\mathcal{F}$ has a vertex cover of size 2 ($\{g_1, g_2\}$). Every marking M reachable in $\mathcal{N}_\mathcal{F}$ defines an assignment to the variables used in $\mathcal{F}$: $q_i = \top$ iff $M(q_i) = 1$. Every assignment can be represented by some reachable marking in this way. We claim that $\mathcal{F}$ is valid iff $\mathcal{N}_\mathcal{F}$ is a model of the LTL formula $\neg(\top\ U\ \neg\mathcal{F})$. If $\mathcal{F}$ is valid, then none of the markings reachable in $\mathcal{N}_\mathcal{F}$ satisfies $\neg\mathcal{F}$. Hence, $\mathcal{N}_\mathcal{F}$ is a model of the LTL formula $\neg(\top\ U\ \neg\mathcal{F})$. On the other hand, if $\mathcal{N}_\mathcal{F}$ is a model of $\neg(\top\ U\ \neg\mathcal{F})$, then none of the markings reachable in $\mathcal{N}_\mathcal{F}$ satisfies $\neg\mathcal{F}$. Hence, $\mathcal{F}$ is valid. This means that model checking LTL (and hence MSO) is CO-NP-hard even when the flow graph has constant vertex cover number.

We claim that $\mathcal{F}$ is valid iff $\mathcal{N}_\mathcal{F}$ is a model of the CTL formula $\neg E[\top\ U\ \neg\mathcal{F}]$. If $\mathcal{F}$ is valid, then there is no run along which a marking defines an assignment satisfying $\neg\mathcal{F}$. Hence, $\mathcal{N}_\mathcal{F}$ is a model of the CTL formula $\neg E[\top\ U\ \neg\mathcal{F}]$. If $\mathcal{N}_\mathcal{F}$ is a model of the CTL formula $\neg E[\top\ U\ \neg\mathcal{F}]$, then none of the reachable markings define an assignment satisfying $\neg\mathcal{F}$. Hence, $\mathcal{F}$ is valid. This means that model checking CTL is Co-NP-hard even when the flow graph has constant vertex cover number. □

Since a run of a 1-safe net $\mathcal{N}$ with set of places P is a sequence of subsets of P, we can think of such sequences as strings over the alphabet $\mathscr{P}(P)$ (the power set of P). It is known [12, 97] that with any LTL or MSO formula ϕ, we can associate a finite state automaton $\mathcal{A}_\phi$ over the alphabet $\mathscr{P}(P)$ accepting the set of finite strings which are its models, as well as a finite state Büchi automaton $\mathcal{B}_\phi$ accepting the set of infinite string models.

Given a Petri net $\mathcal{N}$, consider its flow graph $G(\mathcal{N})$. Any vertex cover of $G(\mathcal{N})$ should include all vertices that have self loops. Suppose VC is a vertex cover for $G(\mathcal{N})$. We use the fact that if $v_1, v_2 \notin VC$ are two vertices not in VC that have the same set of neighbours, v_1 and v_2 have similar properties. This has been used to obtain Fpt algorithms for many hard problems (e.g. [33]).

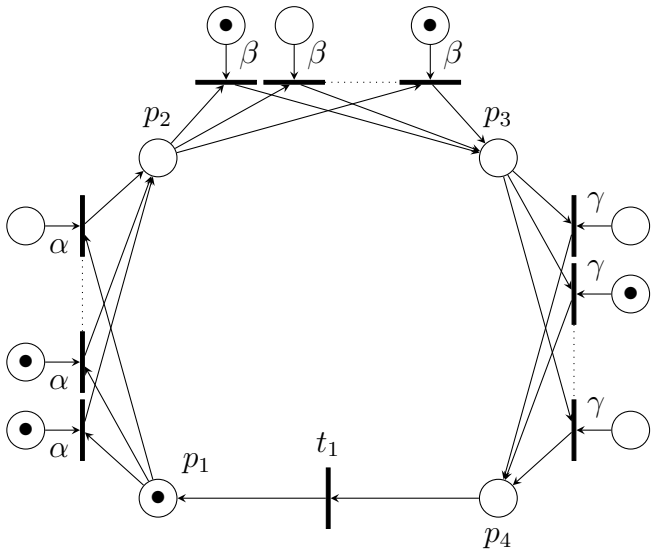


Figure 4.9: An example of a system with small vertex cover

Figure 4.9 shows the schematic of a simple manufacturing system modelled as a 1-safe Petri net. Starting from p_1, it picks up one unit of a raw material α and goes to p_2, then picks up raw material β, then γ. Transition t_1 does some processing and then the system starts from p_1 again. Suppose we want to make sure that whenever the system picks up a unit of raw material β, it is processed immediately. In other words, whenever the system stops at a marking where no transitions are enabled, there should not be a token in p_3. This can be checked by verifying that all finite maximal sequences satisfy the formula $\forall x((\forall y \quad y \leq x) \Rightarrow \neg p_3(x))$. The satisfaction of this formula depends only on the number of units of raw materials α, β and γ at the beginning, i.e., the number of tokens at the initial marking. The naive approach of constructing the whole reachability graph results in an exponentially large state space, due to the different orders in which the raw materials of each type can be drawn. If we want to reason about only the central system (which is the vertex cover $\{p_1, p_2, p_3, p_4\}$ in the above system), it turns out that we can ignore the order and express the requirements on the numbers by integer linear constraints. Roughly speaking, our Fpt algorithm works well for systems which have a small "core" (vertex cover), a small number of "interface types"

with this core, but any number of component processes using these interface types to interact with the core. Thus, we can have a large amount of conflict and concurrency but a limited amount of causality.

Definition 4.14. *Let VC be a vertex cover of $G(\mathcal{N})$. The VC-**neighbourhood** of a transition t is the ordered pair $({}^\bullet t \cap VC, t^\bullet \cap VC)$. We denote by l the number of different VC-neighbourhoods.*

Definition 4.15. *Suppose $\mathcal{N}$ is a Petri net with l neighbourhoods for vertex cover VC, and $p \notin VC$. The VC-**interface** $I[p]$ of p is defined as the function $I[p] : \{1, \ldots, l\} \to \mathscr{P}(\{-1, 1\})$, where for every j between 1 and l and every $w \in \{1, -1\}$, there is a transition t_j of VC-neighbourhood j such that $w = -Pre(p, t_j) + Post(p, t_j)$ iff $w \in I[p](j)$.*

In the net in Fig. 4.9 with $VC = \{p_1, p_2, p_3, p_4\}$, all transitions labelled α have the same VC-neighbourhood and all the corresponding places have the same VC-interface. Since there can be $2k$ arcs between a transition and places in VC if $|VC| = k$, there can be at most 2^{2k} different VC-neighbourhoods of transitions.

Place p can have one incoming or one outgoing arc with each transition of the net. It cannot have both an incoming and an outgoing arc since in that case, p would have a self loop and would be in VC. If p' is another place not in VC, then no transition can have arcs to both p and p', since otherwise, there would have been an edge between p and p' in $G(\mathcal{N})$ and one of the places p and p' would have been in VC. Hence, places not in VC cannot interact with each other directly. Places not in VC can only interact with places in VC through transitions and there are at most l VC-neighbourhoods of transitions. Since $l \leq 2^{2k}$, there are at most $4^{2^{2k}}$ VC-interfaces. The set of interfaces is denoted by *Int*.

Proposition 4.16. *Let $\mathcal{N}$ be a 1-safe net with VC being a vertex cover of $G(\mathcal{N})$. Let $p_1, p_2, \ldots, p_i$ be places not in the vertex cover, all with the same interface. Let M be some marking reachable from the initial marking of $\mathcal{N}$. If $M(p_j) = 1$ for some j between 1 and i, then M does not enable any transition that adds tokens to any of the places $p_1, \ldots, p_i$.*

Proof. Suppose there is a transition t enabled at M that adds a token to $p_{j'}$ for some j' between 1 and i. Then there is a transition t' with the same neighbourhood as t (and hence enabled at M too) that can add a token to p_j. Firing t' from M will create 2 tokens at p_j, contradicting the fact that $\mathcal{N}$ is 1-safe. □

If the initial marking has tokens in many places with the same interface, then no transition can add tokens to any of those places until all the tokens in all those places are removed. Once all tokens are removed, one of the places can receive one token after which, no place can receive tokens until this one is removed. All these places have the same interface. Thus, a set of places with the same interface can be thought of as an initial storehouse of tokens, after depleting which it can be thought of as a single place. However, a formula in our logic can reason about individual places, so we still need to keep track of individual places that occur in the formula.

Let $\mathcal{N}$ be a 1-safe net such that $G(\mathcal{N})$ has a vertex cover VC of size k. Suppose ϕ is a formula and we have to check if $\mathcal{N}$ is a model of ϕ. For each interface I, let $P_I \subseteq P$ be the places not in VC with interface I. If $P_I \setminus P_\phi \neq \emptyset$ (i.e., if there are places in P_I that are not in ϕ), designate one of the places in $P_I \setminus P_\phi$ as p_I. Define the set of special places $\mathcal{S} = VC \cup P_\phi \cup \{p_I \in P_I \setminus P_\phi \mid I \text{ is an interface and } P_I \setminus P_\phi \neq \emptyset\}$. Note that $|\mathcal{S}| \leq k + |\phi| + 4^{2^{2k}}$. Since this is a function of the parameters of the input instance, we will treat it as a parameter.

We need a structure that keeps track of changes in places belonging to $\mathcal{S}$, avoiding a construction involving all reachable markings. This can be done by a finite state machine whose states are subsets of $\mathcal{S}$. Transitions of the Petri net that only affect places in $\mathcal{S}$ can

be simulated by the finite state machine with its usual transitions. To simulate transitions of the net that affect places outside $\mathcal{S}$, we need to impose some conditions on the number of times transitions of the finite state machine can be used. The following definition formalizes this. For a marking M of $\mathcal{N}$, let $M\lceil\mathcal{S} = \{p \in \mathcal{S} \mid M(p) = 1\}$.

Definition 4.17. *Given a 1-safe net $\mathcal{N}$ with initial marking M_0 and $\mathcal{S}$ defined from ϕ as above, the **edge constrained automaton** $\mathcal{A}_\mathcal{N} = (Q_\mathcal{N}, \Sigma, \delta_\mathcal{N}, \ell, F_\mathcal{N})$ is a structure defined as follows. $Q_\mathcal{N} = \mathscr{P}(\mathcal{S})$ and $\Sigma = Int \cup \{\perp\}$ (recall that Int is the set of interfaces in $\mathcal{N}$). The transition relation $\delta \subseteq Q_\mathcal{N} \times \Sigma \times Q_\mathcal{N}$ is such that for all $P_1, P_2 \subseteq \mathcal{S}$ and $I \in Int \cup \{\perp\}$, $(P_1, I, P_2) \in \delta$ iff there are markings M_1, M_2 and a transition t of $\mathcal{N}$ such that*

- *$M_1\lceil\mathcal{S} = P_1$, $M_2\lceil\mathcal{S} = P_2$ and $M_1 \overset{t}{\Longrightarrow} M_2$,*
- *t removes a token from a place $p \in P \setminus \mathcal{S}$ of interface I if $I \in Int$ and*
- *t does not have any of its input or output places in $P \setminus \mathcal{S}$ if $I = \perp$.*

*The **edge constraint** $\ell : Int \to \mathbb{N}$ is given by $\ell(I) = |\{p \in P_I \setminus \mathcal{S} \mid M_0(p) = 1\}|$. A subset $P_1 \subseteq \mathcal{S}$ is in $F_\mathcal{N}$ iff for every marking M with $M\lceil\mathcal{S} = P_1$, the only transitions enabled at M remove tokens from some place not in $\mathcal{S}$.*

Intuitively, the edge constraint ℓ defines an upper bound on the number of times those transitions can be used that reduce tokens from places not in $\mathcal{S}$.

Definition 4.18. *Let $\mathcal{A}_\mathcal{N}$ be an edge constrained automaton as in Definition 4.17 and let $\pi = P_0P_1\ldots$ be a finite or infinite word over $\mathscr{P}(\mathcal{S})$. Then π is a valid run of $\mathcal{A}_\mathcal{N}$ iff for every position $i \geq 1$ of π, we can associate an element $I_i \in Int \cup \{\perp\}$ such that*

- *for every position $i \geq 1$ of π, $(P_{i-1}, I_i, P_i) \in \delta$ and*
- *for every $I \in Int$, $|\{i \geq 1 \mid I_i = I\}| \leq \ell(I)$.*
- *if π is finite and P_i is the last element of π, then $P_i \in F_\mathcal{N}$ and for every interface $I \in Int$ with a marking $M_i\lceil\mathcal{S} = P_i$ enabling some transition that removes tokens from some place in $P_I \setminus \mathcal{S}$, $|\{i \geq 1 \mid I_i = I\}| = \ell(I)$.*

Next we have a run construction lemma.

Lemma 4.19. *Let $\mathcal{N}$ be a 1-safe net with initial marking M_0, ϕ be a formula and $\mathcal{A}_\mathcal{N}$ be as in Definition 4.17. For every infinite (maximal finite) run $\pi = M_0M_1\cdots$ of $\mathcal{N}$, there exists an infinite (finite) run $\pi' = M'_0M'_1\cdots$ such that the word $(M'_0\lceil\mathcal{S})(M'_1\lceil\mathcal{S})\cdots$ is a valid run of $\mathcal{A}_\mathcal{N}$ and for every position i of π, $M'_i\lceil P_\phi = M_i\lceil P_\phi$. If an infinite (finite) word $\pi = P_0P_1\cdots$ over $\mathscr{P}(\mathcal{S})$ is a valid run of $\mathcal{A}_\mathcal{N}$ and $P_0 = M_0\lceil\mathcal{S}$, then there is an infinite (finite maximal) run $M_0M_1\cdots$ of $\mathcal{N}$ such that $M_i\lceil\mathcal{S} = P_i$ for all positions i of π.*

Proof. Let $\pi = M_0M_1\cdots$ be an infinite or a maximal finite run of $\mathcal{N}$. For every interface $I \in Int$, perform the following steps: if for some marking M in the above run, $\{p \in P_I \mid M(p) = 1\} = \emptyset$, let M_I be the first such marking. By Proposition 4.16, no transition occurring before M_I will add any token to any place in P_I. If there is any transition occurring after M_I that adds/removes tokens from $P_I \setminus \mathcal{S}$, replace it with another transition with the same neighbourhood that adds/removes tokens from p_I. By Proposition 4.16, such a replacement will not affect any place in P_ϕ and the new sequence of transitions is still enabled at M_0. After performing this process for every interface $I \in Int$, let the new run be $\pi' = M'_0M'_1\cdots$. By construction, we have $M'_i\lceil P_\phi = M_i\lceil P_\phi$ for all positions $i \geq 0$ of π. If π is a maximal finite run, so is π'.

Now we will prove that the word $(M'_0\lceil\mathcal{S})(M'_1\lceil\mathcal{S})\cdots$ is a valid run of $\mathcal{A}_\mathcal{N}$. Suppose the sequence of transitions producing the run π' is $M'_0 \overset{t_1}{\Longrightarrow} M'_1 \overset{t_2}{\Longrightarrow} M'_2\cdots$. For each position $i \geq 1$ of this run, define $I_i \in \Sigma$ as follows:

- if t_i has all its input and output places among places $\mathcal{S}$, let $I_i = \perp$.
- if t_i removes a token from some place in $P_I \setminus \mathcal{S}$ for some interface I, let $I_i = I$. Due to the way π' is constructed, this kind of transition can only occur before the position of M_I and the number of such occurrences is at most $|\{p \in P_I \setminus \mathcal{S} \mid M_0(p) = 1\}| = \ell(I)$.

Due to the way π' is constructed, there will not be any transition that adds tokens to any place in $P_I \setminus \mathcal{S}$ for any interface I. By definition, it is clear that for every position $i \geq 1$ of π', $(M'_{i-1}\lceil\mathcal{S}, I_i, M'_i\lceil\mathcal{S}) \in \delta_{\mathcal{N}}$. In addition, for every interface $I \in Int$, we have $|\{i \geq 1 \mid I_i = I\}| \leq \ell(I)$. Hence, the word $(M'_0\lceil\mathcal{S})(M'_1\lceil\mathcal{S})\cdots$ is a valid run of $\mathcal{A}_{\mathcal{N}}$ if the word is infinite. If π' is finite, suppose M'_r is the last marking of the sequence π'. Suppose for some variety $I \in Int$, there is some marking M such that $M\lceil\mathcal{S} = M'_r\lceil\mathcal{S}$ and M enables some transition t that removes tokens from some place in $P_I \setminus \mathcal{S}$. Since M'_r does not enable any transition, all transitions (including t) removing tokens from some place in $P_I \setminus \mathcal{S}$ are disabled in M'_r. This means that every place in $P_I \setminus \mathcal{S}$ that had a token in M_0 has lost its token in M'_r. Since such loss of tokens can only happen by firing transitions that remove tokens from places in $P_I \setminus \mathcal{S}$, we have $|\{i \geq 1 \mid I_i = I\}| = \ell(I)$. Hence, to prove that $(M'_0\lceil\mathcal{S})(M'_1\lceil\mathcal{S})\cdots(M'_r\lceil\mathcal{S})$ is a valid run of $\mathcal{A}_{\mathcal{N}}$, it is left to show that $M'_r \in F_{\mathcal{N}}$. To see that this is true, observe that if some marking M with $M\lceil\mathcal{S} = M'_r$ enables a transition that does not remove any token from $P \setminus \mathcal{S}$, then so does M'_r, a contradiction.

Next, suppose $\pi = P_0P_1\cdots$ is an infinite or finite word that is a valid run of $\mathcal{A}_{\mathcal{N}}$ such that $P_0 = M_0\lceil\mathcal{S}$. For every position $i \geq 1$ of π, there are $I_i \in Int \cup \{\perp\}$, transition t'_i and markings M'_{i-1} and M'_i such that $(P_{i-1}, I_i, P_i) \in \delta_{\mathcal{N}}$, $M'_{i-1} \xRightarrow{t'_i} M'_i$, $M'_{i-1}\lceil\mathcal{S} = P_{i-1}$ and $M'_i\lceil\mathcal{S} = P_i$. Define transitions t_i as follows:

- If $I_i = \perp$, transition t'_i has all its input and output places in $\mathcal{S}$. Let $t_i = t'_i$.
- If $I_i = I \in Int$, transition t'_i removes a token from some place in $P_I \setminus \mathcal{S}$. Let t' be a transition of the same neighbourhood as t'_i that removes a token from some place $p_i \in \{p \in P_I \setminus \mathcal{S} \mid M_0(p) = 1\}$ such that no transition among $t_1, \ldots, t_{i-1}$ removes tokens from p_i. This is possible since, due to the validity of π in $\mathcal{A}_{\mathcal{N}}$, $|\{i' \geq 1 \mid I_{i'} = I\}| \leq |\{p \in P_I \setminus \mathcal{S} \mid M_0(p) = 1\}| = \ell(I)$. Let $t_i = t'$.

We will now prove by induction on i that there are markings $M_0, M_1, \ldots$ such that $M_0 \xRightarrow{t_1} M_1 \xRightarrow{t_2} \cdots \xRightarrow{t_i} M_i$ and $M_i\lceil\mathcal{S} = P_i$ for every position i of π.

Base case $i = 1$: If $I_1 = \perp$, the fact that $M_0\lceil\mathcal{S} = P_0$, $M'_0 \xRightarrow{t_1} M'_1$ and that t_1 has all its input and output places in $\mathcal{S}$ implies that $M_0 \xRightarrow{t_1} M_1$ for some M_1 such that $M_1\lceil\mathcal{S} = P_1$. If $I_i = I \in Int$, then t_1 removes a token from some place $p_1 \in P_I \setminus \mathcal{S}$. Again the fact that $M_0\lceil\mathcal{S} = P_0$ and $M'_0 \xRightarrow{t'_1} M'_1$ implies that $M_0 \xRightarrow{t_1} M_1$ for some M_1 such that $M_1\lceil\mathcal{S} = P_1$.

Induction step: If $I_{i+1} = \perp$, the fact that $M_i\lceil\mathcal{S} = P_i$, $M'_i \xRightarrow{t_{i+1}} M'_{i+1}$ and that t_{i+1} has all its input and output places in $\mathcal{S}$ implies that $M_i \xRightarrow{t_{i+1}} M_{i+1}$ for some M_{i+1} such that $M_{i+1}\lceil\mathcal{S} = P_{i+1}$. If $I_{i+1} = I \in Int$, then t_{i+1} removes a token from some place $p_{i+1} \in P_I \setminus \mathcal{S}$. Again the fact that $M_i\lceil\mathcal{S} = P_i$ and $M'_i \xRightarrow{t'_{i+1}} M'_{i+1}$ implies that $M_i \xRightarrow{t_{i+1}} M_{i+1}$ for some M_{i+1} such that $M_{i+1}\lceil\mathcal{S} = P_{i+1}$.

If π is a finite word, we have to prove that the run constructed above is a maximal finite run. Let M_r be the last marking in the sequence constructed above. We will prove that M_r does not enable any transition. Suppose some transition t is enabled at M_r. Since $M_r\lceil\mathcal{S} \in F_{\mathcal{N}}$, t removes a token from some place in $P_I \setminus \mathcal{S}$ for some variety I. Since $|\{i \geq 1 \mid I_i = I\}| = \ell(I)$, there are $\ell(I)$ transition occurrences among $t_1, \ldots, t_r$ that each remove a token from some place in $P_I \setminus \mathcal{S}$. Since there were exactly $\ell(I)$ places in $P_I \setminus \mathcal{S}$ that had

a token in M_0 and no other transition adds any token to any place in $P_I \setminus \mathcal{S}$, t can not be enabled at M_r. Hence, no transition is enabled at M_r. □

Lemma 4.19 implies that in order to check if $\mathcal{N}$ satisfies the formula ϕ, it is enough to check that all words accepted by $\mathcal{A}_\mathcal{N}$ satisfy ϕ. This can be done by checking that no finite sequence is accepted by both $\mathcal{A}_\mathcal{N}$ and $\mathcal{A}_{\neg\phi}$ and no infinite sequence is accepted by both $\mathcal{A}_\mathcal{N}$ and $\mathcal{B}_{\neg\phi}$. As usual, this needs a product construction. Automata $\mathcal{A}_{\neg\phi}$ and $\mathcal{B}_{\neg\phi}$ run on the alphabet $\mathscr{P}(P_\phi)$. Let $Q_\mathcal{A}$ and $Q_\mathcal{B}$ be the set of states of $\mathcal{A}_{\neg\phi}$ and $\mathcal{B}_{\neg\phi}$ respectively. Then, $\mathcal{A}_{\neg\phi} = (Q_\mathcal{A}, \mathscr{P}(P_\phi), \delta_\mathcal{A}, Q_{0\mathcal{A}}, F_\mathcal{A})$ and $\mathcal{B}_{\neg\phi} = (Q_\mathcal{B}, \mathscr{P}(P_\phi), \delta_\mathcal{B}, Q_{0\mathcal{B}}, F_\mathcal{B})$.

Definition 4.20. $\mathcal{A}_\mathcal{N} \times \mathcal{A}_{\neg\phi} = (Q_\mathcal{N} \times Q_\mathcal{A}, \Sigma, \delta^\mathcal{N}_\mathcal{A}, \{M_0 \lceil \mathcal{S}\} \times Q_{0\mathcal{A}}, F_\mathcal{N} \times F_\mathcal{A}, \ell)$, $\mathcal{A}_\mathcal{N} \times \mathcal{B}_{\neg\phi} = (Q_\mathcal{N} \times Q_\mathcal{B}, \Sigma, \delta^\mathcal{N}_\mathcal{B}, \{M_0 \lceil \mathcal{S}\} \times Q_{0\mathcal{B}}, Q_\mathcal{N} \times F_\mathcal{B}, \ell)$ *where*

$$((q_1, q_2), I, (q_1', q_2')) \in \delta^\mathcal{N}_\mathcal{A} \text{ iff } (q_1, I, q_1') \in \delta_\mathcal{N} \text{ and } (q_2, q_1 \cap P_\phi, q_2') \in \delta_\mathcal{A}$$
$$((q_1, q_2), I, (q_1', q_2')) \in \delta^\mathcal{N}_\mathcal{B} \text{ iff } (q_1, I, q_1') \in \delta_\mathcal{N} \text{ and } (q_2, q_1 \cap P_\phi, q_2') \in \delta_\mathcal{B}$$

An accepting path of $\mathcal{A}_\mathcal{N} \times \mathcal{A}_{\neg\phi}$ is a sequence $(q_0, q_0')I_1(q_1, q_1') \cdots I_r(q_r, q_r')$ which is $\delta^\mathcal{N}_\mathcal{A}$-respecting:

- $(q_0, q_0'), (q_1, q_1'), \ldots, (q_r, q_r') \in Q_\mathcal{N} \times Q_\mathcal{A}$,
- *the word $I_1 \cdots I_r \in \Sigma^*$ witnesses the validity of the run $q_0 q_1 \cdots q_r$ in $\mathcal{A}_\mathcal{N}$ (as in Definition 4.18) and*
- *the word $(q_0 \cap P_\phi) \cdots (q_r \cap P_\phi)$ is accepted by $\mathcal{A}_{\neg\phi}$ through the run $q_0' q_1' \cdots q_r' q_F'$ for some $q_F' \in F_\mathcal{A}$ with $(q_r', q_r \cap P_\phi, q_F') \in \delta_\mathcal{A}$.*

An accepting path of $\mathcal{A}_\mathcal{N} \times \mathcal{B}_{\neg\phi}$ is defined similarly.

Proposition 4.21. *A 1-safe net $\mathcal{N}$ with initial marking M_0 is a model of a formula ϕ iff there is no accepting path in $\mathcal{A}_\mathcal{N} \times \mathcal{A}_{\neg\phi}$ and $\mathcal{A}_\mathcal{N} \times \mathcal{B}_{\neg\phi}$.*

Proof. Suppose $\mathcal{N}$ is a model of ϕ. Hence, all maximal runs of $\mathcal{N}$ satisfy ϕ. We will prove that there is no accepting path in $\mathcal{A}_\mathcal{N} \times \mathcal{A}_{\neg\phi}$ and $\mathcal{A}_\mathcal{N} \times \mathcal{B}_{\neg\phi}$. Assume by way of contradiction that there is an accepting path $(q_0, q_0')I_1(q_1, q_1') \cdots I_r(q_r, q_r')$ in $\mathcal{A}_\mathcal{N} \times \mathcal{A}_{\neg\phi}$. By Definition 4.20, $q_0 q_1 \cdots q_r$ is a valid run of $\mathcal{A}_\mathcal{N}$. By Lemma 4.19, there is a finite maximal run $M_0 M_1 \cdots M_r$ of $\mathcal{N}$ with $M_i \lceil \mathcal{S} = q_i$ for all positions $0 \leq i \leq r$. By Definition 4.20, $(q_0 \cap P_\phi) \cdots (q_r \cap P_\phi)$ is accepted by $\mathcal{A}_{\neg\phi}$ and hence satisfies $\neg\phi$. Proposition 4.2 now implies that $M_0 M_1 \cdots M_r$ satisfies $\neg\phi$, a contradiction. The argument for $\mathcal{A}_\mathcal{N} \times \mathcal{B}_{\neg\phi}$ is similar.

Suppose $\mathcal{N}$ is not a model of ϕ. Suppose there is a finite maximal run $M_0 M_1 \cdots M_r$ of $\mathcal{N}$ that satisfies $\neg\phi$. By Lemma 4.19, there is a finite maximal run $\pi' = M_0' M_1' \cdots M_r'$ such that the word $(M_0' \lceil \mathcal{S})(M_1' \lceil \mathcal{S}) \cdots (M_r' \lceil \mathcal{S})$ is a valid run of $\mathcal{A}_\mathcal{N}$ and for every position i of π', $M_i' \lceil P_\phi = M_i \lceil P_\phi$. By Proposition 4.2, $(M_0' \lceil P_\phi)(M_1' \lceil P_\phi) \cdots (M_r' \lceil P_\phi)$ satisfies $\neg\phi$ and hence accepted by $\mathcal{A}_{\neg\phi}$, say with the run $q_0' q_1' \cdots q_r' q_F'$. Let the word $I_1 \cdots I_r \in \Sigma^*$ witness the validity of the run $(M_0' \lceil \mathcal{S})(M_1' \lceil \mathcal{S}) \cdots (M_r' \lceil \mathcal{S})$ in $\mathcal{A}_\mathcal{N}$, as in Definition 4.18. By Definition 4.20, the sequence $(M_0' \lceil P_\phi, q_0')I_1(M_1' \lceil P_\phi, q_1') \cdots I_r(M_r' \lceil P_\phi, q_r')$ is an accepting path of $\mathcal{A}_\mathcal{N} \times \mathcal{A}_{\neg\phi}$. The argument for maximal infinite runs is similar. □

To efficiently check the existence of accepting paths in $\mathcal{A}_\mathcal{N} \times \mathcal{A}_{\neg\phi}$ and $\mathcal{A}_\mathcal{N} \times \mathcal{B}_{\neg\phi}$, it is convenient to look at them as graphs, possibly with self loops and parallel edges. Let the set of states be the vertices and each entry of a transition (q, I_i, q') be an edge leaving q and entering q'. If there is a path μ in the graph from q to q', the number of times an edge e occurs in μ is denoted by $\mu(e)$. If $u \neq q, q'$ is some node occurring in μ, then the number of

edges of μ entering u is equal to the number of edges of μ leaving u. These conditions can be expressed as integer linear constraints.

$$\begin{aligned} \sum_{e \text{ leaves } q} \mu(e) - \sum_{e \text{ enters } q} \mu(e) &= 1 \\ \sum_{e \text{ enters } q'} \mu(e) - \sum_{e \text{ leaves } q'} \mu(e) &= 1 \\ u \neq q, q' : \sum_{e \text{ enters } u} \mu(e) &= \sum_{e \text{ leaves } u} \mu(e) \end{aligned} \tag{4.1}$$

In connected graphs, the above conditions are also sufficient for the existence of a path. The following generalization of Euler's theorem is also useful in other problems of general Petri nets [86, section 2.1].

Lemma 4.22 (Theorem 2.1,[86]). *In a directed graph $G = (V, E)$ (possibly with self loops and parallel edges), let $\mu : E \to \mathbb{N}$ be a function such that the underlying undirected graph induced by edges e such that $\mu(e) > 0$ is connected. Let $q \neq q'$ be two vertices. Then, there is a path from q to q' with each edge e occurring $\mu(e)$ times iff μ satisfies the constraints* (4.1) *above.*

If the beginning and end of a path are same (i.e., if $q = q'$), the conditions of (4.1) can be simplified as follows.

$$u \text{ is a vertex } : \sum_{e \text{ enters } u} \mu(e) = \sum_{e \text{ leaves } u} \mu(e) \tag{4.2}$$

Lemma 4.23 (Theorem 2.1,[86]). *In a directed graph $G = (V, E)$ (possibly with self loops and parallel edges), let $\mu : E \to \mathbb{N}$ be a function such that the underlying undirected graph induced by edges e such that $\mu(e) > 0$ is connected. Let q be a vertex such that there exists an edge e incident on q with $\mu(e) \neq 0$. Then, there is a loop from q back to q with each edge e occurring $\mu(e)$ times iff μ satisfies the constraints* (4.2) *above.*

Theorem 4.24. *Let $\mathcal{N}$ be a 1-safe net with initial marking M_0 and ϕ be a MSO formula. Parameterized by the vertex cover number of $G(\mathcal{N})$ and the size of ϕ, checking whether $\mathcal{N}$ is a model of ϕ is* FPT.

Proof. By Proposition 4.21, it is enough to check that there is no accepting paths in $\mathcal{A}_\mathcal{N} \times \mathcal{A}_{\neg\phi}$ and $\mathcal{A}_\mathcal{N} \times \mathcal{B}_{\neg\phi}$. To check the existence of accepting paths in $\mathcal{A}_\mathcal{N} \times \mathcal{B}_{\neg\phi}$, we have to check if from some initial state in $\{M_0\lceil\mathcal{S}\} \times Q_{0\mathcal{B}}$, we can reach some vertex in a maximal strongly connected component induced by $\perp$-labelled edges, which contains some states from $Q_\mathcal{N} \times F_\mathcal{B}$. For every such initial state q and a vertex q' in such a strongly connected component, check the feasibility of (4.1) along with the following constraint for each interface I:

$$\sum_{e \text{ is } I- \text{ labelled}} \mu(e) \leq \ell(I) \tag{4.3}$$

To check the existence of accepting paths in $\mathcal{A}_\mathcal{N} \times \mathcal{A}_{\neg\phi}$, check the feasibility of (4.1) and (4.3) for every state q in $\{M_0\lceil\mathcal{S}\} \times Q_{0\mathcal{A}}$ and every state (P_1, q'') in $F_\mathcal{N} \times Q_\mathcal{A}$ with some $q_F \in F_\mathcal{A}$ such that $(q'', P_1 \cap P_\phi, q_F) \in \delta_\mathcal{A}$. If some marking M with $M\lceil\mathcal{S} = P_1$ enables some transition removing a token from some place with interface I, then for each such interface, add the following constraint:

$$\sum_{e \text{ is } I- \text{ labelled}} \mu(e) = \ell(I) \tag{4.4}$$

This will ensure that all available tokens are used up so that only maximal runs are considered.

The variables in the above ILP instances are $\mu(e)$ for each edge e. The number of variables in each ILP instance is bounded by some function of the parameters. As ILP is FPT when parameterized by the number of variables [54, 64, 38], the result follows. □

4.6 Open Problems

With the vertex cover number of $G(\mathcal{N})$ and the size of a given CTL formula as parameters, the parameterized complexity of checking whether the 1-safe Petri net $\mathcal{N}$ satisfies the given CTL formula is open.

Chapter 5

Survey of General Petri Nets and Related Concepts

Works on logical characterization of seemingly unrelated models relate to languages of Petri nets and counter automata. In this chapter, we survey some of the logics whose satisfiability or provability relates to languages of Petri nets or counter automata in some way. The logics considered are LTL with freeze operators [22], $FO^2(\sim, <, +1)$ on data words [9], Existential Monadic Second Order (EMSO) logic of Petri net languages [79], provability in fragments of linear and relevance logic [94, 95, 96] and Presburger arithmetic with monotone transitive closure operator [85]. We also look at the coverability and boundedness of Branching Vector Addition Systems (BVAS) [20] and Yen's path logic for deciding the existence of certain Petri net paths [99, 4].

5.1 Petri Nets and Some Properties

Recall the definition of Petri net reachability, coverability and boundedness from Definition 4.1. In the Petri net shown in Fig. 5.1, the initial marking M_0 is given by $M_0(p_1) = 1$ and $M_0(p_2) = M_0(p_3) = 0$. If M_{cov} is defined as $M_{cov}(p_1) = M_{cov}(p_2) = 1$ and $M_{cov}(p_3) = 0$, then M_{cov} is not coverable since p_1 and p_2 cannot have tokens simultaneously in any reachable marking. Since for any $c \in \mathbb{N}$, the Petri net in Fig. 5.1 can reach a marking where p_3 has more than c tokens (by firing the sequence $t_1 t_2$ repeatedly), this Petri net is not bounded. Lipton proved both coverability and boundedness problems to be EXPSPACE-hard [66, 30]. Rackoff provided EXPSPACE upper bounds for both problems [83]. Lipton's EXPSPACE lower bound applies to the reachability problem too, and this is the best known lower bound. Though the reachability problem is known to be decidable [71, 58, 62, 65], no upper bound is known.

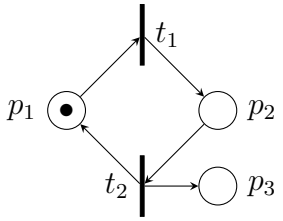


Figure 5.1: An example of a Petri net

5.2 Data Words

Since the semantics for LTL with freeze quantifier and $FO^2(\sim, <, +1)$ are through data words, we start with data words. Let Σ be a finite alphabet. A data word σ over Σ is a finite word $\mathrm{str}(\sigma)$ over Σ together with an equivalence relation $\sim^\sigma$ on its positions. Equivalently, a data word σ can also be thought of as a finite sequence of letters from Σ in which, every position is also associated with a datum from an infinite domain such that any two positions have the same datum iff they are equivalent with respect to $\sim^\sigma$.

5.2.1 LTL Over Data Words

All notation and results of this subsection are from [22]. To reason about data words, the syntax of LTL is extended by freeze operators. In the following syntax, atomic propositions a are elements of Σ and r ranges over positive integers.

$$\phi \ ::= \ a \mid \top \mid \neg\phi \mid \phi \wedge \psi \mid X\phi \mid \phi U \psi \mid X^{-1}\phi \mid \phi U^{-1}\psi \mid \ \downarrow_r \phi \mid \ \uparrow_r$$

The semantics for LTL with freeze operators is defined over data words. A *register valuation* v for a data word σ is a finite partial map from positive integers to the equivalence classes of the positions of σ with respect to $\sim^\sigma$. Given a data word σ, a position i and a register valuation v, the definition of $\sigma, i \models_v \phi$ is the usual definition for Boolean operators and temporal operators X and U. The symbol $[i]_\sim$ stands for the $\sim^\sigma$ equivalence class of the position i of σ. The symbol $v[r \to [i]_\sim]$ stands for the register valuation that is same as v except that r is mapped to $[i]_\sim$.

$\sigma, i \models_v \ a \quad$ iff $\quad \sigma(i) = a$

$\sigma, i \models_v \ X^{-1}\phi \quad$ iff $\quad i - 1 \geq 0$ and $\sigma, i-1 \models_v \ \phi$

$\sigma, i \models_v \ \phi U^{-1} \psi \quad$ iff for some $j \leq i$, $\sigma, j \models_v \ \psi$ and for all $j < j' \leq i$, $\sigma, j' \models_v \ \phi$

$\sigma, i \models_v \downarrow_r \phi \quad$ iff $\quad \sigma, i \models_{v[r \to [i]_\sim]} \phi$

$\sigma, i \models_v \uparrow_r \quad$ iff $\quad r \in dom(v)$ and $i \in v(r)$

Let $LTL^{\downarrow}_n(TO)$ be the fragment of LTL with n registers (i.e., r ranges over $\{1, \ldots, n\}$) and temporal operators in TO.

Theorem 5.1 ([22]). *Satisfiability for $LTL^{\downarrow}_1(X, U)$ is decidable and not primitive recursive.*

The decidability is shown in [22, Theorem 4.4] and non-primitive recursive lower bound in [22, Theorem 5.2]. The decidability is shown in two stages — first, satisfiability is reduced to the non-emptiness problem for alternating automata with 1 register (an extension of the standard translation of LTL to alternating automata, e.g., [97]). Register automata are finite state automata augmented with registers. Transitions can store the datum in the current position into a register. They can also compare the datum of the current position with any register for equality. The second step of decidability consists of reducing non-emptiness of alternating automata with 1 register to non-emptiness for incrementing counter automata. Incrementing counter automata are finite state automata augmented with a finite set of counters that can store natural numbers. Transitions can increase the value of a counter by one or decrease it by one if the current value is not zero. Zero-testing transitions are allowed, which can be executed only if a counter specified in the transition has value zero. The counters may have incrementing errors, that is, the value of any counter may erroneously increase at any time. Non-primitive recursive lower bound of Theorem 5.1 is shown by a logspace reduction from incrementing counter automata.

Small extensions of the logic lead to undecidability.

Theorem 5.2 ([22]). *The satisfiability problems for $LTL^{\downarrow}_2(X, U)$ and $LTL^{\downarrow}_1(X, U, F^{-1})$ are undecidable. The emptiness problem for 2-way alternating register automata is undecidable.*

5.2.2 Two Variable Logic on Data Words

Most of the results in this subsection are from [9]. First order formulas using at most 2 variables built using binary operators $\sim, <$ and $+1$ are considered. $<$ and $+1$ correspond to the usual ordering and successor function on the positions of a data word. Two positions x and y satisfy $x \sim y$ iff both positions have the same datum from the infinite data domain.

First, the following result from [22]. A formula in $LTL_1^{\downarrow}(\{X, X^{-1}, X^2F, X^{-2}F^{-1}\})$ is said to be *simple* iff each occurrence of a temporal operator is immediately preceded by $\downarrow_1$ (and there are no other occurrences of $\downarrow_1$). In the following, equivalent formulas means one is satisfiable iff the other one is.

Theorem 5.3 ([22]). *1. For each sentence of simple $LTL_1^{\downarrow}(\{X, X^{-1}, X^2F, X^{-2}F^{-1}\})$, an equivalent formula of $FO^2(\sim, <, +1)$ is computable in logarithmic space.*

2. For each formula of $FO^2(\sim, <, +1)$ with one free variable, an equivalent sentence of simple $LTL_1^{\downarrow}(\{X, X^{-1}, X^2F, X^{-2}F^{-1}\})$ is computable in polynomial space.

It is proven in [9] that satisfiability of $FO^2(\sim, <, +1)$ is decidable by a reduction to Petri net reachability. This is done in two stages. First, it is shown that an arbitrary formula can be effectively transformed into an equivalent normal form. We need the following notions to define this normal form.

1. The unary predicate $R_\#$ contains exactly those positions whose datum is different from the previous position (that is, $R_\#$ marks the positions where datum changes). It can be defined as follows:

$$\forall x((R_\#(x) \leftrightarrow \forall y(x = y + 1 \rightarrow x \nsim y))$$

2. A formula θ is *data-blind* if it does not use $\sim$.

3. A *type* is a one-variable formula $\alpha(x)$ that is a conjunction of unary predicates or their negations.

4. A formula θ is *type-ordering* if it is in one of the following forms, where α, β indicate arbitrary types:

   (a) Each equivalence class of $\sim$ contains at most one occurrence of α:

   $$\theta = \forall x \forall y((\alpha(x) \wedge \alpha(y) \wedge x \sim y) \Rightarrow x = y)$$

   (b) In each class, all occurrences of α occur strictly before all occurrences of β:

   $$\theta = \forall x \forall y((\alpha(x) \wedge \beta(y) \wedge x \sim y) \Rightarrow x < y)$$

   (c) In each class with at least one occurrence of α, there must be a β too:

   $$\theta = \forall x \exists y(\alpha(x) \Rightarrow (\beta(y) \wedge x \sim y))$$

A formula is in *data normal form* if it is a disjunction of formulas of the form

$$\exists R_1 \ldots R_m R_\# \bigwedge \theta$$

where all predicates $R_i, R_\#$ are unary and each θ is either a data-blind or a type-ordering $FO^2(\sim, <, +1)$ formula. This data normal form is an extension of the classic Scott normal form (see e.g. [42]).

In the second stage, satisfiability of $FO^2(\sim, <, +1)$ formulas in data normal form is reduced to reachability in Petri nets. Intuitively, data-blind formulas can be handled by the usual finite state automata. The other forms of simple $FO^2(\sim, <, +1)$ formulas above induce regular conditions on the class strings of a data word (a class string is the projection of a data word to the set of all positions having a given datum). Each class string can hence be recognized by a finite state automaton but a-priori, there is no limit on the number of classes. So we have to shuffle unboundedly many finite state automata, which results in a Petri net with given initial and final markings.

Theorem 5.4 ([9]). *From each $FO^2(\sim, <, +1)$ formula we can compute an equivalent formula in data normal form with a doubly exponential number of disjuncts, each of exponential size. From each formula ϕ in data normal form we can compute a Petri net of exponential size such that the set of all strings that can be fired from the net's initial marking to reach its final marking is equal to the string projection of the set of data words satisfying ϕ. Emptiness of multicounter automata (without zero tests) can be reduced in polynomial time to the satisfiability problem of $FO^2(\sim, <, +1)$.*

Some decidable and undecidable extensions of $FO^2(\sim, <, +1)$ are also considered in [9].

Theorem 5.5 ([9]). *The satisfiability problem for $FO^2(\sim, <, +1, \ldots, +n)$ is decidable.*

The proof of the above theorem is an extension of the proof of Theorem 5.4. In view of this, [22] extends Theorem 5.3 to establish equivalence between $FO^2(\sim, <, +1, \ldots, +n)$ and $LTL_1^{\downarrow}(X, X^{-1}, \ldots, X^n, X^{-n}, X^{n+1}F, X^{-n-1}F^{-1})$.

Suppose that there is a linear order on the infinite data domain. Let $\prec$ be a linear order on the positions of a data word such that $x \prec y$ iff the data value at x is smaller than the one at y.

Theorem 5.6 ([9]). *The satisfiability problems for $FO^3(\sim, +1)$ and $FO^2(\sim, \prec, +1, <)$ are undecidable.*

The availability of the equivalence relation $\sim$ and linear order $<$ in FO^2 are very powerful, as evidenced by the following decidability results. Consider the logic $FO^2(\Sigma, +_\alpha, +_\beta)$. This logic is interpreted on structures similar to words but have two successor relations $+_\alpha$ and $+_\beta$ instead of the usual one successor relation in words. Formulas in this logic can also reason about letters from the finite alphabet Σ. Another way of thinking about this is to consider data words as usual, but that have a second successor relation inherited from a successor relation of the infinite data domain, assuming that no two positions have the same data value.

Theorem 5.7 ([69]). *Satisfiability of $FO^2(\Sigma, +_\alpha, +_\beta)$ is decidable.*

Small extensions are again undecidable. Let $<_\alpha, <_\beta$ be the linear orders corresponding to the successor relations $+_\alpha, +_\beta$ respectively. Let $+2_\alpha$ be the two step successor function corresponding to the successor relation $+_\alpha$ ($+2_\alpha(x, y)$ iff $\exists z : +_\alpha(x, z) \wedge +_\alpha(z, y)$). The relations $+3_\alpha$ and $+2_\beta$ are defined similarly.

Theorem 5.8 ([69]). *Satisfiability of the following logics are undecidable: $FO^2(\Sigma, +_\alpha, <_\alpha, +_\beta, <_\beta)$, $FO^3(\Sigma, +_\alpha, +_\beta)$ and $FO^2(\Sigma, +_\alpha, +2_\alpha, +3_\alpha, +_\beta, +2_\beta)$.*

Decidability can be regained by giving up the successor relation $+1$ on the positions of a word and the equivalence relation $\sim$ of the infinite data domain. Let $\precsim$ be a binary relation on the positions of a data word such that $x \precsim y$ iff the data value at x is smaller than or equal to the one at y. Let $\leq$ be the reflexive closure of the linear order on the positions of a data word.

Theorem 5.9 ([89]). *The satisfiability of $FO^2(\Sigma, \leq, \precsim)$ is in* EXPSPACE.

5.3 Petri Net Languages

All notation and results in this section are from [79]. We consider Petri nets with a labelling $L : T \to \Sigma$ from the set of transitions to a finite alphabet Σ. We assume that Petri nets come with an initial marking M_0 and a finite set of final markings F. The different languages associated with a Petri net $\mathcal{N}$ are as below.

Definition 5.10. *The language of $\mathcal{N}$ is its labelled firing sequences:*

$$\mathcal{L}(\mathcal{N}) = \{w \in \Sigma^+ \mid \exists M_f \in F \quad \exists \sigma \in T^+ \quad M_0 \stackrel{\sigma}{\Longrightarrow} M_f \quad L(\sigma) = w\} \ .$$

The deadlock language of $\mathcal{N}$ is its maximal labelled firing sequences:

$$\begin{aligned}\mathcal{T}(\mathcal{N}) =&\{w \in \Sigma^+ \mid \exists M \in \mathbb{N}^{|T|} \quad \exists \sigma \in T^+ \quad M_0 \stackrel{\sigma}{\Longrightarrow} M \quad L(\sigma) = w \\ &\text{and no transition is firable at } M\} \ .\end{aligned}$$

The weak language of $\mathcal{N}$ is defined by a "covering" property of firing sequences.

$$\begin{aligned}\mathcal{G}(\mathcal{N}) =&\{w \in \Sigma^* \mid \exists M \in \mathbb{N}^{|T|} \quad \exists M_f \in F \quad \exists \sigma \in T^* \quad M_0 \stackrel{\sigma}{\Longrightarrow} M \quad M \geq M_f \\ &L(\sigma) = w\} \ .\end{aligned}$$

To avoid some technical complexities, only Petri nets that do not have arc weights (this corresponds to $W = 1$ in the definition of a Petri net given at the beginning of section 4.1) are considered in [79]. In addition, it is assumed that initial and all final markings have at most one token in each place. It is shown how to effectively transform an arbitrary Petri net into another one satisfying the above conditions, preserving languages and weak languages. This transformation may however result in an exponential blowup in the size of the Petri net.

To reason about Petri net languages, the usual syntax of MSO on words is extended by second order binary relation symbols $\leq_g$ and $=_g$, introduced in [79]. If X and Y are monadic second order variables representing subsets of domain elements from $[n] = \{0, \ldots, n-1\}$, then

$$\begin{aligned}X \leq_g Y &\triangleq \forall j \leq n, |X \cap [j]| \leq |Y \cap [j]| \\ X =_g Y &\triangleq X \leq_g Y \text{ and } |X| = |Y| \ .\end{aligned}$$

Intuitively, if Y is the set of positions of a string containing opening bracket and X is the set of positions of the string containing closing bracket, then the string is well bracketed iff $X =_g Y$.

The syntax of the logic with the above extension is as follows.

$$\begin{aligned}\chi ::= &x = y \mid x \leq y \mid Xx \mid X \leq_g Y \mid X =_g Y \mid \chi \wedge \xi \mid \chi \vee \xi \mid \neg\chi \mid \\ &\exists x\chi \mid \forall x\chi \mid \exists X\chi \mid \forall X\chi\end{aligned}$$

Various syntactic fragments of the above logic are considered. A quantifier free formula is any formula obtained from the following syntax.

$$\chi ::= x = y \mid x \leq y \mid Xx \mid X \leq_g Y \mid X =_g Y \mid \chi \wedge \xi \mid \chi \vee \xi \mid \neg\chi$$

A first order formula is any formula obtained from the following syntax.

$$\begin{aligned}\chi ::= &x = y \mid x \leq y \mid Xx \mid X \leq_g Y \mid X =_g Y \mid \chi \wedge \xi \mid \chi \vee \xi \mid \neg\chi \mid \\ &\exists x\chi \mid \forall x\chi\end{aligned}$$

A $\mathscr{L}_1$-formula is any formula obtained from the following syntax.

$$\chi ::= x = y \mid x \leq y \mid Xx \mid \chi \wedge \xi \mid \chi \vee \xi \mid \neg\chi \mid \\ \exists x\chi \mid \forall x\chi \mid \exists X\chi \mid \forall X\chi$$

Let $\overline{X}$ be a set of MSO variables. Let $\chi(\overline{X})$ denote a formula whose free variables are among $\overline{X}$. Following is the main theorem in [79].

Theorem 5.11 ([79]). *Let $\mathcal{L}$ be a language over an alphabet Σ. The following are equivalent:*

1. *$\mathcal{L}$ is a Petri net language.*
2. *$\mathcal{L}$ is a Petri net deadlock language.*
3. *$\mathcal{L}$ is defined by a sentence of the from $\exists\overline{X}\chi(\overline{X})$, where $\chi(\overline{X})$ is a first order formula.*
4. *(Normal form I) $\mathcal{L}$ is defined by a sentence of the form $\exists\overline{X}\chi(\overline{X})$, where $\chi(\overline{X})$ is a positive Boolean combination of formulas of the form $X =_g Y$ and first order $\mathscr{L}_1$-formulas.*
5. *(Normal form II) $\mathcal{L}$ is defined by a sentence of the form $\exists\overline{X}\chi(\overline{X})$, where $\chi(\overline{X})$ is a Boolean combination of formulas of the form $X \leq_g Y$ and first order $\mathscr{L}_1$-formulas.*

The direction of proof from the fragments of MSO in the above theorem to Petri nets is very similar to the proof of the classical Büchi-Elgot-Trakhtenbrot theorem. Most of the translation can be done with finite state automata. An extra step is needed while translating the atomic formula $X =_g Y$, where we need to recognize well bracketed strings. Unboundedly many opening brackets may occur before a closing bracket. We need to keep track of number of opening brackets that are not yet closed and this is where the power of Petri nets are used. Each occurrence of $=_g$ in the MSO formula will necessitate a new type of opening-closing bracket pair, so a finite state automaton with one stack is not powerful enough to handle this.

In the direction from Petri net languages to fragments of MSO above, the firing sequences are treated as strings where positions that add tokens to a place are associated with an opening bracket and positions removing tokens from that place are associated with closing bracket. Since transitions can add/remove at most one token at a time from a place and initial and final markings can have at most one token in every place, the firing sequence should be well bracketed (except for a missing bracket at the end). These conditions can be encoded in the fragments of MSO given above. The second order relation $=_g$ is only used to encode well bracketedness. It may be noted that most of the MSO formulas used in this direction of the proof can be written as data blind $FO^2(<, +1)$ formulas defined in section 5.2.2, except the formula that uses $=_g$. To express well bracketedness, type-ordering formulas defined in section 5.2.2 are sufficient.

If we relax the well bracketedness condition to prefix of well bracketedness strings, $\leq_g$ is enough.

Theorem 5.12 ([79]). *Let $\mathcal{L}$ be a language over an alphabet Σ. The following are equivalent:*

1. *$\mathcal{L}$ is a weak Petri net language.*
2. *$\mathcal{L}$ is defined by a sentence of the form $\exists\overline{X}\chi(\overline{X})$, where $\chi(\overline{X})$ is a positive Boolean combination of formulas of the form $X \leq_g Y$ and first order $\mathscr{L}_1$-formulas.*

The proof of Theorem 5.12 is very similar to that of Theorem 5.11. Since we only have to test that a string is a prefix of a well bracketed string, $\leq_g$ suffices and $=_g$ is not needed.

5.4 Exponential Space Upper Bound for Petri net Coverability and Boundedness

In this section, we give a sketch of the EXPSPACE upper bound given by Rackoff [83] for the Petri net coverability and boundedness problems.

We call a function $M : P \to \mathbb{Z}$ a **vector**. A transition t may be taken as a **step** at the vector M yielding a new vector M' given by the equation $M'(p) = M(p) - Pre(p,t) + Post(p,t)$ for all $p \in P$. This is denoted as $M \xrightarrow{t} M'$. Taking a transition t as a step from a vector M is a weaker notion than the firing of t from a marking M'', since the latter doest not allow negative numbers. A finite transition sequence $\sigma = t_1 t_2 \cdots t_r$ is a **walk** from an initial vector M_0 to a vector M_r if there exist intermediate vectors $M_1, M_2, \ldots, M_r$ such that for all i with $1 \leq i \leq r$, we have a step from M_{i-1} to M_i using the transition t_i. We write $M_0 \xrightarrow{\sigma} M_r$.

Let $Q \subseteq P$ be a subset of places. We will need the in-between notion (due to Rackoff [83]) of σ being a Q-**run** in which, for every intermediate vector $M_i, 0 \leq i < |\sigma|$, $M_i(p) \geq Pre(p, t_{i+1})$ for every place p in Q. Thus a firing sequence is a P-run. A $\emptyset$-run is a walk. For two vectors M_1 and M_2, we say $M_1 \geq_Q M_2$ if for every $p \in Q, M_1(p) \geq M_2(p)$. A walk σ from M_1 is said to Q-**cover** a marking M_{cov} if it is a Q-run and the final vector M_2 obtained by walking σ from M_1 satisfies $M_2 \geq_Q M_{cov}$. We say σ **covers** a marking if σ P-covers it.

We will fix for this section M_{cov} as the marking to be covered. For the purpose of complexity analysis, we will denote the maximum of the range of M_{cov} by R.

Definition 5.13. *A Q-**covering** run is a Q-run that Q-covers M_{cov}. Let $Q_0 \subseteq Q$. A Q-run from M_0 to M_r is said to be c-**bounded for** Q_0, $c \in \mathbb{N}$, if for all intermediate vectors $M_i, 0 \leq i < r$, $M_i(p)$ is in $\{0, \ldots, c\}$ for every place p in Q_0.*

Definition 5.14. *[83] Let $Q \subseteq P$. Define $lencov(Q, M, M_{cov})$ to be the length of the shortest Q-covering run from the vector M. If there is no such sequence, define $lencov(Q, M, M_{cov})$ to be 0. For $0 \leq i \leq b$, $\ell(i, M_{cov})$ is defined to be $max\{lencov(Q, M, M_{cov}) \mid M$ a vector, $Q \subseteq P$ and $|Q| = i\}$. In this section we abbreviate $\ell(i, M_{cov})$ to $\ell(i)$.*

The following recurrence relation for $\ell(i)$ is the main technical lemma used in [83] to obtain EXPSPACE upper bound for coverability. A closer analysis of this recurrence relation is performed in the next two chapters and here, only a sketch of the proof is given.

Lemma 5.15 ([83]). *$\ell(0) \leq 1$ and $\ell(i+1) \leq (W\ell(i) + R)^{i+1} + \ell(i)$.*

Proof. Suppose $Q \subseteq P$, $|Q| = i+1$ and there is a Q-covering run. If this run is $(W\ell(i)+R)$-bounded for Q, then we can get a Q-covering run of length at most $(W\ell(i) + R)^{i+1}$ by removing portions of the run between two vectors that are equal with respect to all places in Q. Otherwise, there is an intermediate vector M and a place $q \in Q$ such that $M(q) \geq (W\ell(i) + R)$. The portion of the Q-covering run occurring after M can be replaced by a shorter $(Q \setminus \{q\})$-covering run of length at most $\ell(i)$. □

With the recurrence relation obtained above, we can calculate an upper bound on the length of a shortest P-covering run that is double exponential in the size of the Petri net. Exponential space is enough for a Turing machine to guess and verify such a sequence.

Definition 5.16. *Let M be a vector and $Q \subseteq P$. A sequence of transitions σ is said to be a M, Q-**enabled self-covering** run if σ is a Q-run such that for some two positions $i < j$ of σ, the intermediate vectors M_i and M_j satisfy $M_j \geq_P M_i$ and for some $p \in P$, $M_j(p) > M_i(p)$.*

It is known [55] that self-covering sequences are necessary and sufficient for unboundedness. Suppose σ_c is a Q-covering run with last vector M and σ_{sc} is a Q-enabled self-covering

run with vectors M_i and M_j as defined above. While it is enough to ensure that $M \geq_Q M_{cov}$ for the Q-covering run, we have to ensure that $M_j >_P M_i$ for the Q-enabled self-covering run. Now, we will prove a lemma which states that if there is a self-covering sequence in which the tokens in some subset Q of places always remains less than some $c \in \mathbb{N}$, then there is one such sequence that is not too long.

Lemma 5.17 ([83]). *Let $Q \subseteq P$ be a subset of places, M a vector and $c \in \mathbb{N}$ a number. Suppose there is a M, Q-enabled self-covering run that is c-bounded for Q. Then, there is a M, Q-enabled self-covering run of length at most $(Wc)^{poly(m)}$, where $poly()$ is a polynomial whose degree does not depend on m, W or c.*

Proof. The idea is to remove portions of the self-covering run between intermediate vectors that are equal with respect to all places in Q (such sequences are called Q-loops). However, since we have to maintain the requirement that $M_j \geq_P M_i$ as in Definition 5.16, we can not remove all Q-loops as we did from Q-covering runs in Lemma 5.15. The requirement of $M_j \geq_P M_i$ can be stated in the form of a linear Diophantine equation whose solution will tell us how many Q-loops need to be retained to satisfy the requirement. Using the fact that feasible linear Diophantine equations have small solutions [10], many loops can be removed, leaving behind a short self-covering sequence. □

Definition 5.18. *Let $Q \subseteq P$ and M a vector. Define $\lambda(Q, M)$ to be the length of the shortest M, Q-enabled self-covering sequence and 0 if there is no such sequence. Let $\lambda(i) = \max\{\lambda(Q, M) \mid M$ a vector, $|Q| = i\}$.*

Lemma 5.19. *$\lambda(0) \leq 2^{poly(m)}$ and $\lambda(i+1) \leq (W^2\lambda(i))^{poly(m)}$, where $poly()$ is a polynomial whose degree is a constant independent of m, W and i.*

Proof. Suppose $Q \subseteq P$, $|Q| = i + 1$ and there is a M, Q-enabled self-covering run for some vector M. If this run is $(W\lambda(i))$-bounded for Q, then the result follows from Lemma 5.17. Otherwise, there is an intermediate vector M' and a place $q \in Q$ such that $M'(q) \geq W\lambda(i)$. The portion of the M, Q-enabled self-covering run occurring after M can be replaced by a shorter sequence of length at most $\lambda(i)$. □

With the recurrence relation obtained above, we can calculate an upper bound on the length of a shortest self-covering sequence that is double exponential in the size of the Petri net. Again, exponential space is enough for a Turing machine to guess and verify such a sequence.

5.5 A Unified Approach for Deciding the Existence of Certain Petri Net Paths

The notation and results of this section are from [99, 4]. As usual, we denote by $\mathcal{N} = (P, T, Pre, Post)$ a Petri net with places P and transitions T. If σ is a firing sequence, then its *Parikh image* $\#(\sigma) : T \to \mathbb{N}$ is a mapping such that each transition t appears $\#(\sigma)(t)$ times in σ. Yen [99] defined a class of path formulas for Petri nets consisting of the following elements.

1. *Variables:* There are two types of variables, namely marking variables $M_1, M_2, \ldots$ and variables for transition sequences $\sigma_1, \sigma_2, \ldots$.

2. *Terms:* Terms are recursively defined as follows.

   - For every mapping $\mathbf{c} \in \mathbb{N}^P$, $\mathbf{c}$ is a term.

- For all $j > i$, $M_j - M_i$ is a term, where M_j and M_i are marking variables.
- If $\mathcal{T}_1$ and $\mathcal{T}_2$ are terms, then so are $\mathcal{T}_1 + \mathcal{T}_2$ and $\mathcal{T}_1 - \mathcal{T}_2$.

3. *Atomic predicates:* There are two types of atomic predicates.

   (a) Transition predicates:

   - $\mathbf{z} \odot \#(\sigma_i) \geq c$ and $\mathbf{z} \odot \#(\sigma_i) > c$ are predicates, where $i > 1$, $c \in \mathbb{N}$ is a constant, $\mathbf{z}$ is a mapping from T to $\mathbb{Z}$ and $\odot$ is the vector dot product.
   - $\#(\sigma_1)(t) \geq c$ and $\#(\sigma_1)(t) \leq c$ are predicates, where $c \in \mathbb{N}$ is a constant and $t \in T$ is a transition of $\mathcal{N}$.

   (b) Marking predicates:

   - $M(p) \geq z$ and $M(p) > z$ are predicates, where M is a marking variable, $p \in P$ is a place of $\mathcal{N}$ and $z \in \mathbb{Z}$ is an integer.
   - $\mathcal{T}_1(p_1) = \mathcal{T}_2(p_2)$, $\mathcal{T}_1(p_1) < \mathcal{T}_2(p_2)$ and $\mathcal{T}_1(p_1) > \mathcal{T}_2(p_2)$ are predicates, where $\mathcal{T}_1$ and $\mathcal{T}_2$ are terms and $p_1, p_2 \in P$ are two places of $\mathcal{N}$.

A *predicate* is any positive Boolean combination of transition predicates or marking predicates. A *path formula* is a formula of the form

$$\exists M_1, \ldots, M_r \exists \sigma_1, \ldots, \sigma_r \left((M_0 \stackrel{\sigma_1}{\Longrightarrow} M_1 \stackrel{\sigma_2}{\Longrightarrow} \cdots \stackrel{\sigma_r}{\Longrightarrow} M_r) \wedge \phi(M_1, \ldots, M_r, \sigma_1, \ldots, \sigma_r) \right)$$

where ϕ is a predicate and M_0 is the initial marking of $\mathcal{N}$.

Theorem 5.20 ([4]). *Given a Petri net $\mathcal{N}$ with initial marking M_0 and target marking M_1, another Petri net $\mathcal{N}'$ and a path formula f can be constructed in polynomial time such that M_1 is reachable from M_0 in $\mathcal{N}$ iff $\mathcal{N}'$ satisfies f. Given a Petri net $\mathcal{N}$ and a path formula f, another Petri net $\mathcal{N}'$ can be constructed in polynomial time such that $\mathcal{N}$ satisfies f iff $\mathcal{N}'$ can reach the marking that has 0 tokens in all places.*

An EXPSPACE upper bound is given in [4] for model checking a fragment of path formulas.

Definition 5.21. *A path formula*

$$\exists M_1, \ldots, M_r \exists \sigma_1, \ldots, \sigma_r \left((M_0 \stackrel{\sigma_1}{\Longrightarrow} M_1 \stackrel{\sigma_2}{\Longrightarrow} \cdots \stackrel{\sigma_r}{\Longrightarrow} M_r) \wedge \phi(M_1, \ldots, M_r, \sigma_1, \ldots, \sigma_r) \right)$$

is called **increasing** *if $\phi(M_1, \ldots, M_r, \sigma_1, \ldots, \sigma_r)$ does not contain transition predicates and implies $M_r > M_1$.*

Theorem 5.22 ([4]). *Given a Petri net $\mathcal{N}$ and an increasing path formula f, checking whether $\mathcal{N}$ satisfies f can be done in exponential space.*

The proof of the above theorem involves extending Rackoff's induction strategy for boundedness [83] to increasing path formulas.

5.6 Simulating Exponential Space Turing Machines on Vector Addition Systems with States

A Vector Addition Systems with States (VASS) is a finite directed graph (Q, E), an integer $m \in \mathbb{Z}$ called its *dimension* and a mapping $R : E \to \mathbb{Z}^m$. Here, Q is the set of states and E is the set of transitions. A *configuration* of a VASS is a pair (q, M) where $q \in Q$ is a state and $M \in \mathbb{Z}^m$ is a vector of integers. A *walk* is a pair (σ, M_0) where $M_0 \in \mathbb{Z}^m$ and $e_1 \cdots e_{|\sigma|}$

is a sequence of edges forming a path in the graph (Q, E). The sequence of configurations in the walk is $(q_0, M_0), (q_1, M_1), \ldots, (q_{|\sigma|}, M_{|\sigma|})$, where for each $1 \leq i \leq |\sigma|$, $M_i = M_{i-1} + R(e_i)$ and q_{i-1} and q_i are the start and end vertices respectively of the edge e_i. Each M_i is called an intermediate vector in the walk and M_0, q_0 are called the initial vector and initial state respectively. A positive walk is one the initial vector and all intermediate vectors are made up of natural numbers. A positive walk is represented as $(q_0, M_0) \stackrel{\sigma}{\Longrightarrow} (q_{|\sigma|}, M_{|\sigma|})$.

Definition 5.23 (Reachability, Coverability and Boundedness)**.** *Given a VASS with initial state q_0, initial vector M_0 and a target configuration (q, M_{cov}), the* Coverability *problem is to determine if there is a positive walk (σ, M_0) such that $(q_0, M_0) \stackrel{\sigma}{\Longrightarrow} (q, M_{|\sigma|})$ and for every $1 \leq i \leq m$, $M_{|\sigma|}(i) \geq M_{cov}(i)$ (this is denoted as $M' \geq M_{cov}$). If we replace $M' \geq M_{cov}$ by $M' = M_{cov}$, we get the* reachability *problem. The boundedness problem is to determine if there is a number $c \in \mathbb{N}$ such that for every positive walk σ starting at (q_0, M_0) with $(q_0, M_0) \stackrel{\sigma}{\Longrightarrow} (q_{|\sigma|}, M_{|\sigma|})$ $M_{|\sigma|}(i) \leq c$ for every place $1 \leq i \leq m$.*

VASS are known to be equivalent to Petri nets. In [87], a detailed lower bound is given for the boundedness problem of VASS. There, the contribution of various natural parameters of VASS to the lower bound is studied in detail. VASS provide a natural way to separate control structure and counters. The graph underlying a VASS can be thought of as a control structure, and parameters of the graph can be used to study parameterized complexity. A close examination of the control structure of the VASS used in [87] (which is in turn based on Lipton's lower bound proof [66]) gives us the following result.

Proposition 5.24. *A VASS whose underlying graph has bounded pathwidth can simulate exponential space bounded Turing machines.*

Proof. It is known that counter machines (without incremental errors) can simulate Turing machines [51, 75]. In particular, if a Turing machine operates in space exponential in the size of its input, then it can be simulated by a counter machine with a fixed number of counters such that the numbers in the counters never exceed double exponential of the size of the input. The only reason VASS can not simulate counter machines in general is that VASS can not test numbers for zero. Using the fact that counters never exceed double exponential values, Lipton [66] shows that zero testing can be simulated by VASS when counters are bounded ([87, 30] also contain detailed descriptions of Lipton's construction). The following proof is based on a close examination of Lipton's construction. Suppose a co-ordinate i' that can hold values up to 2^{2^j} is to be tested for zero. Another co-ordinate i is maintained with the invariant that the sum of values in i' and i is 2^{2^j} throught the VASS's operation. To test i' for zero, it is enough to check that i has value 2^{2^j}. Figure 5.2 shows a schematic of the control structure of the VASS that does this.

The idea is that two co-ordinates i_1 and i_2 that can hold values up to $2^{2^{j-1}}$ are used as indices to run nested loops that decrement i. The control begins at q_0 with values $2^{2^{j-1}}$ in i_1, i_2 and reaches q_N if i' is not zero (this can be easily tested by decrementing and immediately incrementing i' once). In the other case, control reaches q_1 from where the outer loop begins. From q_1 to q_2, i_1 is decremented and from q_2, i_2 is decremented. Then i is decremented once. At q_i, if i_2 is not zero, control goes back to q_2 for the next iteration of the inner loop. Otherwise, i_2 is tested for zero by transferring the control to another module of the VASS that performs zero tests for co-ordinates that can hold values up to $2^{2^{j-1}}$. The loop between q_3 and q_4 implement a "argument passing system" to ensure that i_2 is the one being tested for zero. When this module returns control at q_0, if i_1 is non-zero, control is transferred to q_1 for the next iteration of the outer loop. Otherwise, i_1 is tested for zero by passing it as a parameter to the same inner module through the loop formed by q_5 and q_6. If i_1 also happens to be zero, then the control is transferred to q_Y to indicate that i' was zero. Indeed, since

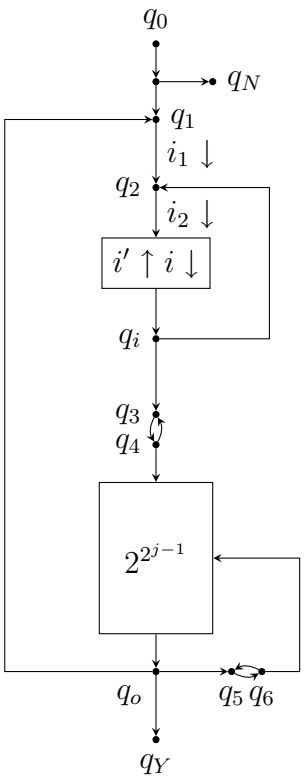


Figure 5.2: Control structure of VASS for zero testing

both i_1 and i_2 are zero, decrementing i (the block with $i \downarrow$) is executed $2^{2^{j-1}} \times 2^{2^{j-1}} = 2^{2^j}$ times, which means that i had value 2^{2^j}, which means that i' had 0.

The above sketch skips many intricate details and the interested reader is referred to [87, 30]. The main point to be noted is that the inner VASS module referred to above is constructed inductively in the same way, with loops controlled by counters with value $2^{2^{j-2}}$. Let Q_j be the set of control states of the VASS above, without taking into account the states of inner modules. If a co-ordinate with maximum value 2^{2^n} has to be tested for zero, the set of states of the VASS constructed inductively as described above would have set of states $Q_n \cup Q_{n-1} \cup \cdots \cup Q_0$. Since there is no edge across states in Q_j and Q_{j-2} for any j, following is a path decomposition of the entire control graph of the VASS for zero testing: $(Q_0 \cup Q_1) - (Q_1 \cup Q_2) - \cdots - (Q_{n-1} \cup Q_n)$. For larger n, this decomposition will be longer but the size of each bag remains the same. Addition of other parts of the counter machine's finite state control will increase the size of the bags by only a constant. □

Hence, to get better algorithms, we will have to consider parameters that also take into account the way counters interact with each other. This is the motivation for considering the parameter benefit depth in the next chapter.

5.7 Presburger Arithmetic with Monotone Transitive Closure

Since Petri net reachability sets need not be semilinear, Presburger arithmetic (first order theory of natural numbers with addition) cannot define reachability sets. In [85], Presburger arithmetic is generalized by introducing an operation called *monotone transitive closure*. Let $\Phi = \{x, y, \ldots\}$ be a set of variable symbols. The generalization of Presburger arithmetic

mentioned above is made up of terms τ and formulas ϕ, given by the following syntax.

$$\begin{aligned}
&\tau ::= 0 \mid x \in \Phi \mid S(\tau) \mid \tau + \tau \\
&\psi ::= \tau = \tau \mid \neg\psi \mid \psi \vee \psi \mid \exists x\psi \\
&\phi ::= \tau = \tau \mid \tau < \tau \mid \neg\phi \mid \phi \vee \phi \mid \exists x\phi \mid \mathrm{mTC}(\psi(x_1, \ldots, x_k, y_1, \ldots, y_k))
\end{aligned} \tag{5.1}$$

In the formula $\mathrm{mTC}(\psi(x_k, \ldots, x_k, y_k, \ldots, y_k))$, $x_1, \ldots, x_k, y_1, \ldots, y_k$ are the free variables of ψ.

Let $s : \Phi \to \mathbb{N}$ be an assignment to the variable symbols. This assigns a natural number $[\![\tau]\!]_s$ to every term τ as defined below.

$$\begin{aligned}
[\![0]\!]_s &= 0, \text{ the smallest natural number.} \\
[\![x]\!]_s &= s(x) \\
[\![S(\tau)]\!]_s &= [\![\tau]\!]_s + 1 \\
[\![\tau_1 + \tau_2]\!]_s &= [\![\tau_1]\!]_s + [\![\tau_2]\!]_s
\end{aligned}$$

If s is an assignment and $n \in \mathbb{N}$ is a natural number, then $s[x \to n]$ is the assignment that is same as s except that x is assigned n. The definition of an assignment s satisfying a formula ϕ (denoted as $\mathbb{N}, s \models \phi$) is as below. We write $\mathbb{N} \models \psi(n_1, \ldots, n_k, n'_1, \ldots, n'_k)$ to denote the fact that $\mathbb{N}, s[x_1 \to n_1, \ldots, x_k \to n_k, y_1 \to n'_1, \ldots, y_k \to n'_k] \models \psi(x_1, \ldots, x_k, y_1, \ldots, y_k)$.

$\mathbb{N}, s \models \tau_1 = \tau_2$ iff $[\![\tau_1]\!]_s = [\![\tau_2]\!]_s$
$\mathbb{N}, s \models \tau_1 < \tau_2$ iff $[\![\tau_1]\!]_s < [\![\tau_2]\!]_s$
$\mathbb{N}, s \models \neg\phi$ iff $\mathbb{N}, s \not\models \phi$
$\mathbb{N}, s \models \phi_1 \vee \phi_2$ iff $\mathbb{N}, s \models \phi_1$ or $\mathbb{N}, s \models \phi_2$
$\mathbb{N}, s \models \exists x\phi$ iff there is a $n \in \mathbb{N}$ such that $\mathbb{N}, s[x \to n] \models \phi$
$\mathbb{N}, s \models \mathrm{mTC}(\psi(x_1, \ldots, x_k, y_1, \ldots, y_k))$ iff $(s(x_1), \ldots, s(x_k), s(y_1), \ldots, s(y_k)) \in Tc$,
where $Tc \in \mathbb{N}^{2k}$ is the smallest set satisfying the following properties:

1. $(n_1, \ldots, n_k, n_1, \ldots, n_k) \in Tc$ for $n_1, \ldots, n_k \in \mathbb{N}$,
2. $(n_1, \ldots, n_k, n'_1, \ldots, n'_k) \in Tc$ whenever $\mathbb{N} \models \psi(n_1, \ldots, n_k, n'_1, \ldots, n'_k)$,
3. $(n_1, \ldots, n_k, n''_1, \ldots, n''_k) \in Tc$ whenever $(n_1, \ldots, n_k, n'_1, \ldots, n'_k) \in Tc$, $(n'_1, \ldots, n'_k, n''_1, \ldots, n''_k) \in Tc$ and $n'_1, \ldots, n'_k \in \mathbb{N}$, and
4. $(n_1 + n''_1, \ldots, n_k + n''_k, n'_1 + n''_1, \ldots, n'_k + n''_k) \in Tc$ whenever $(n_1, \ldots, n_k, n'_1, \ldots, n'_k) \in Tc$ and $n''_1, \ldots, n''_k \in \mathbb{N}$.

Condition (1) above can be thought of as indicating the fact that in a Petri net, a marking can be reached from itself by firing the empty firing sequence. Condition 3 indicates that if $M \stackrel{\sigma}{\Longrightarrow} M'$ and $M' \stackrel{\sigma'}{\Longrightarrow} M''$, then $M \stackrel{\sigma\sigma'}{\Longrightarrow} M''$. Condition 4 indicates that if $M \stackrel{\sigma}{\Longrightarrow} M'$, then $M + M'' \stackrel{\sigma}{\Longrightarrow} M' + M''$ for every marking M''.

Theorem 5.25 ([85]). *Given a formula ϕ as given in* (5.1)*, checking for the existence of a satisfying assignment to its free variables is decidable.*

The proof of the above theorem is through usage of some newly defined operators operating on sets of multisets. Expressions made up of such operators can be built up corresponding to the given formula (or as shown in [85], corresponding to a given Petri net with initial and final markings). The expression can then be simplified resulting in a lengthier expression, following the style of proof of decidability of Petri net reachability.

5.8 Branching Vector Addition Systems

The notation and results of this section are from [20]. A *Branching Vector Addition System* (BVAS) is a tuple $\mathcal{B} = (m, A_0, R_1, R_2)$ where $m \in \mathbb{N}$ is the dimension, $A_0 \subseteq \mathbb{N}^m$ is a non-empty finite set of axioms and R_1, R_2 are non-empty finite sets of unary and binary rules, respectively. A *derivation* of $\mathcal{B}$ is a labelling $\mathcal{D} : \mathcal{T} \to \mathbb{Z}^m$ such that:

- $\mathcal{T}$ is a finite binary tree,
- if η has one child in $\mathcal{T}$, then $\mathcal{D}(\eta) \in R_1$ and
- if η has two children in $\mathcal{T}$, then $\mathcal{D}(\eta) \in R_2$.

Vectors that are derived at every node are recursively obtained as follows:

- if η is a leaf in $\mathcal{T}$, then $\widehat{\mathcal{D}}(\eta) = \mathcal{D}(\eta)$.
- if η has one child η' in $\mathcal{T}$, then $\widehat{\mathcal{D}}(\eta) = \mathcal{D}(\eta) + \widehat{\mathcal{D}}(\eta')$.
- if η has two children η' and η'' in $\mathcal{T}$, then $\widehat{\mathcal{D}}(\eta) = \mathcal{D}(\eta) + \widehat{\mathcal{D}}(\eta') + \widehat{\mathcal{D}}(\eta'')$.

Now, we say that $\mathcal{D}$ is:

- *initialized* iff, for each leaf η of $\mathcal{T}$, $\mathcal{D}(\eta) \in A_0$.
- *admissible* iff, for each node η of $\mathcal{T}$, $\widehat{\mathcal{D}}(\eta) \in \mathbb{N}^m$.
- *derives* $\widehat{\mathcal{D}}(\epsilon)$, which is the vector derived at the root.

A BVAS *produces* a vector $\mathbf{v}$ if $\mathbf{v}$ is derived from an initialized admissible derivation. Given a BVAS $\mathcal{B}$ and a target vector $\mathbf{t}$ of the same dimension, the *reachability* problem is to decide if $\mathcal{B}$ can produce the vector $\mathbf{t}$. Given a BVAS $\mathcal{B}$ and a target vector $\mathbf{t}$ of the same dimension, the *coverability* problem is to decide if $\mathcal{B}$ can produce a vector $\mathbf{v}$ such that $\mathbf{v} \geq \mathbf{t}$. The boundedness problem is to decide if the set of all vectors produced by $\mathcal{B}$ is finite.

Theorem 5.26 ([20]). *The covering and boundedness problems for BVAS are complete for doubly-exponential time.*

The proof of the above theorem involves extending Lipton's counting strategy in [66] and Rackoff's induction strategy in [83] to branching systems. For boundedness, this extension requires proving new upper bounds for small solutions of linear Diophantine equations.

Theorem 5.27 ([63]). *The reachability problem for BVAS is hard for double exponential space.*

The proof of the above theorem involves a careful combination of Lipton's counting strategy [66] and branching to compute triply-exponentially large numbers.

5.9 Petri Nets and the Theory of Tensor $\star$

Linear logic has been described as a "resource conscious logic". In its proofs occurrences of propositions can not be used more than once, neither can they disappear unless they are explicitly used up by the rules of inference. Similar ideas being present in Petri nets, relations between them have been explored. In this section, we use the fragment of linear logic built up from the connectives tensor $\star$ and implication $\to$. Meanings of these connectives are implied by the proof rules given below. The following relation between the deducibility problem for

sequents in finitely axiomatized $\star$-theories and the Petri net reachability problem is from [94, Chapter 1].

We begin with the theory consisting of axioms

$$\text{id}{:}X \to X$$

(for any string X constructed from the places $p_1, p_2, \ldots$ and $\star$) and two general rules

$$\frac{t : p_1 \to p_2 \quad t' : p_2 \to p_3}{t \cdot t' : p_1 \to p_3} \qquad \frac{t : p_1 \to p_2 \quad t' : p_3 \to p_4}{t \parallel t' : p_1 \star p_3 \to p_2 \star p_4}$$

(the first rule corresponds to sequential composition of transitions; the second one corresponds to parallel composition). For each Petri net transition, an axiom is added. If a transition t for example removes 2 tokens from p_1, removes 3 tokens from p_2, adds one token to p_3 and adds 2 tokens to p_4, we add the axiom

$$t : p_1^2 \star p_2^3 \to p_3 \star p_4^2 \ .$$

Here, p_1^2 denotes $p_1 \star p_1$. Since $\star$ is associative and commutative, this shorthand notation can be used. A final marking M_f is reachable from an initial marking M_0 in a Petri net $\mathcal{N}$ iff $M_0 \to M_f$ is deducible in the theory formulated above.

5.10 Relevance Logic

Formulas of relevance logic are built up from atomic propositions using the Boolean connectives $\wedge, \vee$ in addition to implication $\to$ and fusion $\circ$. Different choices of connectives and axioms lead to different fragments of the logic. In the following sub-sections, we look at two such fragments.

5.10.1 Implication Conjunction Fusion Fragment of Relevance Logic

The notation and results of this section are from [95]. The logic discussed is the system $R_{\to\wedge\circ}$, which contains the fusion connective $\circ$ in addition to $\to$ and $\wedge$. The axioms for the system are

1. $\top$
2. $\top \to (q \to q)$
3. $(q \to r) \to ((r \to s) \to (q \to s))$
4. $(q \to (r \to s)) \to (r \to (q \to s))$
5. $(q \to (r \to s)) \to ((q \to r) \to (q \to s))$
6. $(q \wedge r) \to q$
7. $(q \wedge r) \to r$
8. $(q \to r) \wedge (q \to s) \to (q \to (r \wedge s))$
9. $q \to (r \to (q \circ r))$
10. $(q \to (r \to s)) \to ((q \circ r) \to s)$

Following are the inference rules of $R_{\rightarrow\wedge\circ}$.

$$\frac{q \rightarrow r \quad q}{r}(\text{modus ponens}) \qquad \frac{q \quad r}{q \wedge r}(\text{adjunction})$$

The semantics of relevance logic is based on a structure with a ternary relation. A *model structure* is a triple (o, W, E), where W is a set, $o \in W$ and E is a ternary relation on W satisfying the following conditions for all $w, w_1, w_2, w_3, w_4, w_5 \in W$:

1. $E(o, w, w)$
2. $E(w, w, w)$
3. $(E(w_1, w_2, w_3) \wedge E(w_3, w_4, w_5)) \rightarrow \exists w_6(E(w_1, w_4, w_6) \wedge E(w_6, w_2, w_5))$
4. $(E(o, w_1, w_2) \wedge E(w_2, w_3, w_4)) \rightarrow E(w_1, w_3, w_4)$
5. $(E(w_1, w_2, w_3) \wedge E(o, w_3, w_4)) \rightarrow E(w_1, w_2, w_4)$.

If $M = (o, W, E)$ is a model structure, then a valuation V is a function that assigns to each propositional variable q a set $V(q) \subseteq W$, and which satisfies the condition: if $w \in V(q)$ and $E(o, w, w')$ then $w' \in V(q)$. A *model* is a model structure with a valuation.

If $\mathcal{M}$ is a model, the truth relative to a point is defined recursively as follows:

1. $\mathcal{M}, w \models q$ iff $w \in V(q)$
2. $\mathcal{M}, w \models \top$ iff $E(o, o, w)$
3. $\mathcal{M}, w \models q \rightarrow r$ iff $\forall w_1 w_2((E(w, w_1, w_2)$ and $\mathcal{M}, w_1 \models q) \Rightarrow \mathcal{M}, w_2 \models r)$
4. $\mathcal{M}, w \models q \circ r$ iff $\exists w_1 w_2(E(w_1, w_2, w)$ and $\mathcal{M}, w_1 \models q$ and $\mathcal{M}, w_2 \models r)$
5. $\mathcal{M}, w \models q \wedge r$ iff $\mathcal{M}, w \models q$ and $\mathcal{M}, w \models r$

A formula q is *valid* if $\mathcal{M}, o \models q$ in all models $\mathcal{M}$.

Exponential space lower bound is shown for the validity problem through *semi-Thue* systems. If Σ is a finite alphabet, a semi-Thue system S is a finite set of pairs of strings in Σ^*. Pairs of strings in the semi-Thue system are written as $\alpha \rightarrow \beta$ and referred to as *production rules* of the system.

If S is a semi-Thue system, we write $\gamma\alpha\delta \Longrightarrow \gamma\beta\delta(S)$ if γ and δ are words in the alphabet of S and the production rule $\alpha \rightarrow \beta$ belongs to S. The relation $\overset{\star}{\Longrightarrow}$ is the reflexive transitive closure of $\Longrightarrow$.

A semi-Thue system over an alphabet Σ is said to be *commutative* if it includes all productions of the form $ab \rightarrow ba$ for all $a, b \in \Sigma$; it is *contractive* if it includes all productions of the form $aa \rightarrow a$ for all $a \in \Sigma$. If S is a commutative contractive semi-Thue system, then a production in S is said to be *proper* if it is not of the form $aa \rightarrow a$ or $ab \rightarrow ba$.

If S is semi-Thue system over Σ, define a ternary relation E over Σ^* as $E(\alpha, \beta, \gamma)$ iff $\alpha\beta \overset{\star}{\Longrightarrow} \gamma(S)$. With every letter $a \in \Sigma$, associate a propositional variable $q(a)$. If $\alpha \in \Sigma^*$, then $q(\alpha)$ is the propositional expression corresponding to α, where concatenation is represented by the fusion operator. The triple (ϵ, Σ^*, E) will form a model structure, where ϵ is the empty string. The valuation V over this model structure defined as $V(q(a)) = \{\alpha \in \Sigma^* \mid a \overset{\star}{\Longrightarrow} \alpha(S)\}$ for all $a \in \Sigma$, gives the canonical model $\mathcal{M}(S)$ associated with S. Suppose $\alpha_1 \rightarrow \beta_1, \ldots, \alpha_n \rightarrow \beta_n$ are the proper productions of S. If γ, δ are words in Σ^*, then $\psi(S, \gamma, \delta)$ is the formula

$$[q(\beta_1) \rightarrow q(\alpha_1) \wedge \cdots \wedge q(\beta_n) \rightarrow q(\alpha_n) \wedge \top] \rightarrow (q(\delta) \rightarrow q(\gamma)) \ .$$

Theorem 5.28 ([95]). *Let S be a commutative, contractive semi-Thue system over Σ and γ, δ words in Σ^*. Then $\mathcal{M}(S), \gamma \models q(\delta)$ iff $\delta \stackrel{\star}{\Longrightarrow} \gamma(S)$. In addition, $R_{\rightarrow\wedge\circ} \vdash \psi(S, \gamma, \delta)$ iff $\gamma \stackrel{\star}{\Longrightarrow} \delta(S)$.*

It is also shown in [95] how to construct in logspace a commutative, contractive semi-Thue system from a given doubly exponential bounded 3-counter machine so that the termination problem for such 3-counter machines can be reduced to the word problem for commutative, contractive semi-Thue systems. This gives an exponential space lower bound for the deducibility problem of $R_{\rightarrow\wedge\circ}$. It is also shown in [95] how to apply McAloon's result [72] about upper bounds for Dickson's lemma to conclude that Kripke's decision procedure for deducibility in $R_{\rightarrow\wedge\circ}$ (described in [92, pp. 30–39]) is primitive recursive in the Ackermann function.

5.10.2 LR+ Fragment of Relevance Logic

Results from this section are from [96], where complexity of decision procedures for the system LR+ is considered. This system consists of the connectives fusion $\circ$, $\rightarrow$, $\wedge$ and $\vee$. The axiom system consists of all axioms from the previous subsection and some additional ones for $\vee$.

1. $\top$
2. $\top \rightarrow (q \rightarrow q)$
3. $(q \rightarrow r) \rightarrow ((r \rightarrow s) \rightarrow (q \rightarrow s))$
4. $(q \rightarrow (r \rightarrow s)) \rightarrow (r \rightarrow (q \rightarrow s))$
5. $(q \rightarrow (r \rightarrow s)) \rightarrow ((q \rightarrow r) \rightarrow (q \rightarrow s))$
6. $(q \wedge r) \rightarrow q$
7. $(q \wedge r) \rightarrow r$
8. $(q \rightarrow r) \wedge (q \rightarrow s) \rightarrow (q \rightarrow (r \wedge s))$
9. $q \rightarrow (r \rightarrow (q \circ r))$
10. $(q \rightarrow (r \rightarrow s)) \rightarrow ((q \circ r) \rightarrow s)$
11. $q \rightarrow q \vee r$
12. $r \rightarrow q \vee r$
13. $(q \rightarrow s) \wedge (r \rightarrow s) \rightarrow ((q \vee r) \rightarrow s)$

Following are the inference rules of $R_{\rightarrow\wedge\circ}$.

$$\frac{q \rightarrow r \quad q}{r}(\text{modus ponens}) \qquad \frac{q \quad r}{q \wedge r}(\text{adjunction})$$

A non-primitive recursive lower bound is shown for validity in systems of LR+ through a series of reductions. First, 3-counter machines whose counters are bounded by Ackermann function of the size of the machine are considered. The termination problem for such machines is known to be non-primitive recursive.

The above termination problem is reduced to termination of Expansive counter machines with zero tests (ECMs). An ECM is a counter machine where the automaton can test

counters for zero and counters that have non-zero value can arbitrarily increase their value. The zero testing ability is critically used to ensure that these expansion errors do not hinder simulation of Ackermann bounded 3-counter machines.

The termination problem for ECMs is reduced to the acceptance problems for Expansive And branching Counter Machines (EACM). An EACM is similar to an ECM but zero tests are not available. Instead, EACMs have branching transitions. On executing a branching transition, two copies of the counter machine are created, each copy in a new control state and values of the counter same as those before the transition. An initial configuration thus gives rise to a computation tree. An initial configuration is said to be accepted if there is a computation tree rooted at that configuration such that all leaves are in accepting state and all counters are zero in all leaves. It is easy to simulate ECM by EACM — whenever we need to perform a zero test in the ECM, we provide a branching transition in the EACM. One branch assumes that the counter under consideration is zero and continues the computation. The other branch tests that the counter is actually zero.

Finally, the acceptance problem for EACMs is reduced to deducibility in LR+. A sequent calculus equivalent to the axiom system given above is given in [96]. A sequent is associated with every transition of the EACM. In the resulting theory, the sequent corresponding to the initial configuration of the EACM is derivable iff the initial configuration is accepted by the EACM. Finally, it is shown that deducibility in the sequent calculus is equivalent to theoremhood in LR+, so that the decision problem for LR+ is not primitive recursive. Utilizing some translations from LR+ to implication conjunction fragment of some other logics [73, 74], this lower bound is extended to $R_{\rightarrow\wedge}$.

As is done in the previous subsection, an upper bound that is primitive recursive in the Ackermann function is shown by analysing a decision procedure for LR. LR results from R by dropping the distribution axiom.

5.11 Summary of results

The following table summarizes the results mentioned in this chapter. The lower bound mentioned in the "Lower bound" column of the table need not imply a formal complexity theoretic lower bound in all cases, since some of the reductions involve exponential blowups, such as those for satisfiability of EMSO with $=_g$.

Problem	Lower bound	Upper bound
$LTL_1^{\downarrow}(X,U)$ Satisfiability	Non-primitive recursive. Reduction from non-emptiness of Expansive counter machines (ECMs) with zero tests [22].	Reduction to non-emptiness of alternating register automata to non-emptiness of ECMs with zero tests [22].
$LTL_2^{\downarrow}(X,U)$ and $LTL_1^{\downarrow}(X,U,F^{-1})$ satisfiability.	Undecidable. Reduction from termination problem for Minsky machines [22].	–
Non-emptiness of 2-way alternating register automata.	Undecidable. Reduction from termination problem for Minsky machines [22].	–
$FO^2(\sim,<,+1)$ satisfiability.	Logspace reduction from Petri net reachability [9].	Double exponential time reduction to Petri net reachability [9].

Problem	Lower bound	Upper bound
$FO^3(\sim, +1)$ and $FO^2(\sim, \prec, +1, <)$ satisfiability.	Undecidable [9]. Reduction from Post's Correspondence Problem [9].	–
Satisfiability of existential MSO with $=_g$.	Reduction from Petri net reachability [79].	Reduction to Petri net reachability [79].
Satisfiability of existential MSO with $\leq_g$.	Reduction from Petri net coverability [79].	Reduction to Petri net coverability [79].
Petri net coverability and boundedness.	EXPSPACE-hard [66].	EXPSPACE [83].
Model checking Yen's path logic formulas on Petri nets	Polynomial time reduction from Petri net reachability [4].	Polynomial time reduction to Petri net reachability [4].
Model checking increasing path formulas of Yen's path logic on Petri nets	EXPSPACE [66].	EXPSPACE [4].
Emptiness and satisfiability for Presburger arithmetic with monotone transitive closure.	Reduction from reachability in Petri nets with inhibitor arcs [85].	Reduction to emptiness of expressions built from operators operating on sets of multisets [85].
BVAS covering and boundedness problems	Doubly exponential time. Reduction from acceptance in alternating doubly exponential bounded counter machines [20].	Doubly exponential time. Extension of Rackoff's induction to BVAS [20].
Deducibility in the theory of tensor $\star$.	Polynomial time reduction from Petri net reachability [94].	Polynomial time reduction to Petri net reachability [94].
Deducibility in conjunction, implication, fusion fragment of Relevance logic.	Reduction from termination of doubly exponential bounded 3-counter machines to the word problem in commutative contractive semi-Thue systems to deducibility [95].	McAloon's ordinal recursive bound on Dickson's lemma [95].
Validity in LR+ fragment of Relevance logic.	Reduction from termination of Ackermann bounded 3-counter machines to termination of ECMs with zero tests to acceptance in expansive and-branching counter machines without zero tests to deducibility to validity [96].	McAloon's ordinal recursive bound on Dickson's lemma [96].

Chapter 6

Petri Nets and Benefit Depth

Communicating automata with buffers [11] is a model of concurrent communicating systems. In this chapter, we consider a small generalization where 1-safe Petri nets (which we call components) communicate through buffers. Thus we have a system model which allows communication by message-passing as well as by synchronization, since 1-safe Petri nets can model the latter.

We introduce a logic to express some counting properties of Petri nets. This logic can express coverability, boundedness and some extensions, so its model checking problem is EXPSPACE-hard. We consider the parameter benefit depth that we introduced in Chapter 4 and extend it to 1-safe Petri nets communicating through buffers. With this extension, benefit depth measures how much buffers can affect one another. Benefit depth is upper bounded by the number of buffers but it seems reasonable that, in a loosely coupled distributed system, the communication graph amongst buffers is not dense and benefit depth can be low. We will show PARAPSPACE upper bound for model checking the above mentioned logic with respect to this parameter. The main idea behind this result is that if a transition benefitting from a place occurs before a transition that does not benefit from that place, the two transitions can be swapped. This fact can then be used for finer combinatorial analysis of recurrence relations that are the main tools for Rackoff's EXPSPACE upper bounds [83].

6.1 System Model

Our results work for any Petri net. But we divide the net into bounded and unbounded portions to emphasize the fact that our problem formulation strictly generalizes reachability for 1-safe Petri nets. The model of systems we consider in this chapter consists of some 1-safe nets, called **components**, which can add or remove tokens to/from a set of unbounded places that we refer to as **buffers**.

Definition 6.1. *A* **net of communicating automata with buffers** *(we just use the word "net" in the rest of this chapter) is a Petri net where the set of places is partitioned into a set of* **buffers** B *and* **component** *places $C = P \setminus B$. We require that all places in C remain 1-bounded regardless of the number of tokens in the buffers at the initial marking. We will use the notation $|B| = b$ and $|C| = a$.*

Definition 6.2. *Recall the definition of the set of places $Ben(p) \subseteq P$ and the set of transitions $T_{ben}(p) \subseteq T$ benefited by a place p from Definition 4.9. $Ind(p) = P \setminus Ben(p)$ and $T_{ind}(p) = T \setminus T_{ben}(p)$ are the places and transitions independent of p. The* **benefit depth** *of a net is defined as $K = max\{|Ben(p) \cap B| - 1 \mid p \in B\}$.*

The diagram shown in Fig. 6.1 illustrates a communicating automaton with buffers. The

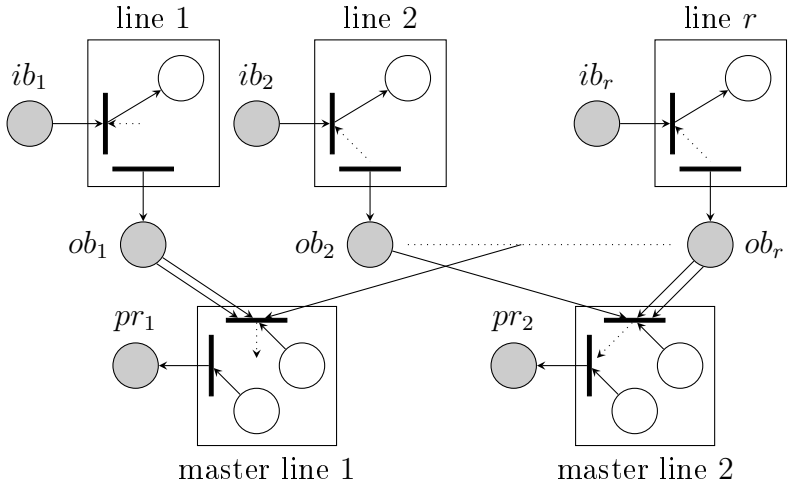


Figure 6.1: Illustration of communicating automata with buffers

boxes labelled as line 1, line 2 etc. can be thought of as assembly lines represented by 1-safe Petri nets, drawing raw materials from buffers ib_1, ib_2 etc. Output of these assembly lines are deposited into buffers ob_1, ob_2 etc. Boxes labelled master line 1 and master line 2 can be thought of as master assembly lines that use the output of earlier assembly lines as their input. They deposit their output in buffers pr_1 and pr_2 respectively. Irrespective of the number of assembly lines, benefit depth is 3 since only ob_i, pr_1 and pr_2 can benefit by decreasing tokens from ib_i.

6.2 Benefit Depth and Coverability

Recall the concept of vector from section 5.4. In this chapter, we will call only those functions $M : P \to \mathbb{Z}$ as vectors that satisfy $M(p) \in \{0, 1\}$ for all places $p \in C$. The following is same as Definition 5.14 adapted to nets of communicating automata with buffers.

Definition 6.3. *[83] Let $C \subseteq Q \subseteq P$. Define $lencov(Q, M, M_{cov})$ to be the length of the shortest Q-covering run from the vector M. If there is no such sequence, let $lencov(Q, M, M_{cov})$ to be 0. For $0 \leq i \leq b$, $\ell(i, M_{cov})$ is defined to be $max\{lencov(Q, M, M_{cov}) \mid M$ a vector, $C \subseteq Q \subseteq P$ and $|Q \setminus C| = i\}$. In this section we abbreviate $\ell(i, M_{cov})$ to $\ell(i)$. In section 6.4 we will abbreviate $\ell(b, M_{cov})$ to $\ell'(M_{cov})$.*

Definition 6.4. *Let $C \subseteq Q \subseteq P$ and $p \in B$ be a buffer. Define $covind^p(Q, M, M_{cov})$ to be the length of the shortest Q-covering run in $T_{ben}(p)^*$ from the vector M. If there is no such sequence, define $covind^p(Q, M, M_{cov})$ to be 0. Let $\ell_1(i) = max\{covind^p(Q, M, M_{cov}) \mid M$ a vector, p a buffer, $|Q \cap Ben(p) \cap B| = i\}$.*

Our strategy is to segregate covering sequences into two parts, the first made of transitions in $T_{ind}(p)$ and the second one made of transitions in $T_{ben}(p)$. We need the following technical lemma, which is a generalization of the **exchange lemma** [23, Lemma 2.14] to Petri nets with weighted arcs.

Lemma 6.5. *Let p be a place, transitions $t_{ben} \in T_{ben}(p)$ and $t_{ind} \in T_{ind}(p)$. Let $Q \subseteq P$ be some subset of places. If $t_{ben}t_{ind}$ is a Q-run from some vector M, then so is $t_{ind}t_{ben}$.*

Proof. We will first prove that t_{ind} is a Q-run from M. Suppose not. Now, suppose $p' \in Q$ is one of the places that do not have sufficient tokens at M to enable t_{ind}. Since $t_{ind} \in T_{ind}(p)$,

we know from Definition 6.2 that for all $p'' \in Ben(p)$, $Pre(p'', t_{ind}) = 0$. Hence, $p' \notin Ben(p)$, i.e., $p' \in Ind(p) \cap Q$. Now, we have $M \xrightarrow{t_{ben}} M_1 \xrightarrow{t_{ind}} M_2$ for some vector M_1, t_{ind} is a Q-run from M_1 but not from M since a place $p' \in Ind(p) \cap Q$ does not have enough tokens at M. Since p' has enough tokens at M_1, t_{ben} adds some tokens to p', i.e., $Post(p', t_{ben}) \geq 1$. This contradicts the fact that $t_{ben} \in T_{ben}(p)$. Therefore, t_{ind} is a Q-run from M. So, $M \xrightarrow{t_{ind}} M_3$ for some vector M_3.

Now, we will prove that t_{ben} is a Q-run from M_3. Suppose not. Let $p' \in Q$ be one of the places that do not have enough tokens at M_3 to enable t_{ben}. Since t_{ben} is a Q-run from M, t_{ind} must decrease the number of tokens in p'. Since $t_{ind} \in T_{ind}(p)$, we know from Definition 6.2 that t_{ind} does not decrease tokens in any place that belongs to $Ben(p)$. Hence, $p' \notin Ben(p)$, i.e., $p' \in Ind(p) \cap Q$. Let q' be the number of tokens in p' at M and let t_{ind} decrease the number of tokens in p' by q_1. Now, if $d_2 = Pre(p', t_{ben})$ is the number of tokens needed by t_{ben}, then $d_2 > q' - q_1$. Now, if $t_{ben}t_{ind}$ is run from M, number of tokens in p' at the end will be $q' - d_2 - q_1 < 0$ ($Post(p', t_{ben}) = 0$ since $p' \in Ind(p)$), which contradicts the fact that $t_{ben}t_{ind}$ is a Q-run from M. Therefore, p' cannot be in $Ind(p) \cap Q$ and hence there is no such p'. This means that t_{ben} is a Q-run from M_3 and hence $t_{ind}t_{ben}$ is a Q-run from M. □

In the following lemma, we give a finer analysis of a recurrence relation introduced by Rackoff in [83] that we mentioned in Lemma 5.15. The idea is that using the exchange lemma above, all transitions benefiting from a place can be moved to the right. Properties of the resulting sub-sequence can be used to get better upper bounds.

Lemma 6.6. *Let $\mathcal{N}$ be a net with maximum arc weight W, benefit depth K and let R be the maximum of the range of M_{cov}, the marking to be covered. If $K \leq i < b$, then $\ell(i+1) \leq (W\ell_1(K) + R)^{i+1}2^a + \ell(i) + \ell_1(K)$.*

Proof. Suppose that $Q_{i+1} = C \cup A$ where $|A| = i + 1$ and that there is a Q_{i+1}-covering run from some vector M. If this run is $W\ell_1(K) + R$-bounded for Q_{i+1}, then there is a similar run where no two intermediate vectors are equal when restricted to Q_{i+1}. The length of such a sequence is at most $(W\ell_1(K) + R)^{i+1}2^a$.

Otherwise, there is a Q_{i+1}-covering run from M that is *not* $W\ell_1(K)+R$-bounded for Q_{i+1}. Then there exist runs σ_1 and σ_2 such that $\sigma_1\sigma_2$ is Q_{i+1}-covering from M, σ_1 is $W\ell_1(K) + R$-bounded for Q_{i+1} and the final vector M' obtained by walking σ_1 at M has more than $W\ell_1(K) + R$ tokens at some place $p \in A$. Let $Q_i = Q_{i+1} \setminus \{p\}$. As above, we can assume that length of σ_1 is at most $(W\ell_1(K) + R)^{i+1}2^a$.

Now, σ_2 is a Q_i-covering run from M'. By definition, there is a Q_i-covering run σ_2' from M' whose length is at most $\ell(i)$. Since σ_2' is a Q_i-run from M', we can apply Lemma 6.5 repeatedly to rearrange σ_2' into another sequence $\tau_1\tau_2$ such that $\tau_1 \in T_{ind}(p)^*$, $\tau_2 \in T_{ben}(p)^*$, $\tau_1\tau_2$ is a Q_i-run from M' and $|\tau_1\tau_2| = |\sigma_2'|$ (see Fig. 6.2). This rearrangement of σ_2' could potentially cause places in C to get more than 1 token in an arbitrary Petri net. However, our assumption that places in C remain 1-bounded regardless of the number of tokens in the buffers at the initial marking ensures that the rearrangement doesn't disturb the 1-boundedness of places in C. Let M'' be the final vector obtained by walking τ_1 at M'. Now, $\tau_2 \in T_{ben}(p)^*$ and is a Q_i-covering run from M''. Hence, by Definition 6.4, there is a Q_i-covering run τ_2' from M'' with $\tau_2' \in T_{ben}(p)^*$ and $|\tau_2| \leq \ell_1(|Ben(p) \cap B| - 1)$. Since $|\tau_1| \leq \ell(i)$ and $\ell_1(|Ben(p) \cap B| - 1) \leq \ell_1(K)$, $|\tau_1\tau_2'| \leq \ell(i) + \ell_1(K)$. Since $\tau_1 \in T_{ind}(p)^*$, Definition 6.2 implies that no transition in τ_1 decreases tokens from p. Since $M''(p) \geq M'(p) \geq W\ell_1(K)+R$ and each transition in τ_2' removes at most W tokens from p, $\sigma_1\tau_1\tau_2'$ is a Q_{i+1}-covering run from M whose length is at most $(W\ell_1(K) + R)^{i+1}2^a + \ell(i) + \ell_1(K)$. □

The bound on $\ell(i+1)$ given by Rackoff in [83] is similar to the one in Lemma 6.6 but uses $\ell(i)$ in place of $\ell_1(K)$. Since $\ell_1(K)$ can be much smaller than $\ell(i)$, the bound in Lemma 6.6

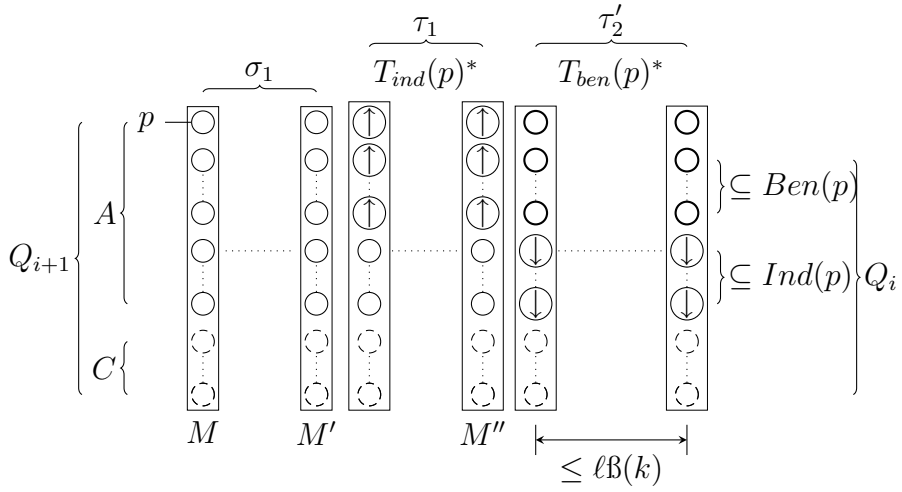


Figure 6.2: Sequences and bounds used in the proof of Lemma 6.6
$\uparrow$ (resp. $\downarrow$) inside places indicates that tokens are non-decreasing (resp. non-increasing).

is better. This is the fact that enables us to restrict exponential space complexity to K. The following lemma gives a recurrence relation for length of covering sequences made of transitions in $T_{ben}(p)$.

Lemma 6.7. *$\ell_1(0) \leq 2^a$ and $\ell_1(i+1) \leq (W\ell_1(i) + R)^{i+1}2^a + \ell_1(i)$.*

Proof. (Following [83].) We will first prove the bound on $\ell_1(0)$. Let $Q = C \cup A$ and $A \cap Ben(p) = \emptyset$ for some buffer p. Suppose $\sigma \in T_{ben}(p)^*$ is a Q-covering run from some vector M. If any two intermediate vectors reached by walking σ at M are equal when restricted to C, remove the subsequence between these two intermediate vectors. Since the removed subsequence never added any tokens to any place in A, such removals will never decrease tokens from places in A. Therefore, after all such removals, the sequence that is left is still a Q-covering run from M. The length of this run is at most 2^a.

Next, we will prove the bound on $\ell_1(i+1)$. Suppose that $Q = Q_{i+1} = C \cup A \cup A'$ where $|A'| = i+1$, with $A \cap Ben(p) = \emptyset$ for some buffer p. Suppose that there is a Q_{i+1}-covering run in $T_{ben}(p)^*$ from some vector M.

Case 1: There is a Q_{i+1}-covering run from M that is $W\ell_1(i) + R$-bounded for A'. Then, as above, there is a Q_{i+1}-covering run σ from M that is $W\ell_1(i) + R$-bounded for A' such that no two intermediate vectors obtained from walking σ at M are equal when restricted to $Q_{i+1} \setminus A$. The length of such a run is at most $(W\ell_1(i) + R)^{i+1}2^a$.

Case 2: Otherwise, there is a Q_{i+1}-covering run from M that is *not* $W\ell_1(i) + R$-bounded for A'. Then there exist sequences σ_1 and σ_2 such that $\sigma_1\sigma_2 \in T_{ben}(p)^*$ is a Q_{i+1}-covering run from M, σ_1 is $W\ell_1(i) + R$-bounded for A' and the final vector M' obtained by walking σ_1 at M has more than $W\ell_1(i) + R$ tokens at some place $p' \in A'$. Let $Q_i = Q_{i+1} \setminus \{p'\}$. As in case 1, we can assume that length of σ_1 is at most $(W\ell_1(i) + R)^{i+1}2^a$.

Now, $\sigma_2 \in T_{ben}(p)^*$ is a Q_i-covering run from M'. By definition, there is a Q_i-covering run $\sigma_2' \in T_{ben}(p)^*$ from M' whose length is at most $\ell_1(i)$. Since $M'(p) \geq W\ell_1(i) + R$ and each transition in σ_2' removes at most W tokens from p', $\sigma_1\sigma_2'$ is a Q_{i+1}-covering run from M whose length is at most $(W\ell_1(i) + R)^{i+1}2^a + \ell_1(i)$. □

It now only remains to solve the recurrence relations we have obtained and use them in a nondeterministic algorithm that guesses covering sequences to get our first main theorem.

Definition 6.8. *Let $W' = \max\{W, 2\}$, $R' = \max\{R, 2\}$. Define a growth function $g : \mathbb{N} \to \mathbb{N}$ as $g(0) = W'R'2^a$ and $g(i+1) = (g(i))^{3(i+1)}2^a$.*

Lemma 6.9. $\ell(K+j) \leq (K+j)(W\ell_1(K)+R)^{K+j}2^a + j\ell_1(K) + \ell(K)$.

Proof. By induction on j. The base case $j = 0$ is clear since RHS of the inequation is at least $\ell(K)$.

$$\begin{aligned}
\ell(K+j+1) &\leq (W\ell_1(K)+R)^{K+j+1}2^a + \ell(K+j) + \ell_1(K) \\
&\leq (W\ell_1(K)+R)^{K+j+1}2^a \\
&\quad + (K+j)(W\ell_1(K)+R)^{K+j}2^a + j\ell_1(K) \\
&\quad + \ell(K) + \ell_1(K) \\
&\leq (K+j+1)(W\ell_1(K)+R)^{K+j+1}2^a + (j+1)\ell_1(K) \\
&\quad + \ell(K)
\end{aligned}$$

□

Lemma 6.10. $\ell_1(i) \leq g(i)$, $\ell(i) \leq g(i)$, $\ell(K+j) \leq (K+j)(g(K))^{3(K+j)}2^a$ *and* $g(i) \leq (W'R')^{3^i i!}2^{6^i i! a}$.

Proof. Bounds on $\ell_1(i)$ and $\ell(i)$ are by induction on i. For the base case $i = 0$, we have $\ell_1(0) \leq 2^a \leq g(0)$ and $\ell(0) \leq 2^a \leq g(0)$ (this bound on $\ell(0)$ can be obtained by arguments similar to those used for the bound on $\ell_1(0)$ in Lemma 6.7).

$$\begin{aligned}
\ell_1(i+1) &\leq (W\ell_1(i)+R)^{i+1}2^a + \ell_1(i) \\
&\leq (Wg(i)+R)^{i+1}2^a + g(i) \\
&\leq (W'R')^{i+1}(g(i))^{i+1}2^a + g(i) \\
&\leq (g(i))^{2(i+1)}2^a + g(i) \\
&\leq (g(i))^{3(i+1)}2^a \\
&= g(i+1)
\end{aligned}$$

For the bound on $\ell(i)$, we will use Rackoff's result from [83], which states that $\ell(i+1) \leq (W\ell(i)+R)^{i+1}2^a + \ell(i)$.

$$\begin{aligned}
\ell(i+1) &\leq (W\ell(i)+R)^{i+1}2^a + \ell(i) \\
&\leq (Wg(i)+R)^{i+1}2^a + g(i) \\
&\leq (g(i))^{3(i+1)}2^a \\
&= g(i+1)
\end{aligned}$$

Next, we will prove the bound on $\ell(K+j)$.

$$\begin{aligned}
\ell(K+j) &\leq (K+j)(Wg(K)+R)^{K+j}2^a + jg(K) + g(K) \\
&\leq (K+j)(W'R')^{K+j}(g(K))^{K+j}2^a + (j+1)g(K) \\
&\leq (K+j)(g(K))^{3(K+j)}2^a
\end{aligned}$$

Finally, the bound on $g(i)$ is by induction i. For the base case $i = 0$, we have $g(0) = (W'R')2^a = (W'R')^{3^0 0!}2^{6^0 0! a}$.

$$\begin{aligned}
g(i+1) &= (g(i))^{3(i+1)}2^a \\
&\leq \left((W'R')^{3^i i!}2^{6^i i! a}\right)^{3(i+1)}2^a \\
&\leq (W'R')^{3^{i+1}(i+1)!}2^{(6^i i! 3(i+1)+1)a} \\
&\leq (W'R')^{3^{i+1}(i+1)!}2^{(6^i i! 3(i+1)2)a} \\
&= (W'R')^{3^{i+1}(i+1)!}2^{6^{i+1}(i+1)! a}
\end{aligned}$$

□

Theorem 6.11. *Let $q(i) = (6^{K+2}K!m^2)^i$. Suppose a net under consideration has benefit depth K. There is a non-deterministic algorithm that decides if there is a firing sequence covering M_{cov} from M_0 in space $\mathcal{O}(\log|M_0| + \log n + q(1)\log W' \log R' m \log m)$.*

Proof. Since there are b buffers in the net, $\ell(b)$ gives an upper bound on the length of the shortest P-covering run. Therefore, there exists a P-covering run iff there is one of length at most $\ell(b)$. From Lemma 6.10 we get

$$\ell(b) \leq b(g(K))^{3b}2^a \leq m(g(K))^{3m}2^a \leq m\left((W'R')^{3^K K!}2^{6^K K!a}\right)^{3m}2^a \leq m\left((W'R')^{6^{K+1}K!a}\right)^{3m}2^a$$

Hence $\ell(b) \leq m(W'R')^{6^{K+2}K!m^2}$. A nondeterministic algorithm can guess a sequence of transitions of this length and verify that it is P-covering from M_0. The memory needed is dominated by a counter to count up to maximum $\ell(b)$ and the memory needed to store intermediate markings. The memory needed for the counter is $\mathcal{O}(q(1)\log m \log W' \log R')$ and to store markings we need $\mathcal{O}(\log|M_0| + \log n + q(1)m \log m \log W' \log R')$. □

Given a net, its benefit depth K can be computed in polynomial time. Hence, the upper bound on the memory requirement in the above theorem is space constructible and the well known Savitch's theorem can be applied to determinize the above algorithm. The memory required will still be polynomial in the size of the input net and this gives us the ParaPspace algorithm.

6.3 Logics for Specifying Petri Net Properties

Let $\mathcal{N} = (P, T, Pre, Post)$ be a Petri net. Following is a logic of counting properties such that its model checking can be reduced to coverability (κ) and boundedness (β) problems, but is designed to avoid expressing reachability. This is a fragment of counting CTL.

$$\begin{aligned}
\tau &::= p \in P \mid \tau_1 + \tau_2 \mid c\tau,\ c \in \mathbb{N} \\
\kappa &::= \tau \geq c,\ c \in \mathbb{N} \mid \kappa_1 \wedge \kappa_2 \mid \kappa_1 \vee \kappa_2 \mid \mathbf{EF}\kappa \\
\beta &::= \{\tau_1, \ldots, \tau_r\} < \omega \mid \neg\beta \mid \beta_1 \vee \beta_2 \\
\phi &::= \beta \mid \kappa \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2
\end{aligned}$$

Formally, the semantics of the above logic is defined on the computation tree generated by a Petri net, rooted at the initial marking. We give the semantics below in an equivalent format based on the Petri net itself. The satisfaction of a formula ϕ by a Petri net $\mathcal{N}$ with initial marking M_0 (denoted as $\mathcal{N}, M_0 \models \phi$) is defined below. The Boolean operators work as usual. Note that every term τ gives a function $L_\tau : P \to \mathbb{N}$ such that τ is syntactically equivalent to $\sum_{p\in P} L_\tau(p)p$. By fixing an order on the set of places P, we can treat any function L_τ and marking M as a vector with $|P|$ coordinates.

- $\mathcal{N}, M_0 \models \tau \geq c$ iff $L_\tau \odot M_0 \geq c$, where $\odot$ is the vector dot product as in the previous chapter.
- $\mathcal{N}, M_0 \models \mathbf{EF}\kappa$ iff there is a marking M reachable from M_0 such that $\mathcal{N}, M \models \kappa$.
- $\mathcal{N}, M_0 \models \{\tau_1, \ldots, \tau_r\} < \omega$ iff $\exists c \in \mathbb{N}$ such that for all markings M reachable from M_0, there is a $j \in \{1, \ldots, r\}$ such that $\sum_{p\in P} L_{\tau_j}(p)M(p) \leq c$.

With the example of Fig. 6.1, the formula $\mathbf{EF}(ob_1 + ob_2 \geq 50 \wedge \mathbf{EF}(pr_1 + pr_2 \geq 30))$ is satisfied if we can reach a marking M such that $M(ob_1) + M(ob_2) \geq 50$ and from M, we can reach a marking M' such that $M'(pr_1) + M'(pr_2) \geq 30$.

6.4 Model Checking Logic of Counting Properties

Though the ParaPspace result for coverability can be extended to the boundedness problem, we consider the more general problem of checking whether a given net satisfies a given formula ϕ of the logic defined in section 6.3. We show that this can be done in ParaPspace with benefit depth as the parameter. First of all, we simplify the kind of formulas that our algorithm has to handle by nondeterministically choosing a disjunct from a disjunctive sub-formula. We then end up with ϕ a sequence of conjuncts $\beta_1, \ldots, \beta_c, \kappa$, where each β_i is of the form $\{\tau_1, \cdots \tau_r\} < \omega$ or $\{\tau_1, \cdots \tau_r\} = \omega$ and κ consists of conjunctions of nested **EF** modalities over $\tau \geq c$ formulas.

6.4.1 Nested Coverability Properties

We first consider verifying the formulas κ, which are of the form $\gamma \wedge \mathbf{EF}(\kappa_1) \wedge \cdots \wedge \mathbf{EF}(\kappa_r)$, with γ being a conjunction of $\tau \geq c$ formulas. We call γ the **content** of κ and $\kappa_1, \ldots, \kappa_r$ as the **children** of κ. Each of the children may have their own content and children, thus generating a tree with nodes Γ, with κ at the root of this tree. We will represent nodes of this tree by sequences of natural numbers, 0 being the root.

The maximum length of sequences in Γ is one more than the nesting depth of the **EF** modality in κ and we denote it by D. Let $[D] = \{0, 1, \ldots, D-1\}$. If $\alpha \in \Gamma$ is a tree node that represents the formula $\kappa(\alpha) = \gamma \wedge \mathbf{EF}(\kappa_1) \wedge \cdots \wedge \mathbf{EF}(\kappa_r)$, $content(\alpha) = \gamma$ denotes the content of the node α. Let $ratio(\tau \geq c) = max\{\lceil c/L_\tau(p) \rceil \mid L_\tau(p) \neq 0, p \in P\}$. Defining $\max(\emptyset) = 0$, we define the maximum ratio at height i in the tree by $ratio(i) = \max\{ratio(\tau \geq c) \mid \tau \geq c$ appears as a conjunct in $content(\alpha)$ for some $\alpha \in \Gamma, |\alpha| = i+1\}$. Recall from Definition 6.3 that b is the number of buffers and $\ell'(M)$ the length of the shortest run covering M using all the buffers $\ell(b, M)$.

Definition 6.12. *Given a formula κ and a net $\mathcal{N}$ with initial marking M_0, the bound function $f : [D] \times P \to \mathbb{N}$ is defined as follows. We use $f(j)$ for the marking defined by $f(j)(p) = f(j,p)$.*

- $f(D-1, p) = ratio(D-1)$,
- $f(D-i, p) = \max\{ratio(D-i), W\ell'(f(D-i+1)) + f(D-i+1, p)\}$, $1 < i < D$,
- $f(0, p) = M_0(p)$.

A guess function $h : \Gamma \times P \to \mathbb{N}$ is any function that satisfies $h(\alpha, p) \leq f(|\alpha| - 1, p)$ for all $\alpha \in \Gamma$ and $p \in P$. If h is a guess function, $h(\alpha)$ is the marking defined by $h(\alpha)(p) = h(\alpha, p)$.

If a given system satisfies the formula $\kappa = \gamma \wedge \mathbf{EF}(\kappa_1) \wedge \cdots \wedge \mathbf{EF}(\kappa_r)$, then there exist firing sequences $\sigma_{01}, \ldots, \sigma_{0r}$ that are all enabled at the initial marking M_0 such that $M_0 \stackrel{\sigma_{0i}}{\Longrightarrow} M_{0i}$ and M_{0i} satisfies κ_i. In general, if κ generates a tree with set of nodes Γ, then there is a set of sequences $\{\sigma_\alpha \mid \alpha \in \Gamma \setminus \{0\}\}$ and set of markings $\{M_\alpha \mid \alpha \in \Gamma\}$ such that $M_\alpha \stackrel{\sigma_{\alpha j}}{\Longrightarrow} M_{\alpha j}$ for all $\alpha, \alpha j \in \Gamma$ and M_α satisfies $content(\alpha)$ for all $\alpha \in \Gamma$.

Lemma 6.13. *There exist sequences $\{\mu_\alpha \mid \alpha \in \Gamma \setminus \{0\}\}$ and markings $\{M_\alpha \mid \alpha \in \Gamma\}$ such that $M_\alpha \stackrel{\mu_{\alpha j}}{\Longrightarrow} M_{\alpha j}$ for all $\alpha, \alpha j \in \Gamma$ with M_α satisfying $content(\alpha)$ and $|\mu_\alpha| \leq \ell'(f(|\alpha| - 1))$ iff there exist sequences $\{\sigma_\alpha \mid \alpha \in \Gamma \setminus \{0\}\}$ and markings $\{M'_\alpha \mid \alpha \in \Gamma\}$ (M'_0 should be equal to M_0) such that $M'_\alpha \stackrel{\sigma_{\alpha j}}{\Longrightarrow} M'_{\alpha j}$ for all $\alpha, \alpha j \in \Gamma$ with M'_α satisfying $content(\alpha)$.*

Proof. ($\Rightarrow$) Since M_α satisfies *content*(α), we can take $M'_\alpha = M_\alpha$ and $\sigma_\alpha = \mu_\alpha$.

($\Leftarrow$) Consider the following guess function:

$$h(\alpha, p) = \begin{cases} M_0(p) & \text{if } \alpha = 0 \\ M'_\alpha(p) & \text{if } \alpha \neq 0 \text{ and } M'_\alpha(p) \leq f(|\alpha| - 1, p) \\ f(|\alpha| - 1, p) & \text{otherwise} \end{cases}$$

By definition, $h(\alpha) \leq M'_\alpha$ and $h(\alpha) \leq f(|\alpha| - 1)$. Since $\sigma_{\alpha j}$ is a firing sequence that covers $M'_{\alpha j}$ from M'_α, there exist sequences $\mu_{\alpha j}$ that cover $h(\alpha j)$ starting from M'_α whose length is at most $\ell'(h(\alpha j))$ (and hence at most $\ell'(f(|\alpha j| - 1))$). We claim that there exist markings $\{M_\alpha \mid \alpha \in \Gamma\}$ such that $M_\alpha \stackrel{\mu_{\alpha j}}{\Longrightarrow} M_{\alpha j}$ for all $\alpha, \alpha j \in \Gamma$ and that M_α satisfies *content*(α) for all $\alpha \in \Gamma$.

First, we claim that every $\mu_{\alpha j}$ can be fired from M_α and that every place p will satisfy at least one of the following two conditions:

1. $M_{\alpha j}(p) \geq M'_{\alpha j}(p)$
2. $M_{\alpha j}(p) \geq f(|\alpha j| - 1, p)$

We will prove this claim by induction on $|\alpha|$.

Base case: $|\alpha| = 1$. μ_{0j} is a firing sequence of length at most $\ell'(h(0j))$ that covers $h(0j)$ starting from M_0. The claim is clear by the definition of $h(0j)$.

Induction step: We want to prove that $\mu_{\alpha j}$ can be fired at M_α and that $M_{\alpha j}$ satisfies the stated claims. We will prove these for an arbitrary place p. By induction hypothesis, either $M_\alpha(p) \geq M'_\alpha(p)$ or $M_\alpha(p) \geq f(|\alpha| - 1, p)$.

First, suppose that $M_\alpha(p) \geq M'_\alpha(p)$. Since $\mu_{\alpha j}$ covers $h(\alpha j)$ starting from M'_α, $M_{\alpha j}(p) \geq h(\alpha j)(p)$ and there are no intermediate markings between M_α and $M_{\alpha j}$ where p receives negative number of tokens. Also, since $M_{\alpha j}(p) \geq h(\alpha j)(p)$, either $M_{\alpha j}(p) \geq M'_{\alpha j}(p)$ or $M(\alpha j)(p) \geq f(|\alpha j| - 1, p)$.

Second, suppose that $M_\alpha(p) \geq f(|\alpha| - 1, p)$. $|\mu_{\alpha j}| \leq \ell'(h(\alpha j))$ and $h(\alpha j) \leq f(|\alpha j| - 1)$ by definition. Hence $\ell'(h(\alpha j)) \leq \ell'(f(|\alpha j| - 1))$ and $|\mu_{\alpha j}| \leq \ell'(f(|\alpha j| - 1))$. By definition of $f(|\alpha| - 1, p)$, we get $M_\alpha(p) \geq W\ell'(f(|\alpha j| - 1)) + f(|\alpha j| - 1, p)$. The sequence $\mu_{\alpha j}$ will remove at most $W\ell'(f(|\alpha j| - 1))$ tokens from p and hence, at least $f(|\alpha j| - 1, p)$ tokens will be left in place p at marking $M_{\alpha j}$. Therefore, $M_{\alpha j}(p) \geq f(|\alpha j| - 1, p)$.

This completes the induction and hence the claim.

Now, we will prove that each M_α satisfies *content*(α). For each conjunct $\tau \geq c$ in *content*(α), we will prove that $\sum_{p \in P} L_\tau(p) M_\alpha(p) \geq c$, where L_τ is the positive linear combination represented by τ. If $c = 0$, then the required result can be obtained by just observing that both $L_\tau(p)$ and $M_\alpha(p)$ are positive for all $p \in P$. So suppose that $c \neq 0$. Let $Q_\tau = \{p \in P \mid L_\tau(p) \neq 0\}$. We distinguish between two cases:

1. For some $p \in Q_\tau$, $M_\alpha(p) \geq f(|\alpha| - 1, p)$. In this case, $M_\alpha(p) \geq f(|\alpha| - 1, p) \geq \frac{c}{L_\tau(p)}$. Hence, $L_\tau(p) M_\alpha(p) \geq c$.
2. For all $p \in Q_\tau$, $M_\alpha(p) < f(|\alpha| - 1, p)$. In this case, for all $p \in Q_\tau$, $M_\alpha(p) \geq M'_\alpha(p)$. Since M'_α satisfies *content*(α), we have $\sum_{p \in Q_\tau} L_\tau(p) M'_\alpha(p) \geq c$. Therefore, $\sum_{p \in Q_\tau} L_\tau(p) M_\alpha(p) \geq c$.

□

To derive an upper bound for $f(i)$ to use in a nondeterministic algorithm, let $R = \max\{ratio(\tau \geq c) \mid \tau \geq c \text{ is a sub-formula of } \kappa\}$, $R' = \max\{R, 2\}$ and $W' = \max\{W, 2\}$. Recall that $D - 1$ is the nesting depth of **EF** and note that boundedness and coverability can be expressed with $D \leq 2$.

Lemma 6.14. *For $i \geq 2$, $f(D-i,p) \leq (i+1)R'W\ell'(f(D-i+1))$.*

Proof. By induction on i.

Base case: $i = 2$

$$\begin{aligned} f(D-2,p) &\leq \max\{R, W\ell'(f(D-1)) + f(D-1,p)\} \\ &\leq R + W\ell'(f(D-1)) + f(D-1,p) \\ &\leq R + W\ell'(f(D-1)) + R \\ &\leq 2R + W\ell'(f(D-1)) \\ &\leq 2R'W\ell'(f(D-1)) \end{aligned}$$

Induction step:

$$\begin{aligned} f(D-i-1,p) &\leq \max\{R, W\ell'(f(D-i)) + f(D-i,p)\} \\ &\leq R + W\ell'(f(D-i)) + (i+1)R'W\ell'(f(D-i+1)) \\ &\leq R'W\ell'(f(D-i)) + (i+1)R'W\ell'(f(D-i)) \\ &= (i+2)R'W\ell'(f(D-i)) \end{aligned}$$

□

Lemma 6.15. *Let $q(i) = (6^{K+2}K!m^2)^i$ represent the term in the space bound of Theorem 6.11. Then*

1. *$\ell'(f(D-1)) \leq m(W'R')^{q(1)}$ and*
2. *$\ell'(f(D-i)) \leq m\prod_{j=D-i}^{D}((D-j+1)W'^2R'm)^{q(i+j+1-D)}$.*

Proof. The first result $\ell'(f(D-1)) \leq m(W'R')^{q(1)}$ is by Lemma 6.10. Next result is by induction on i.

Base case: $i = 2$. Since $f(D-2,p) \leq 2R'W\ell'(f(D-1))$ and $\ell'(f(D-2)) \leq m(W'r')^{q(1)}$ where $r' = \max\{f(D-2,p) \mid p \in P\}$, we get

$$\begin{aligned} \ell'(f(D-2)) &\leq m(W'2R'W\ell'(f(D-1)))^{q(1)} \\ &\leq m(2W'^2R'm(W'R')^{q(1)})^{q(1)} \\ &\leq m(2W'^2R'm)^{q(1)}(W'R')^{q(2)} \end{aligned}$$

Induction step: Since $f(D-i-1,p) \leq (i+2)R'W'\ell'(f(D-i))$, we have

$$\begin{aligned} \ell'(f(D-i-1)) &\leq m(W'(i+2)R'W'\ell'(f(D-i-1)))^{q(1)} \\ &\leq m\left((i+2)W'^2R'm\prod_{j=D-i}^{D}((D-j+1)W'^2R'm)^{q(i+j+1-D)}\right)^{q(1)} \\ &= m\left((i+2)W'^2R'm\right)^{q(1)}\prod_{j=D-i}^{D}\left((D-j+1)W'^2R'm\right)^{q(i+1+j+1-D)} \\ &= m\prod_{j=D-i-1}^{D}\left((D-j+1)W'^2R'm\right)^{q(i+1+j+1-D)} \end{aligned}$$

□

Theorem 6.16. *Given a net and a κ formula ϕ, if the benefit depth of the net is treated as a parameter and the nesting depth D of **EF** modality in the formula is treated as a constant, then there is a* ParaPspace *algorithm that checks if the net satisfies the given formula.*

Proof. By Lemma 6.13, it is enough for a nondeterministic algorithm to guess sequences $\sigma_{\alpha j}$, $\alpha j \in \Gamma$ of lengths at most $\ell'(f(|\alpha j| - 1))$ and verify that they satisfy the formula. Using bounds given by Lemma 6.15 and an argument similar to the one in the proof of Theorem 6.11, it can be shown that the space used is exponential in K and polynomial in the size of the net and numeric constants in the formula. This gives the ParaPspace algorithm. □

The space requirement of the above algorithm will have terms like m^{2D} and hence it will not be ParaPspace if D is treated as a parameter instead of a constant.

6.4.2 Boundedness Properties

We will now look at model checking β formulas. We will adapt the concept of *disjointness sequence* [19] to our notation. Let $\Delta[\sigma]$ be the total effect of walking the sequence of transitions $\sigma = t_1 \cdots t_r$ so that for any place p, $\Delta[\sigma](p) = \sum_{i=1}^{r} Post(p, t_i) - Pre(p, t_i)$ is the change in the number of tokens in p as a result of walking the sequence σ.

Definition 6.17 ([19])**.** *Let $X \subseteq B$ be a non-empty subset of buffers. A firing sequence σ enabled at the initial marking M_0 is said to be a **X-pumping sequence** if σ can be decomposed into $\sigma'_1\underline{\sigma_1}\sigma'_2\underline{\sigma_2}\cdots\sigma'_e\underline{\sigma_e}$ such that*

1. *$X \subseteq \{p \in B \mid \Delta[\underline{\sigma_\rho}](p) > 0, 1 \leq \rho \leq e\}$.*
2. *$\Delta[\underline{\sigma_1}](p) \geq 0$ for all places $p \in P$ and for each ρ between 2 and e, for each $p \in P$, $\Delta[\underline{\sigma_\rho}](p) < 0$ implies there is a $j \leq \rho - 1$ such that $\Delta[\underline{\sigma_j}](p) > 0$.*
3. *For each ρ between 1 and e, each $p \in C$, $\Delta[\underline{\sigma_\rho}](p) = 0$.*

*The subsequences $\underline{\sigma_1}, \underline{\sigma_2}, \ldots, \underline{\sigma_e}$ are called **pumping portions**. They are underlined to distinguish them from other parts of the sequence.*

Lemma 6.18 ([19])**.** *$\mathcal{N}, M_0 \models \{\tau_1, \ldots, \tau_r\} = \omega$ iff there exists a X-pumping sequence for some $X \subseteq P$ such that for every $j \in \{1, \cdots, r\}$, there is a $p_j \in X$ with $L_{\tau_j}(p_j) \geq 1$.*

Proof. ($\Leftarrow$) Suppose there is a X-pumping sequence σ as given in the lemma. Let $\sigma'_1\underline{\sigma_1}\cdots\sigma'_e\underline{\sigma_e}$ be the decomposition of σ as in Definition 6.17. By repeating the pumping portions $\underline{\sigma_1}, \ldots, \underline{\sigma_e}$ arbitrarily many times (see [19, Lemma 3.1]), we can ensure that for all $c \in \mathbb{N}$, $\exists M \in \mathcal{R}(\mathcal{N}, M_0)$ such that for all $j \in \{1, \ldots, r\}$, $\Sigma_{p \in P} L_{\tau_j}(p) M(p) \geq c$.

($\Rightarrow$) Suppose $\mathcal{N}, M_0 \models \{\tau_1, \ldots, \tau_r\} = \omega$. By semantics, we get $\forall c \in \mathbb{N}$, $\exists M \in \mathcal{R}(\mathcal{N}, M_0)$ such that for all $j \in \{1, \ldots, r\}$, $\sum_{p \in P} L_{\tau_j}(p) M(p) > c$. Hence, we can conclude that for all $c \in \mathbb{N}$, there are buffers $p_1^c, p_2^c, \ldots, p_r^c$ and $M^c \in \mathcal{R}(\mathcal{N}, M_0)$ such that $M^c(p_j^c) > c \wedge L_{\tau_j}(p_j^c) \geq 1$ for all $j \in \{1, \ldots, r\}$. For each $c \in \mathbb{N}$, let $X^c = \{p_1^c, \ldots, p_r^c\}$. Since the sequence $X^1, X^2, \ldots$ is infinite and there are only finitely many subsets of B, at least one subset of B occurs infinitely often in this sequence. Let X be this subset. We will now prove that there is a X-pumping sequence using some results about coverability trees [24, Section 4.6].

Recall [24] that in a coverability tree, markings $M : P \to \mathbb{N}$ are extended to ω-markings $\overline{M} : P \to \mathbb{N} \cup \{\omega\}$, by mapping unbounded places to ω. We first claim that there is some reachable ω-marking $\overline{M}$ in the coverability tree of $(\mathcal{N}, M_0)$ such that for all $p \in X$, $\overline{M}(p) = \omega$. Suppose not. Then, for every reachable ω-marking $\overline{M}$, there is some place $p \in X$ such that $\overline{M}(p) < \omega$. Let c be the maximum of such bounds. Then, by [24, Theorem 22], for every marking $M \in \mathcal{R}(\mathcal{N}, M_0)$, there exists $p \in X$ such that $M(p) \leq c$, a contradiction. Hence, there is a reachable ω-marking $\overline{M}$ in the coverability tree of $(\mathcal{N}, M_0)$ such that for all $p \in X$, $\overline{M}(p) = \omega$. Now, the required X-pumping sequence can be constructed (see [19, Lemma 3.1] for details). □

Definition 6.19 ([19]). *Let $Q \subseteq P$ be a subset of places such that $C \subseteq Q$, $X \subseteq B$ be a non-empty subset of buffers and M a vector. A sequence of transitions σ is said to be a M, Q-**enabled** X-**pumping sequence** if it can be decomposed into $\sigma'_1\underline{\sigma_1}\sigma'_2\underline{\sigma_2}\cdots\sigma'_e\underline{\sigma_e}$ such that*

1. *$X \subseteq \{p \in B \mid \Delta[\underline{\sigma_\rho}](p) > 0, 1 \leq \rho \leq e\}$.*
2. *$\Delta[\underline{\sigma_1}](p) \geq 0$ for all places $p \in P$ and for each ρ between 1 and e, each $p \in C$, $\Delta[\underline{\sigma_\rho}](p) = 0$.*
3. *For each ρ between 2 and e, for each $p \in B$, $\Delta[\underline{\sigma_\rho}](p) < 0$ implies there is a $j \leq \rho - 1$ such that $\Delta[\underline{\sigma_j}](p) > 0$.*
4. *For any intermediate vector M' occurring when walking σ from M and any $p \in Q$, $M'(p) \geq 0$.*

As we did for coverability, we could try to bound the lengths of M, Q-enabled X-pumping sequences by induction on $|Q|$. However, reasoning about M, Q-enabled X-pumping sequences may involve high numbers. Instead, we will reason about another structure defined below, whose existence will imply existence of M, Q-enabled X-pumping sequences.

Definition 6.20 ([19]). *Let $Q \subseteq P$ be a subset of places such that $C \subseteq Q$, $X, Y \subseteq B$ with X non-empty. Let M be a vector and $c \in \mathbb{N} \cup \{\omega\}$. A sequence of transitions σ is said to be a Y-**neglecting weakly** M, Q, c-**enabled** X-**pumping sequence** if it can be decomposed into $\sigma'_1\underline{\sigma_1}\sigma'_2\underline{\sigma_2}\cdots\sigma'_e\underline{\sigma_e}$ such that*

1. *$X \subseteq \{p \in B \mid \Delta[\underline{\sigma_\rho}](p) > 0, 1 \leq \rho \leq e\}$.*
2. *For each ρ between 1 and e, for each $p \in C$, $\Delta[\underline{\sigma_\rho}](p) = 0$.*
3. *For each ρ between 1 and e and any $p \in B$, $\Delta[\underline{\sigma_\rho}](p) < 0$ implies (there is a $j \leq \rho - 1$ such that $\Delta[\underline{\sigma_j}](p) > 0$ or $p \in Y$).*
4. *For any intermediate vector M' occurring while walking σ from M and any $p \in Q \setminus C$, $M'(p) < c$.*
5. *For any intermediate vector M' occurring while walking σ from M and any $p \in Q$, $M'(p) < 0$ implies ($p \in Y$ or there is a j such that $\underline{\sigma_j}$ occurs before M' and $\Delta[\underline{\sigma_j}](p) > 0$).*

Condition 5 above means that an intermediate vector M' may have negative number of tokens in a place p belonging to Q only if that place is in Y (all places in Y are neglected) or there is a way to pump up tokens in p before reaching M'.

Lemma 6.21 ([19]). *Let $Q \subseteq P$ be a subset of places such that $C \subseteq Q$, $X \subseteq B$ be a non-empty subset of buffers and M a vector. Suppose σ is a $\emptyset$-neglecting weakly M, Q, ω-enabled X-pumping sequence with decomposition $\sigma'_1\underline{\sigma_1}\sigma'_2\underline{\sigma_2}\cdots\sigma'_e\underline{\sigma_e}$ as in Definition 6.20. There are $n_1, n_2, \cdots, n_e \in \mathbb{N}$ such that $\sigma'_1\underline{\sigma_1}^{n_1}\sigma'_2\underline{\sigma_2}^{n_2}\cdots\sigma'_e\underline{\sigma_e}^{n_e}$ is a M, Q-enabled X-pumping sequence.*

Proof. We define $n_e, \ldots, n_1$ in that order as follows:

- $n_e = 1$.
- Suppose $1 \leq \rho < e$ and $n_{\rho+1}, \ldots, n_e$ have already been defined. Define n_ρ to be $(e - \rho)(|\sigma| - 1)W + \Sigma_{j=\rho+1}^{j=e}(|\sigma| - 1)Wn_j$.

Let

$$M \xrightarrow{\sigma'_1} M'_1 \xrightarrow{\underline{\sigma_1^{n_1}}} M_1 \xrightarrow{\sigma'_2} M'_2 \xrightarrow{\underline{\sigma_2^{n_2}}} M_2 \longrightarrow \cdots \xrightarrow{\sigma'_e} M'_e \xrightarrow{\underline{\sigma_e^{n_e}}} M_e \ .$$

We will verify that each of the 4 conditions of Definition 6.19 are satisfied by the sequence $\sigma'_1\underline{\sigma_1^{n_1}}\sigma'_2\underline{\sigma_2^{n_2}}\cdots\sigma'_e\underline{\sigma_e^{n_e}}$.

1. This condition follows from condition 1 of Definition 6.20.

2. The condition for each ρ between 1 and e, for each $p \in C$, $\Delta[\underline{\sigma_\rho^{n_\rho}}](p) = 0$ follows from condition 2 of Definition 6.20. Let us verify that $\Delta[\underline{\sigma_1^{n_1}}](p) \geq 0$ for all places $p \in P$. If this is not the case, then $\underline{\sigma_1}$ decreases the number of tokens in some buffer, contradicting condition 3 of Definition 6.20.

3. This follows from condition 3 of Definition 6.20.

4. We will prove the following claim by induction on ρ: for any intermediate vector M' between M and M_ρ and any $p \in Q$, $M'(p) \geq 0$ and for any intermediate vector M'' (that can occur anywhere) and any $p' \in (Q \cap \bigcup_{j=1}^{j=\rho}\{p \in B \mid \Delta[\underline{\sigma_j^{n_j}}](p) > 0\})$, $M''(p') \geq 0$.

   Base case $\rho = 1$: Let $p \in Q$. For any intermediate vector M' between M and M'_1, $M'(p) < 0$ would imply a contradiction of condition 5 of Definition 6.20. For any intermediate marking M' between M'_1 and M_1, $M'(p) < 0$ would imply a contradiction of condition 3 or 5 of Definition 6.20. Hence, $M'_1(p) \geq 0$ for any $p \in Q \cap \{p' \in B \mid \Delta[\underline{\sigma_1}](p') > 0\}$. Hence, for any $p \in Q \cap \{p' \in B \mid \Delta[\underline{\sigma_1^{n_1}}](p') > 0\}$, $M_1(p) \geq (e-1)(|\sigma|-1)W + \Sigma_{j=2}^{j=e}(|\sigma|-1)Wn_j$. After M_1, what remains is $\sigma'_2, \ldots, \sigma'_e$ and $\underline{\sigma_2^{n_2}}, \ldots, \underline{\sigma_e^{n_e}}$. Their lengths are bounded by $(e-1)(|\sigma|-1)$ and $\Sigma_{j=2}^{j=e}(|\sigma|-1)n_j$ respectively. Since no transition can decrease more than W tokens at a time, it follows that no place in $Q \cap \{p' \in B \mid \Delta[\underline{\sigma_1^{n_1}}](p') > 0\}$ will ever have negative number of tokens after M_1.

   Induction step: Let $p \in Q$ and M' be any intermediate marking between M_ρ and $M_{\rho+1}$. Suppose $M'(p) < 0$. By induction hypothesis, $p \in Q \setminus \bigcup_{j=1}^{j=\rho}\{p' \in B \mid \Delta[\underline{\sigma_j}](p') > 0\}$. We would then get a contradiction to conditions 3 and 5 of Definition 6.20. This means for any place $p \in Q \cap \{p' \in B \mid \Delta[\underline{\sigma_{\rho+1}}](p') > 0\}$, $M'_{\rho+1}(p) \geq 0$. Hence, $M_{\rho+1}(p)$ contains enough tokens to ensure that p will never have negative number of tokens in any intermediate vector after $M_{\rho+1}$.

□

Thanks to Lemma 6.21, it is sufficient to check the existence of $\emptyset$-neglecting weakly M_0, Q, ω-enabled X-pumping sequences. Any M, Q-enabled X-pumping sequence is a $\emptyset$-neglecting weakly M, Q, ω-enabled X-pumping sequence by definitions, so it is also necessary to check the existence of $\emptyset$-neglecting weakly M_0, Q, ω-enabled X-pumping sequences. We will bound the lengths of such sequences by induction on $|Q|$.

Definition 6.22. *Let $C \subseteq Q \subseteq P$ and $p \in B$ be a buffer. Let $X, Y \subseteq B$ be subsets of buffers with X non-empty and M be a vector. Define $\lambda^p(Q, M, X, Y)$ to be the length of the shortest $(Ind(p) \cup Y)$-neglecting weakly M, Q, ω-enabled X-pumping sequence in $T_{ben}(p)^*$. If there is no such sequence, define $\lambda^p(Q, M, X, Y)$ to be 0. Let $\lambda_1(i) = \max\{\lambda^p(Q, M, X, Y) \mid M$ a vector, $X, Y \subseteq B, X \neq \emptyset, |Q \cap Ben(p) \cap B| = i\}$. Also define $\lambda(Q, M, X, Y)$ to be the length of the shortest Y-neglecting weakly M, Q, ω-enabled X-pumping sequence and 0 if there is no such sequence. Let $\lambda(i) = \max\{\lambda(Q, M, X, Y) \mid M$ a vector, $|Q \cap B| = i, X, Y \subseteq B, X \neq \emptyset\}$.*

During the induction, we will need to split, rearrange and combine sequences to build new sequences. The following proposition states that under certain conditions, two pumping sequences can be combined to create one pumping sequence.

Proposition 6.23. *Let $Y, X^1, X^2 \subseteq B$ be subsets of buffers with $X^1 \cap X^2 = \emptyset$, $X^1, X^2 \neq \emptyset$ and M a vector. Let $C \subseteq Q \subseteq P$ be a subset of places. Let σ_1 be a Y-neglecting weakly M, Q, ω-enabled X^1-pumping sequence, ending at some vector M^1. Let M^2 be any vector such that for all $p \in Q \setminus (Y \cup X^1)$, $M^2(p) = M^1(p)$. Let σ_2 be a $Y \cup X^1$-neglecting weakly M^2, Q, ω-enabled X^2-pumping sequence. Then, $\sigma_1\sigma_2$ is a Y-neglecting weakly M, Q, ω-enabled $(X^1 \cup X^2)$-pumping sequence.*

Proof. According to Definition 6.20, let the decomposition of σ_1 be

$$M \xrightarrow{\delta_1'} M_1' \xrightarrow{\underline{\delta_1}} M_1 \longrightarrow \cdots \xrightarrow{\delta_e'} M_e' \xrightarrow{\underline{\delta_e}} M^1$$

and that of σ_2 be

$$M^2 \xrightarrow{\eta_1'} M_1^{2\prime} \xrightarrow{\underline{\eta_1}} M_1^2 \longrightarrow \cdots \xrightarrow{\eta_{e'}'} M_{e'}^{2\prime} \xrightarrow{\underline{\eta_{e'}}} M^3 \ .$$

Now consider the concatenated sequence

$$M \xrightarrow{\delta_1'} M_1' \xrightarrow{\underline{\delta_1}} M_1 \longrightarrow \cdots \xrightarrow{\delta_e'} M_e' \xrightarrow{\underline{\delta_e}} M^1 \xrightarrow{\eta_1'} M_1^{3\prime} \xrightarrow{\underline{\eta_1}} M_1^3 \longrightarrow \cdots \xrightarrow{\eta_{e'}'} M_{e'}^{3\prime} \xrightarrow{\underline{\eta_{e'}}} M^4 \ .$$

We will show that the above decomposition satisfies all the conditions of a Y-neglecting weakly M, Q, ω-enabled $(X^1 \cup X^2)$-pumping sequence. We will prove that each of the 5 conditions of Definition 6.20 is true for the above decomposition.

1. This follows from the fact that σ_1 and σ_2 satisfy condition 1 of Definition 6.20.
2. This follows from the fact that σ_1 and σ_2 satisfy condition 2 of Definition 6.20.
3. For some ρ and $p \in B$, if $\Delta[\underline{\delta_\rho}](p) < 0$, then the fact that σ_1 satisfies condition 3 of Definition 6.20 suffices. Suppose $\Delta[\underline{\eta_\rho}](p) < 0$. From the fact that σ_2 satisfies condition 3 of Definition 6.20, we get that either there is a $j \leq \rho - 1$ with $p \in \{p' \in B \mid \Delta[\underline{\eta_j}](p') > 0\}$ or $p \in (Y \cup X^1)$. In both cases, the new sequence will satisfy condition 3 of Definition 6.20.
4. This condition follows since in this case $c = \omega$.
5. If $p \in Q \setminus (Y \cup X^1)$ and M' is any intermediate marking between M^1 and M_1^3, then $M'(p) \geq 0$ from the fact that σ_2 satisfies condition 5 of Definition 6.20 and that $M^2(p) = M^1(p)$. If M' is an intermediate marking after M_1^3 and $M'(p) < 0$, then we will have some j such that $\underline{\eta_j}$ occurs before M' with $p \in \{p' \in B \mid \Delta[\underline{\eta_j}](p') > 0\}$ due to the fact that σ_2 satisfies condition 5 of Definition 6.20.

□

Pumping portions of weakly enabled pumping sequences can be repeated without losing their property.

Lemma 6.24 ([19])**.** *Suppose $\sigma = \sigma_1'\underline{\sigma_1}\sigma_2' \cdots \sigma_e'\underline{\sigma_e}$ is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence. Then the sequence $\sigma' = \sigma_1'\sigma_1^{n_1}\underline{\sigma_1}\sigma_1^{n_1'}\sigma_2' \cdots \sigma_e'\sigma_e^{n_e}\underline{\sigma_e}$ is also a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence for any $n_1, n_1', \ldots, n_e \in \mathbb{N}$ (σ_ρ is same as $\underline{\sigma_\rho}$, except that σ_ρ is not considered a pumping portion while $\underline{\sigma_\rho}$ is considered a pumping portion).*

Proof. We will prove that the new sequence satisfies all the conditions of Definition 6.20. Conditions 1,2 and 3 are satisfied since the set of pumping portions of the new sequence is equal to that of the old one and occurs in the same order. Condition 4 is trivially satisfied since in this case, $c = \omega$. Suppose for some intermediate vector M' and some place $p \in Q$, $M'(p) < 0$. Let j be the maximum number such that $\underline{\sigma_j}$ occurs before M'. Suppose $M \xrightarrow{\sigma'_1\sigma_1^{n_1}\underline{\sigma_1}\sigma_1'^{n_1'}\sigma'_2\cdots\sigma'_j\sigma_j^{n_j}\underline{\sigma_j}} M''$ and $M'' \xrightarrow{\eta} M'$. If $p \in Y$ or $p \in \bigcup_{j'=1}^{j}\{p' \in P \mid \Delta[\underline{\sigma_{j'}}](p') > 0\}$, there is nothing else to prove. Otherwise, $\Delta[\underline{\sigma_{j'}}](p) = 0$ for every j' between 1 and j. This implies that if $M \xrightarrow{\sigma'_1\underline{\sigma_1}\sigma'_2\cdots\sigma'_j\underline{\sigma_j}} M_2$ and $M_2 \xrightarrow{\eta} M_3$, then $M_3(p) < 0$, contradicting the fact that σ satisfies condition 5 of Definition 6.20. □

Now, we will prove a generalization of Lemma 5.17, which states that if there is a weakly enabled pumping sequence in which the tokens in some subset Q of places always remains less than some $c \in \mathbb{N}$, then there is one such sequence that is not too long. The proof is essentially an adaptation of the proof of [19, Lemma 4.2].

Lemma 6.25 ([19]). *Let $X, Y \subseteq B$ be subsets of buffers with X being non-empty. Let $C \subseteq Q \subseteq P$ be a subset of places, M a vector and $c \in \mathbb{N}$ a number. Suppose there is a Y-neglecting weakly M, Q, c-enabled X-pumping sequence σ that decomposes into $\sigma'_1\underline{\sigma_1}\cdots, \sigma'_e\underline{\sigma_e}$ as in Definition 6.20. Then, there is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence σ' of length at most $e(Wc2^a)^{poly(m)}$, where $poly()$ is a polynomial whose degree does not depend on m, W, e, a or c. In addition, if $M \xrightarrow{\sigma} M'$, then σ' will satisfy the property that $M \xrightarrow{\sigma'} M''$ and for all $p \in Q \setminus (Y \cup X)$, $M''(p) = M'(p)$.*

Proof. We will prove the result by induction on e. If $e = 1$, we have just the sequence $M \xrightarrow{\sigma'_1} M'_1 \xrightarrow{\sigma_1} M_1$. If any two intermediate vectors between M and M'_1 is same with respect to $Q \setminus Y$, the subsequence between them can be removed without violating the remaining sequence's property. Repeating this as many times as needed, we can shorten σ'_1 to be of length at most $(c2^a)^{|Q|}$. Let a $(Q \setminus Y)$-loop be any subsequence of $\underline{\sigma_1}$ whose effect on all places in $Q \setminus Y$ is 0 such that no subsequence inside this loop satisfies the same property. As is done in [87, Lemma 2.2], $(Q \setminus Y)$-loops can be carefully removed from σ_1 without affecting its X-pumping property such that the total length of shortened σ'_1 and $\underline{\sigma_1}$ is at most $(Wc2^a)^{poly(m)}$. Since we only removed $Q \setminus Y$-loops from σ to obtain the new sequence σ', it satisfies the condition that $M \xrightarrow{\sigma'} M''$ and for all $p \in Q \setminus (Y \cup X)$, $M''(p) = M_1(p)$.

For the induction step, suppose σ decomposes as $\sigma'_1\underline{\sigma_1}\cdots\sigma'_{e+1}\underline{\sigma_{e+1}}$ and for $1 \leq \rho \leq e+1$, $X_\rho = \{p \in B \mid \Delta[\underline{\sigma_\rho}](p) > 0\}$. Let $M \xrightarrow{\sigma'_1\sigma_1} M_1$. We observe that $\sigma'_2\underline{\sigma_2}\cdots\sigma'_{e+1}\underline{\sigma_{e+1}}$ is a $(Y \cup X_1)$-neglecting weakly M_1, Q, c-enabled $(X_2 \cup \cdots \cup X_{e+1})$-pumping sequence. By induction hypothesis, there is a $(Y \cup X_1)$-neglecting weakly M_1, Q, ω-enabled $(X_2 \cup \cdots \cup X_{e+1})$-pumping sequence δ of length at most $e(Wc2^a)^{poly(m)}$. As is done in the base case, σ'_1 and $\underline{\sigma_1}$ can be replaced by shorter δ'_1 and $\underline{\delta_1}$ respectively such that $\delta'_1\underline{\delta_1}$ is a Y-neglecting weakly M, Q, ω-enabled X_1-pumping sequence of length at most $(Wc2^a)^{poly(m)}$. Since the change from $\sigma'_1\underline{\sigma_1}$ to $\delta'_1\underline{\delta_1}$ only involves removal of $(Q \setminus Y)$-loops, we have $M \xrightarrow{\delta'_1\delta_1} M^2$ such that for all $p \in (Q \setminus Y)$, $M^2(p) = M_1(p)$. By Proposition 6.23, $\delta'_1\underline{\delta_1}\delta$ is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence. Its length is at most $(e+1)(Wc2^a)^{poly(m)}$. If $M_1 \xrightarrow{\sigma'_2\underline{\sigma_2}\cdots\sigma'_{e+1}\underline{\sigma_{e+1}}} M_{e+1}$ and $M_1 \xrightarrow{\delta} M'_{e+1}$, we have by induction hypothesis that for all $p \in Q \setminus (Y \cup X_2 \cup \cdots \cup X_{e+1})$, $M'_{e+1}(p) = M(e+1)(p)$. Combining this with the property that for all $p \in (Q \setminus Y)$, $M^2(p) = M_1(p)$, we conclude that $M \xrightarrow{\delta'_1\underline{\delta_1}\delta} M''_{e+1}$ for some vector M''_{e+1} with $M''_{e+1}(p) = M_{e+1}(p)$ for all $p \in Q \setminus (Y \cup X)$. □

We are now ready to give the most important recurrence relation for $\lambda(i)$, extending Lemma 5.19. Recall that K denotes $\max\{|Ben(p) \cap B| - 1 \mid p \in B\}$ and b denotes the number of buffers.

Lemma 6.26. *For $0 \leq i < b$, $\lambda(i+1) \leq m(W^2\lambda_1(K)2^a)^{poly(m)} + \lambda(i) + \lambda_1(K)$, where poly() is a polynomial whose degree is a constant independent of m, W, K, a and i.*

Proof. Let $C \subseteq Q \subseteq P$ be a subset of places such that $|Q \cap B| = i + 1$. Suppose σ is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence for some vector M and $X, Y \subseteq B$, $X \neq \emptyset$.

Case 1: σ is a Y-neglecting weakly $M, Q, W\lambda_1(K)$-enabled X-pumping sequence. By Lemma 6.25, there is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence of length at most $m(W^2\lambda_1(K)2^a)^{poly(m)}$.

Case 2: Suppose that according to Definition 6.20, σ decomposes as

$$M \xrightarrow{\sigma'_1} M'_1 \xrightarrow{\underline{\sigma_1}} M_1 \longrightarrow \cdots \xrightarrow{\sigma'_e} M'_e \xrightarrow{\underline{\sigma_e}} M_e$$

and there is some $\rho \in \{2, \ldots, e\}$ and an intermediate vector M' between $M_{\rho-1}$ and strictly before $M_{\rho'}$ (or between M and strictly before M'_1) such that for some buffer $p' \in Q \cap B$, $M'(p') \geq W\lambda_1(K)$. Let M' be the first such vector. For $1 \leq j \leq e$, let $X_j = \{p \in B \mid \Delta[\underline{\sigma_j}](p) > 0\}$. If there is some $\rho > 1$ such that $\{p \in P \mid \Delta[\underline{\sigma_\rho}](p) > 0\} \subseteq \bigcup_{j=1}^{\rho-1}\{p \in P \mid \Delta[\underline{\sigma_j}](p) > 0\}$, then $\underline{\sigma_\rho}$ can be considered as a non-pumping portion and the resulting sequence will still be a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence. Hence, without loss of generality, we can assume that $e \leq m$.

Suppose $M_{\rho-1} \xrightarrow{\sigma^{1'}_\rho} M' \xrightarrow{\sigma^{2'}_\rho} M'_\rho$. The sequence $\sigma'_1\underline{\sigma_1} \cdots \sigma'_{\rho-1}\underline{\sigma_{\rho-1}}$ is a Y-neglecting weakly $M, Q, W\lambda_1(K)$-enabled $(X_1 \cup \cdots X_{\rho-1})$-pumping sequence. By Lemma 6.25, there is a Y-neglecting weakly M, Q, ω-enabled $(X_1 \cup \cdots X_{\rho-1})$-pumping sequence δ of length at most $(\rho-1)(W^2\lambda_1(K)2^a)^{poly(m)}$ such that $M \xrightarrow{\delta} M''_{\rho-1} \xrightarrow{\sigma^{1'}_\rho} M''$ where for all $p \in (Q \setminus (Y \cup \bigcup_{j=1}^{\rho-1} X_j))$, $M''(p) = M'(p)$.

The sequence $\sigma^{2'}_\rho\underline{\sigma_\rho}\sigma'_{\rho+1} \cdots \underline{\sigma_e}$ is a $(Y \cup X_1 \cup \cdots \cup X_{\rho-1})$-neglecting weakly M', Q, ω-enabled $(X_\rho \cup \cdots \cup X_e)$-pumping sequence. By definition, there is a $(Y \cup X_1 \cup \cdots \cup X_{\rho-1})$-neglecting weakly $M', Q \setminus \{p'\}, \omega$-enabled $(X_\rho \cup \cdots \cup X_e)$-pumping sequence δ' of length at most $\lambda(\rho)$. If the buffer p' belonged to $Y \cup X_1 \cup \cdots \cup X_{\rho-1}$, we could have combined δ' with $\delta\sigma^{1'}_\rho$ using Proposition 6.23 to conclude the required result. However, if $p' \in Q \setminus \bigcup_{j=1}^{\rho-1} X_j$, then at the end of $\delta\sigma^{1'}_\rho$, p' will only have $W\lambda_1(K)$ tokens. If the sequence δ' is appended to this, then the number of tokens in p' may become negative in some intermediate vector since δ' is too long. To overcome this, we will prove that δ' can be replaced by another sequence where the number of transitions capable of decreasing tokens from p' (i.e., transitions in $T_{ben}(p')$) is at most $\lambda_1(K)$.

Let δ' decompose as $\delta'_\rho\underline{\delta_\rho} \cdots \delta'_e\underline{\delta_e}$ according to Definition 6.20. For every j between ρ and e, use exchange lemma repeatedly to rearrange δ'_j into $\eta'_j\pi'_j$ and $\underline{\delta_j}$ into $\underline{\eta_j\pi_j}$, where $\eta'_j, \underline{\eta_j} \in T_{ind}(p')^*$ and $\pi'_j, \underline{\pi_j} \in T_{ben}(p')^*$. The resulting sequence $\eta'_\rho\pi'_\rho\underline{\eta_\rho\pi_\rho} \cdots \eta'_e\pi'_e\underline{\eta_e\pi_e}$ is still a $(Y \cup X_1 \cup \cdots \cup X_{\rho-1})$-neglecting weakly $M', Q \setminus \{p'\}, \omega$-enabled $(X_\rho \cup \cdots \cup X_e)$-pumping sequence: conditions 1,2,3 and 5 of Definition 6.20 are about the total effect of pumping portions, which have not changed due to the change in positions of transitions. Condition 4 is trivial since in this case, $c = \omega$.

For each j between ρ and e, we define subset of buffers Y^1_j and Y^2_j as follows.

- $Y^1_\rho = \{p \in B \mid \Delta[\underline{\eta_\rho}](p) > 0\} \setminus \bigcup_{l=1}^{\rho-1} X_l$. Intuitively, Y^1_ρ consists of all those buffers whose tokens are pumped up by $\underline{\eta_\rho}$ except those buffers whose tokens have already been pumped up earlier in δ. Similarly, $Y^2_\rho = \{p \in B \mid \Delta[\underline{\pi_\rho}](p) > 0\} \setminus \bigcup_{l=1}^{\rho-1} X_l \setminus Y^1_\rho$.

- Suppose Y_l^1 and Y_l^2 have already been defined for all l between ρ and j. $Y_{j+1}^1 = \{p \in B \mid \Delta[\underline{\eta_{j+1}}](p) > 0\} \setminus \bigcup_{l=1}^{\rho-1} X_l \setminus \bigcup_{l=\rho}^{j}(Y_l^1 \cup Y_l^2)$. Similarly, $Y_{j+1}^2 = \{p \in B \mid \Delta[\underline{\pi_{j+1}}](p) > 0\} \setminus \bigcup_{l=1}^{\rho-1} X_l \setminus \bigcup_{l=\rho}^{j}(Y_l^1 \cup Y_l^2) \setminus Y_{j+1}^1$.

Again use the exchange lemma repeatedly to get the sequence $\eta'_\rho\underline{\eta_\rho} \cdots \eta'_e\underline{\eta_e}\pi'_\rho\underline{\pi_\rho} \cdots \pi'_e\underline{\pi_e}$. We claim that $\delta\sigma_\rho^{1\prime}\eta'_\rho\underline{\eta_\rho} \cdots \eta'_e\underline{\eta_e}\pi'_\rho\underline{\pi_\rho} \cdots \pi'_e\underline{\pi_e}$ is a Y-neglecting weakly $M, Q \setminus \{p'\}, \omega$-enabled X-pumping sequence. The sequence δ is already a Y-neglecting weakly $M, Q \setminus \{p'\}, \omega$-enabled $(X_1 \cup \cdots \cup X_{\rho-1})$-pumping sequence, such that $M \xrightarrow{\delta\sigma_\rho^{1\prime}} M''$ with $M''(p) = M'(p)$ for all $p \in (Q \setminus (Y \cup \bigcup_{j=1}^{\rho-1} X_j))$. Hence by Proposition 6.23, it is sufficient to prove that $\eta'_\rho\underline{\eta_\rho} \cdots \eta'_e\underline{\eta_e}\pi'_\rho\underline{\pi_\rho} \cdots \pi'_e\underline{\pi_e}$ is a $(Y \cup X_1 \cup \ldots X_{\rho-1})$-neglecting weakly $M', Q \setminus \{p'\}, \omega$-enabled $X \setminus (X_1 \cup \ldots X_{\rho-1})$-pumping sequence. By definition, $X \setminus (X_1 \cup \ldots X_{\rho-1}) \subseteq (\bigcup_{j=\rho}^{e} Y_j^1 \cup \bigcup_{j=\rho}^{e} Y_j^2)$. Therefore, it is sufficient to prove that $\eta'_\rho\underline{\eta_\rho} \cdots \eta'_e\underline{\eta_e}\pi'_\rho\underline{\pi_\rho} \cdots \pi'_e\underline{\pi_e}$ is a $(Y \cup X_1 \cup \ldots X_{\rho-1})$-neglecting weakly $M', Q \setminus \{p'\}, \omega$-enabled $(\bigcup_{j=\rho}^{e} Y_j^1 \cup \bigcup_{j=\rho}^{e} Y_j^2)$-pumping sequence. We will prove that each condition of Definition 6.20 is satisfied.

1. This condition is satisfied since by definition, $(\bigcup_{j=\rho}^{e} Y_j^1 \cup \bigcup_{j=\rho}^{e} Y_j^2) \subseteq \{p \in B \mid \Delta[\underline{\eta_j}](p) > 0, \rho \leq j \leq e\} \cup \{p \in B \mid \Delta[\underline{\pi_j}](p) > 0, \rho \leq j \leq e\}$.

2. We will prove that for each $p \in C$, $\Delta[\underline{\eta_j}](p) = \Delta[\underline{\pi_j}](p) = 0$ for each j between ρ and e. Neither $\Delta[\underline{\eta_j}](p)$ nor $\Delta[\underline{\pi_j}](p)$ can be greater than 0 since if it were so, $\underline{\eta_j}$ or $\underline{\pi_j}$ can be repeated arbitrarily many times to increase the number of tokens in p beyond 1, violating our assumption that for any initial marking, p remains 1-bounded. Suppose $\Delta[\underline{\eta_j}](p) < 0$. In order to maintain $\Delta[\underline{\eta_j\pi_j}](p) = 0$, we should have $\Delta[\underline{\pi_\rho}](p) > 0$. This is not possible as already shown above. We argue similarly if $\Delta[\underline{\pi_j}](p) < 0$.

3. Suppose for some $p \in B$, $\Delta[\underline{\eta_j}](p) < 0$. Since $\underline{\eta_j}$ is in $T_{ind}(p')^*$ that can not decrease tokens in $Ben(p')$, p has to be in $Ind(p')$. Since $\underline{\pi_j}$ is in $T_{ben}(p')^*$ that can not increase tokens in $Ind(p')$, we have $\Delta[\underline{\eta_j\pi_j}](p) < 0$. Since $\eta'_\rho\pi'_\rho\underline{\eta_\rho\pi_\rho} \cdots \eta'_e\pi'_e\underline{\eta_e\pi_e}$ is a $(Y \cup X_1 \cup \cdots \cup X_{\rho-1})$-neglecting weakly M', Q, ω-enabled $(X_\rho \cup \cdots \cup X_e)$-pumping sequence, either $p \in (Y \cup X_1 \cup \cdots \cup X_{\rho-1})$ or there is some $j' \leq j-1$ such that $\Delta[\underline{\eta_{j'}\pi_{j'}}](p) > 0$. In the latter case, $\Delta[\underline{\eta_{j'}}](p) > 0$ since $\underline{\pi_{j'}} \in T_{ben}(p')^*$ can not increase tokens in $Ind(p')$. Hence, either $p \in (Y \cup X_1 \cup \cdots \cup X_{\rho-1})$ or there is a $j' \leq j-1$ such that $\Delta[\underline{\eta_{j'}}](p) > 0$.

   Suppose for some place $p \in B$, $\Delta[\underline{\pi_j}](p) < 0$. If $\Delta[\underline{\eta_j\pi_j}](p) \geq 0$, then $\Delta[\underline{\eta_j}](p) > 0$ and we are done since $\underline{\eta_j}$ occurs before $\underline{\pi_j}$. If $\Delta[\underline{\eta_j\pi_j}](p) < 0$, since $\eta'_\rho\pi'_\rho\underline{\eta_\rho\pi_\rho} \cdots \eta'_e\pi'_e\underline{\eta_e\pi_e}$ is a $(Y \cup X_1 \cup \cdots \cup X_{\rho-1})$-neglecting weakly M', Q, ω-enabled $(X_\rho \cup \cdots \cup X_e)$-pumping sequence, either $p \in (Y \cup X_1 \cup \cdots \cup X_{\rho-1})$ or there is a $j' \leq j-1$ such that $\Delta[\underline{\eta_{j'}\pi_{j'}}](p) > 0$. Hence, either $p \in (Y \cup X_1 \cup \cdots \cup X_{\rho-1} \cup Y_j^1)$ or there is a $j' \leq j-1$ such that $\Delta[\underline{\eta_{j'}}](p) > 0$ or $\Delta[\underline{\pi_{j'}}](p) > 0$.

4. This is trivial since in this case, $c = \omega$.

5. Suppose for some intermediate marking M''' and some place $p \in Q \setminus \{p'\}$, $M'''(p) < 0$. If $p \in Y \cup X_1 \cup \cdots \cup X_\rho$, then there is nothing else to prove. Otherwise, we have $M'(p) \geq 0$. Suppose M''' occurs when walking $\eta'_\rho\underline{\eta_\rho} \cdots \eta'_e\underline{\eta_e}$ from M'. Since $\eta'_\rho\underline{\eta_\rho} \cdots \eta'_e\underline{\eta_e} \in T_{ind}(p')^*$ can not decrease tokens from $Ben(p')$ and $M'(p) \geq 0$, $p \in Ind(p')$. Suppose $\underline{\eta_j}$ is the last pumping portion occurring before M''' and η is the prefix of $\eta'_{j+1}\underline{\eta_{j+1}}$ that occurs before M''' so that $M' \xrightarrow{\eta'_\rho\underline{\eta_\rho} \cdots \eta'_j\underline{\eta_j}\eta} M'''$. Let $M' \xrightarrow{\eta'_\rho\pi'_\rho\underline{\eta_\rho\pi_\rho} \cdots \eta'_j\pi'_j\underline{\eta_j\pi_j}\eta} M'''_j$. If all intermediate vectors between M' and M'''_j have non-negative number of tokens in p, then by exchange lemma, $M'''(p) \geq 0$, a contradiction. Hence, there is some

intermediate marking between M' and M'''_j that has negative number of tokens in p. Since $\eta'_\rho\pi'_\rho\underline{\eta_\rho\pi_\rho}\cdots\eta'_e\pi'_e\underline{\eta_e\pi_e}$ is a $Y \cup X_1 \cup \cdots X_{\rho-1}$-neglecting weakly $M', Q \setminus \{p'\}, \omega$-enabled $(\bigcup_{j=\rho}^{e} Y_j^1 \cup \bigcup_{j=\rho}^{e} Y_j^2)$-pumping sequence, there is some $\rho \leq j' \leq j$ such that $\Delta[\underline{\eta_{j'}\pi_{j'}}](p) > 0$. Since $p \in Ind(p')$ and $\pi_{j'} \in T_{ben}(p')^*$ can not increase tokens in p, we get $\Delta[\underline{\eta_{j'}}](p) > 0$.

On the other hand, suppose M''' occurs after walking $\eta'_\rho\underline{\eta_\rho}\cdots\eta'_e\underline{\eta_e}$ from M' (i.e., when walking $\pi'_\rho\underline{\pi_\rho}\cdots\pi'_e\underline{\pi_e}$). First suppose that $p \in Ind(p')$. If all intermediate vectors occurring when walking $\eta'_\rho\pi'_\rho\underline{\eta_\rho\pi_\rho}\cdots\eta'_e\pi'_e\underline{\eta_e\pi_e}$ from M' have non-negative number of tokens in p, then by exchange lemma, $M'''(p) \geq 0$, a contradiction. Hence, there is some intermediate marking that has negative number of tokens in p. Since $\eta'_\rho\pi'_\rho\underline{\eta_\rho\pi_\rho}\cdots\eta'_e\pi'_e\underline{\eta_e\pi_e}$ is a $Y \cup X_1 \cup \cdots X_{\rho-1}$-neglecting weakly $M', Q \setminus \{p'\}, \omega$-enabled $(\bigcup_{j=\rho}^{e} Y_j^1 \cup \bigcup_{j=\rho}^{e} Y_j^2)$-pumping sequence, there is some $\rho \leq j \leq e$ such that $\Delta[\underline{\eta_j\pi_j}](p) > 0$. Since $p \in Ind(p')$ and $\pi_j \in T_{ben}(p')^*$ can not increase tokens in p, we get $\Delta[\underline{\eta_j}](p) > 0$. Next suppose that $p \in Ben(p')$. Let $\underline{\pi_j}$ be the last pumping portion occurring before M''' and π be the prefix of $\pi'_{j+1}\underline{\pi_{j+1}}$ occurring before M''' so that $M' \xrightarrow{\eta'_\rho\underline{\eta_\rho}\cdots\eta'_e\underline{\eta_e}\pi'_\rho\underline{\pi_\rho}\cdots\pi'_j\underline{\pi_j}\pi} M'''$. Since $M'(p) + \Delta[\eta'_\rho\underline{\eta_\rho}\cdots\eta'_e\underline{\eta_e}\pi'_\rho\underline{\pi_\rho}\cdots\pi'_j\underline{\pi_j}\pi](p) < 0$ and $\underline{\eta_{j+1}}\eta'_{j+2}\underline{\eta_{j+2}}\cdots\eta'_e\underline{\eta_e} \in T_{Ind}(p')^*$ can not decrease tokens from $p \in Ben(p')$, $M'(p) + \Delta[\eta'_\rho\pi'_\rho\underline{\eta_\rho\pi_\rho}\cdots\underline{\eta_j\pi_j}\eta'_{j+1}\pi](p) < 0$. Therefore, there is an intermediate vector with negative number of tokens in p that occurs while walking $\eta'_\rho\pi'_\rho\underline{\eta_\rho\pi_\rho}\cdots\underline{\eta_j\pi_j}\eta'_{j+1}\pi$ from M'. Since $\eta'_\rho\pi'_\rho\underline{\eta_\rho\pi_\rho}\cdots\eta'_e\pi'_e\underline{\eta_e\pi_e}$ is a $Y \cup X_1 \cup \cdots X_{\rho-1}$-neglecting weakly $M', Q \setminus \{p'\}, \omega$-enabled $(\bigcup_{j=\rho}^{e} Y_j^1 \cup \bigcup_{j=\rho}^{e} Y_j^2)$-pumping sequence, there is some $\rho \leq j' \leq j$ such that $\Delta[\underline{\eta_{j'}\pi_{j'}}](p) > 0$. Therefore, either $\Delta[\underline{\eta_{j'}}](p) > 0$ or $\Delta[\underline{\pi_{j'}}](p) > 0$

If for some $p \in Ind(p')$, $\Delta[\underline{\pi_j}](p) < 0$, then either $p \in (X_1 \cup \cdots \cup X_{\rho-1})$ or there is a $j' \leq j-1$ such that $\Delta[\underline{\eta_{j'}}](p) > 0$ (it cannot be the case that there is some $j' \leq j$ with $\Delta[\underline{\pi_{j'}}](p) > 0$ since $p \in Ind(p')$ and $\underline{\pi_{j'}} \in T_{ben}(p')^*$ cannot increase tokens in $Ind(p')$). Hence, $\pi'_\rho\underline{\pi_\rho}\cdots\pi'_e\underline{\pi_e}$ is a $(Ind(p') \cup \bigcup_{j=1}^{\rho-1} X_j \cup \bigcup_{j=\rho}^{e} Y_j^1)$-neglecting weakly $M^3, Q \setminus \{p'\}, \omega$-enabled $(Y_\rho^2 \cup \cdots \cup Y_e^2)$-pumping sequence in $T_{ben}(p')^*$, where $M \xrightarrow{\delta\sigma_\rho^{1'}\eta'_\rho\underline{\eta_\rho}\cdots\eta'_e\underline{\eta_e}} M^3$. By definition, there is a $(Ind(p') \cup \bigcup_{j=1}^{\rho-1} X_j \cup \bigcup_{j=\rho}^{e} Y_j^1)$-neglecting weakly $M^3, Q \setminus \{p'\}, \omega$-enabled $(Y_\rho^2 \cup \cdots \cup Y_e^2)$-pumping sequence π' of length at most $\lambda_1(K)$ (since at most K buffers benefit from p').

As mentioned earlier, if $p' \in (Y \cup X_1 \cup \cdots X_{\rho-1})$, then we can conclude the required result by observing that $\delta\sigma_\rho^{1'}\delta'$ is a $\emptyset$-neglecting weakly M, Q, ω-enabled X-pumping sequence of the required length. Now, if $p' \in (Q \setminus (Y \cup \bigcup_{j=1}^{\rho-1} X_j))$ and $M \xrightarrow{\delta\sigma_\rho^{1'}} M''$, then $M''(p') \geq W\lambda_1(K)$. Since $\eta'_\rho\underline{\eta_\rho}\cdots\eta'_e\underline{\eta_e} \in T_{ind}(p')^*$ can not decrease tokens in p' and π' can decrease at most $W\lambda_1(K)$ tokens from p', $\delta\sigma_\rho^{1'}\eta'_\rho\underline{\eta_\rho}\cdots\eta'_e\underline{\eta_e}\pi'$ is a $\emptyset$-neglecting weakly M, Q, ω-enabled X-pumping sequence. The length of δ is at most $(\rho-1)(W^2\lambda(K)2^a)^{poly(m)}$. The length of $\sigma_\rho^{1'}$ is at most $(W\lambda_1(K))^{|Q|}$. The length of $\eta'_\rho\underline{\eta_\rho}\cdots\eta'_e\underline{\eta_e}$ is at most $\lambda(i)$. The length of π' is at most $\lambda_1(K)$. The result follows.

Case 3: Suppose that according to Definition 6.20, σ decomposes as

$$M \xrightarrow{\sigma'_1} M'_1 \xrightarrow{\underline{\sigma_1}} M_1 \longrightarrow \cdots \xrightarrow{\sigma'_e} M'_e \xrightarrow{\underline{\sigma_e}} M_e$$

and there is some $\rho \in \{1, \ldots, e\}$ and an intermediate vector M' between M'_ρ and M_ρ such that for some buffer $p' \in Q \cap B$, $M'(p') \geq W\lambda_1(K)$. Consider the sequence $\sigma'_1\underline{\sigma_1}\cdots\sigma'_\rho\sigma_\rho\underline{\sigma_\rho}\cdots\sigma'_e\underline{\sigma_e}$ obtained by repeating σ_ρ and treating only its second occurrence as pumping portion. By Lemma 6.24, $\sigma'_1\underline{\sigma_1}\cdots\sigma'_\rho\sigma_\rho\underline{\sigma_\rho}\cdots\sigma'_e\underline{\sigma_e}$ is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence. Now, we are back in case 2. □

Lemma 6.27. *$\lambda(0) \leq m(W2^a)^{poly(m)}$ and for $1 \leq i \leq b$, $\lambda(i) \leq im(W^2\lambda_1(K)2^a)^{poly(m)} + i\lambda_1(K) + m(W2^a)^{poly(m)}$.*

Proof. For $\lambda(0)$, we do not care if any buffer has negative number of tokens, so the problem is an integer linear programming one where we have to simply check the existence of certain combination of transitions satisfying some conditions. The result stated in the lemma can be obtained by substituting $c = 1$ in Lemma 6.25. The second result of the lemma is proved by induction on i. The base case $i = 1$ is a direct consequence of Lemma 6.26 and the bound on $\lambda(0)$.

Induction step: By Lemma 6.26,

$$\begin{aligned}\lambda(i+1) &\leq m(W^2\lambda_1(K)2^a)^{poly(m)} + \lambda(i) + \lambda_1(K)\\ &\leq m(W^2\lambda_1(K)2^a)^{poly(m)} + im(W^2\lambda_1(K)2^a)^{poly(m)}\\ &\quad + i\lambda_1(K) + m(W2^a)^{poly(m)} + \lambda_1(K)\\ &= (i+1)m(W^2\lambda_1(K)2^a)^{poly(m)} + (i+1)\lambda_1(K)\\ &\quad + m(W2^a)^{poly(m)}\end{aligned}$$

□

Next, we will give a recurrence relation for bounding lengths of pumping sequences consisting of transitions in $T_{ben}(p)$ for some place p.

Lemma 6.28. *Let $X, Y \subseteq B$ be subset of buffers with X being non-empty. Let $C \subseteq Q \subseteq P$ be a subset of places, M a vector, $c \in \mathbb{N}$ a number and $p \in B$ a buffer. Suppose there is a $(Ind(p) \cup Y)$-neglecting weakly M, Q, c-enabled X-pumping sequence $\sigma \in T_{ben}(p)^*$ that decomposes as $\sigma'_1\underline{\sigma_1}\cdots\sigma'_e\underline{\sigma_e}$ as in Definition 6.20. Then, there is a $(Ind(p) \cup Y)$-neglecting weakly M, Q, ω-enabled X-pumping sequence in $T_{ben}(p)^*$ of length at most $e(Wc2^a)^{poly(K)}$, where poly() is a polynomial whose degree does not depend on m, K, W, e, a or c.*

Proof. We will prove the result by induction on e. If $e = 1$, we have just the sequence $M \xrightarrow{\sigma'_1} M'_1 \xrightarrow{\sigma_1} M_1$. If any two intermediate vectors between M and M'_1 is same with respect to $Q \setminus Y \setminus Ind(p)$, the subsequence between them can be removed without violating the remaining sequence's property. Repeating this as many times as needed, we can shorten σ'_1 to be of length at most $(c2^a)^K$. Let a $(Q \setminus Y \setminus Ind(p))$-loop be any subsequence of $\underline{\sigma_1}$ whose effect on all places in $(Q \setminus Y \setminus Ind(p))$ is 0 such that no subsequence inside this loop satisfies the same property. As is done in [87, Lemma 2.2], $(Q \setminus Y \setminus Ind(p))$-loops can be carefully removed from $\underline{\sigma_1}$ without affecting its X-pumping property such that the total length of shortened σ'_1 and $\underline{\sigma_1}$ is at most $(Wc2^a)^{poly(m)}$. Since the sequence we are looking neglects all places in $Ind(p)$, the inequation system to be solved while carefully removing loops above can neglect rows corresponding to $Ind(p)$. Hence there are at most K rows and hence we can in fact get a sequence of length at most $(Wc2^a)^{poly(K)}$.

For the induction step, suppose σ decomposes as follows.

$$M \xrightarrow{\sigma'_1} M'_1 \xrightarrow{\underline{\sigma_1}} M_1 \xrightarrow{\sigma'_2} M'_2 \xrightarrow{\underline{\sigma_2}} M_2 \longrightarrow \ldots \xrightarrow{\sigma'_{e+1}} M'_{e+1} \xrightarrow{\underline{\sigma_{e+1}}} M_{e+1}$$

For $1 \leq \rho \leq e+1$, let $X_\rho = \{p' \in B \mid \Delta[\underline{\sigma_\rho}](p') > 0\}$. We observe that $\sigma'_2\underline{\sigma_2}\cdots\sigma'_{e+1}\underline{\sigma_{e+1}}$ is a $(Y \cup Ind(p) \cup X_1)$-neglecting weakly M_1, Q, c-enabled $(X_2 \cup \cdots \cup X_{e+1})$-pumping sequence in $T_{ben}(p)^*$. By induction hypothesis, there is a $(Y \cup Ind(p) \cup X_1)$-neglecting weakly M_1, Q, ω-enabled $(X_2 \cup \cdots \cup X_{e+1})$-pumping sequence $\delta \in T_{ben}(p)^*$ of length at most $e(Wc2^a)^{poly(K)}$. As is done in the base case, σ'_1 and $\underline{\sigma_1}$ can be replaced by shorter δ'_1 and $\underline{\delta_1}$ respectively such that $\delta'_1\underline{\delta_1}$ is a $(Y \cup Ind(p))$-neglecting weakly M, Q, c-enabled X_1-pumping sequence in

$T_{ben}(p)^*$ of length at most $(Wc2^a)^{poly(K)}$. Since the change from $\sigma'_1\underline{\sigma_1}$ to $\delta'_1\underline{\delta_1}$ only involves removal of $(Q \setminus Y \setminus Ind(p))$-loops, we have $M \xrightarrow{\delta'_1\underline{\delta_1}} M^2$ such that for all $p \in (Q \setminus Y \setminus Ind(p))$, $M^2(p) = M_1(p)$. By Proposition 6.23, $\delta'_1\underline{\delta_1}\delta$ is a $(Y \cup Ind(p))$-neglecting weakly M, Q, ω-enabled X-pumping sequence in $T_{ben}(p)^*$. Its length is at most $(e+1)(Wc2^a)^{poly(K)}$. □

Lemma 6.29. $\lambda_1(0) \leq m(W2^a)^{poly(K)}$.

Proof. By Lemma 6.28, putting $c = 1$. □

Lemma 6.30. *For $0 \leq i < K$, $\lambda_1(i+1) \leq m(W^2\lambda_1(i)2^a)^{poly(K)} + \lambda_1(i)$, where poly() is a polynomial whose degree is a constant independent of m, K, W or a.*

Proof. Let $C \subseteq Q \subseteq P$ be a subset of places such that $|Q \cap Ben(p) \cap B| = i+1$ for some buffer p. Suppose $\sigma \in T_{ben}(p)^*$ is a $(Ind(p) \cup Y)$-neglecting weakly M, Q, ω-enabled X-pumping sequence for some vector M and some subsets $X, Y \subseteq B$.

Case 1: σ is a $(Ind(p) \cup Y)$-neglecting weakly $M, Q, W\lambda_1(i)$-enabled X-pumping sequence. By Lemma 6.28, there is a $(Ind(p) \cup Y)$-neglecting weakly $M, Q, W\lambda_1(i)$-enabled X-pumping sequence of length at most $m(W^2\lambda_1(i)2^a)^{poly(K)}$.

Case 2: Suppose that according to Definition 6.20, σ decomposes as

$$M \xrightarrow{\sigma'_1} M'_1 \xrightarrow{\underline{\sigma_1}} M_1 \longrightarrow \cdots \xrightarrow{\sigma'_e} M'_e \xrightarrow{\underline{\sigma_e}} M_e$$

and there is some $\rho \in \{2, \ldots, e\}$ and an intermediate vector M' between $M_{\rho-1}$ and strictly before $M_{\rho'}$ (or between M and strictly before M'_1) such that for some buffer $p' \in Q \cap Ben(p) \cap B$, $M'(p') \geq W\lambda_1(i)$. Let M' be the first such marking. For $1 \leq \rho \leq e$, let $X_\rho = \{p'' \in B \mid \Delta[\underline{\sigma_\rho}](p'') > 0\}$.

Suppose $M_{\rho-1} \xrightarrow{\sigma^{1\prime}_\rho} M' \xrightarrow{\sigma^{2\prime}_\rho} M'_\rho$. The sequence $\sigma'_1\underline{\sigma_1} \cdots \sigma'_{\rho-1}\underline{\sigma_{\rho-1}}$ is a $(Ind(p) \cup Y)$-neglecting weakly $M, Q, W\lambda_1(i)$-enabled $(X_1 \cup \cdots X_{\rho-1})$-pumping sequence. By Lemma 6.28, there is a $(Ind(p) \cup Y)$-neglecting weakly M, Q, ω-enabled $(X_1 \cup \cdots X_{\rho-1})$-pumping sequence δ of length at most $(\rho-1)(W^2\lambda_1(i, X)2^a)^{poly(K)}$ such that $M \xrightarrow{\delta} M''_{\rho-1} \xrightarrow{\sigma^{1\prime}_\rho} M''$ where for all $p'' \in (Q \setminus \bigcup_{j=1}^{\rho-1} X_j \setminus Ind(p) \setminus Y)$, $M''(p'') = M'(p'')$.

The sequence $\sigma^{2\prime}_\rho\underline{\sigma_\rho}\sigma'_{\rho+1} \cdots \underline{\sigma_e}$ is a $(Ind(p) \cup Y \cup X_1 \cup \cdots \cup X_{\rho-1})$-neglecting weakly M', Q, ω-enabled $(X_\rho \cup \cdots \cup X_e)$-pumping sequence. By definition, there is a $(Ind(p) \cup Y \cup X_1 \cup \cdots \cup X_{\rho-1})$-neglecting weakly $M', Q \setminus \{p'\}, \omega$-enabled $(X_\rho \cup \cdots \cup X_e)$-pumping sequence δ' of length at most $\lambda_1(\rho)$. By Proposition 6.23, $\delta\sigma^{1\prime}_\rho\delta'$ is a $Ind(p) \cup Y$-neglecting weakly $M, Q \setminus \{p'\}, \omega$-enabled X-pumping sequence in $T_{ben}(p)^*$ of length at most $m(W^2\lambda_1(i)2^a)^{poly(K)} + \lambda_1(i)$. If the buffer p' belonged to $Y \cup X_1 \cup \cdots \cup X_{\rho-1}$, $\delta\sigma^{1\prime}_\rho\delta'$ is also a $Ind(p) \cup Y$-neglecting weakly M, Q, ω-enabled X-pumping sequence in $T_{ben}(p)^*$. Otherwise, $M''(p') \geq W\lambda_1(i)$ and δ' will remove at most $W\lambda_1(i)$ tokens from p'. Hence, again $\delta\sigma^{1\prime}_\rho\delta'$ is a $(Ind(p) \cup Y)$-neglecting weakly M, Q, ω-enabled X-pumping sequence in $T_{ben}(p)^*$.

Case 3: Suppose that according to Definition 6.20, σ decomposes as

$$M \xrightarrow{\sigma'_1} M'_1 \xrightarrow{\underline{\sigma_1}} M_1 \longrightarrow \cdots \xrightarrow{\sigma'_e} M'_e \xrightarrow{\underline{\sigma_e}} M_e$$

and there is some $\rho \in \{1, \ldots, e\}$ and an intermediate vector M' between M'_ρ and M_ρ such that for some buffer $p' \in Q \cap Ben(p) \cap B$, $M'(p') \geq W\lambda_1(\rho)$. Consider the sequence $\sigma'_1\underline{\sigma_1} \cdots \sigma'_\rho\sigma_\rho\underline{\sigma_\rho} \cdots \sigma'_e\underline{\sigma_e}$ obtained by repeating σ_ρ and treating only its second occurrence as pumping portion. It is routine to verify that $\sigma'_1\underline{\sigma_1} \cdots \sigma'_\rho\sigma_\rho\underline{\sigma_\rho} \cdots \sigma'_e\underline{\sigma_e}$ is a $(Ind(p) \cup Y)$-neglecting weakly M, Q, ω-enabled X-pumping sequence in $T_{ben}(p)^*$. Now, we are back in case 2. □

Lemma 6.31. *For $0 \leq i \leq K$, $\lambda_1(i) \leq 2m(2mW^2 2^a)^{(i+1)poly(K)^{i+1}}$.*

Proof. By induction on i. The base case $i = 0$ is clear from Lemma 6.29.

Induction step: by Lemma 6.30

$$\begin{aligned}
\lambda_1(i+1) &\leq m(W^2\lambda_1(i)2^a)^{poly(K)} + \lambda_1(i) \\
&\leq 2m(W^2\lambda_1(i)2^a)^{poly(K)} \\
&\leq 2m(W^2 2m(2mW^2 2^a)^{(i+1)poly(K)^{i+1}} 2^a)^{poly(K)} \\
&\leq 2m(2mW^2 2^a)^{poly(K)}(2mW^2 2^a)^{(i+1)poly(K)^{i+2}} \\
&\leq 2m(2mW^2 2^a)^{(i+2)poly(K)^{i+2}}
\end{aligned}$$

□

Theorem 6.32. *With the benefit depth K as parameter, model checking β formulas is in* ParaPspace.

Proof. As mentioned earlier, disjunctions are resolved by non-deterministically choosing a disjunct so that we are left with checking a conjunction of atomic β formulas. By Lemma 6.18, it is enough to guess a subset X of buffers and check if there is a X-pumping sequence. By Lemma 6.21, it is sufficient and necessary to verify the existence of a $\emptyset$-enabled weakly M_0, P, ω-enabled X-pumping sequence. If there is such a sequence, there is one of length at most $\lambda(b)$ (where b is the number of buffers), by Definition 6.22. Let $|M_0|$ be the maximum of the range of the initial marking M_0. A non-deterministic Turing machine can check the existence of such a sequence by guessing and verifying a sequence of length at most $\lambda(b)$. By combining the results of Lemma 6.27 and Lemma 6.31 and using m as an upper bound for b, we conclude that the memory needed by such a machine is $\mathcal{O}(m \log |M_0| + \log m + \log W + (poly(m) + poly(K)^{K+1})(\log m + \log W + a))$. An application of Savitch's theorem then gives us the required ParaPspace algorithm. □

Chapter 7

Graph Structure and Vertex Cover for Petri Nets

The EXPSPACE-hardness proof in [66] uses Petri nets that simulate deeply nested loops that manipulate counters with high values. The minimum number of nodes needed to remove all loops (*called feedback vertex set number*) is high for this structure. As a step towards studying parameterized complexity with respect to feedback vertex set number as a parameter, we obtain a weaker result in this chapter, by considering a stronger parameter called *vertex cover number*. Vertex cover number is the minimum number of nodes needed to remove all edges. We will show PARAPSPACE algorithms for coverability, boundedness and model checking the logic of counting properties defined in section 6.3.

The main idea behind these results is that if a Petri net has a small vertex cover, rest of the net can be grouped into a small number of parts, each consisting of places sharing similar properties. We can then reason about each part as a whole instead of reasoning about individual places. This kind of reasoning can again be used for a finer analysis of Rackoff's recurrence relations [83]. Similar strategy has been used to design efficient algorithms for graph problems that are hard with treewidth as parameter, or whose parameterized complexity with respect to treewidth is not known [33].

One-counter automata are closely related to Petri nets. Precise complexity of reachability and many other problems of this model have been recently obtained in [44, 40]. We have adapted some of the techniques used in [44, 40], in particular the use of [61, Lemma 42].

7.1 Vertex Cover for Petri Nets

As is done in Chapter 4, we consider the flow graph $G(\mathcal{N})$ of a Petri net $\mathcal{N}$ and look at the smallest vertex cover. Figure 7.1 shows an example of a Petri net with vertex cover $\{p_1, \ldots, p_4\}$.

Suppose VC is a vertex cover for some graph G. If $v_1, v_2 \notin VC$ are two vertices not in VC that have the same set of neighbours (neighbours of a vertex v are vertices that have an edge connecting them to v), v_1 and v_2 have similar properties. This fact is used to obtain FPT algorithms for many hard problems, e.g., see [33]. The same phenomenon leads to PARAPSPACE algorithms for Petri net coverability and boundedness. In the rest of this section, we will define the formalisms needed to prove these results.

Definition 7.1. *Let the places of a Petri net $\mathcal{N}$ be $p_1, p_2, \ldots, p_m$. Suppose there is a vertex cover VC consisting of places $p_1, \ldots, p_k$. We say that two transitions t_1 and t_2 have the same VC-neighbourhood if $Pre(p_i, t_1) = Pre(p_i, t_2)$ and $Post(p_i, t_1) = Post(p_i, t_2)$ for all i between 1 and k.*

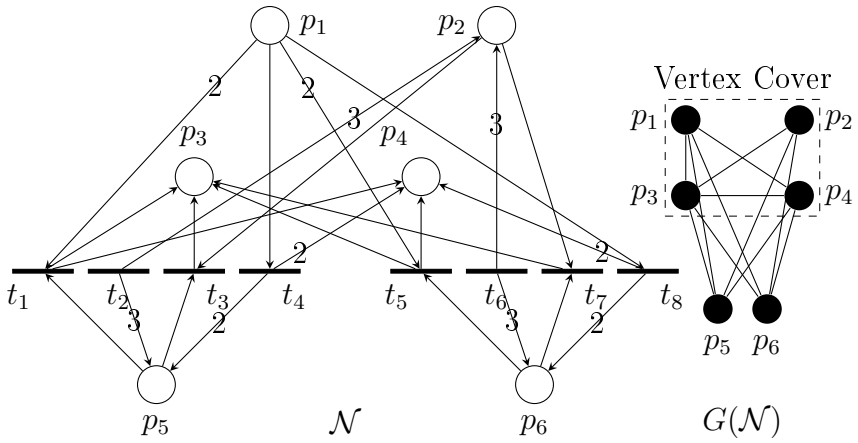


Figure 7.1: A Petri net with vertex cover $\{p_1, \ldots, p_4\}$

In Fig. 7.1, transitions t_1 and t_5 have the same VC-neighbourhood. Intuitively, two transitions with the same VC-neighbourhood behave similarly as far as places in the vertex cover are concerned. Since there can be $2k$ arcs between a transition and places in VC and each arc can have weight between 0 and W, there can be at most $(W+1)^{2k}$ different VC-neighbourhoods of transitions.

Let p be a place not in the vertex cover VC. Suppose there are $l \leq (W+1)^{2k}$ VC-neighbourhoods of transitions. Place p can have one incoming arc from or one outgoing arc to each transition of the net (it cannot have both an incoming and an outgoing arc since in that case, p would have a self loop and would be in VC). If p' is another place not in VC, then no transition can have arcs to both p and p', since otherwise, there would haven been an edge between p and p' in $G(\mathcal{N})$ and one of the places p and p' would have been in VC. Hence, places not in VC cannot interact with each other directly. Places not in VC can only interact with places in VC through transitions and there are at most l VC-neighbourhoods of transitions. Suppose p and p' have the following property: for every transition t that has an arc to/from p with weight w, there is another transition t' of the same VC-neighbourhood as t that has an arc to/from p' with weight w. Then, p and p' interact with VC in the same way in the following sense: whenever a transition involving p fires, an "equivalent" transition can be fired that involves p' instead of p, provided there are enough tokens in p'. In Fig. 7.1, places p_5 and p_6 satisfy the property stated above. Transition t_5 can be fired instead of t_1, t_6 can be fired instead of t_2 etc.

Definition 7.2. *Suppose $\mathcal{N}$ is a Petri net with vertex cover VC and l VC-neighbourhoods of transitions. Let $p \notin VC$ be a place not in the vertex cover. The VC-**interface** $I[p]$ of p is defined as the function $I[p] : \{1, \ldots, l\} \to 2^{\{-W, \ldots, W\} \setminus \{0\}}$, where for every j between 1 and l and every $w \neq 0$ between $-W$ and W, there is a transition t_j of VC-neighbourhood j such that $w = -Pre(p, t_j) + Post(p, t_j)$ iff $w \in I[p](j)$. We denote VC-interfaces of places by I, I' etc.*

In the above definition, since $p \notin VC$, at most one among $Pre(p, t_j)$ and $Post(p, t_j)$ will be non-zero.

The fact that transitions can be exchanged between two places of the same VC-interface can be used to obtain better bounds on the length of firing sequences. For example, suppose a firing sequence σ is fired in the Petri net of Fig. 7.1, with an initial marking that has no tokens in p_5 and p_6. Let c be the maximum number of tokens in any place in any intermediate marking during the firing of σ. Since there are 6 places and each intermediate marking has at most c tokens in every place, the number of possible distinct intermediate markings is

$(c+1)^6$. This is also an upper bound on the length of σ (if two intermediate markings are equal, then the subsequence between those two markings can be removed without affecting the final marking reached). Now, suppose that in the final marking reached, p_5 and p_6 do not have any tokens and we replace all occurrences of t_5, t_6, t_7 and t_8 in σ by t_1, t_2, t_3 and t_4 respectively. After this replacement, the final marking reached will be same as the one reached after firing σ. Number of tokens in p_5 will be at most $2c$ in any intermediate marking and there will be no tokens at all in p_6. Variation in the number of tokens in p_1, p_2, p_3 and p_4 do not change (since as far as these places are concerned, transitions t_5, t_6, t_7 and t_8 behave in the same way as do t_1, t_2, t_3 and t_4 respectively). Hence, in any intermediate marking, each of the places p_1, p_2, p_3 and p_4 will still have at most c tokens. When we exchange the transitions as mentioned above, there might be some intermediate markings that are same, so that we can get a shorter firing sequence achieving the same effect as the original one. These duplicate markings signify the "redundancy" that was present in the original firing sequence σ, but was not apparent due to the distribution of tokens among places. After removing such redundancies, the new upper bound on the length of the firing sequence is $(2c+1).(c+1)^4$, which is asymptotically smaller than the previous bound $(c+1)^6$. A careful observation of the effect of this phenomenon on Rackoff's induction strategy in [83] leads us to the main results of this chapter.

Definition 7.3. *Let p_1 and p_2 be two places of the same VC-interface. Let σ be a firing sequence. A sequence of transitions $\sigma' = t_1 \dots t_r$ is said to be a* sub-word *of σ if there are positions $i_1 < \dots < i_r$ in σ such that for each j between 1 and r, $i_j{}^{th}$ transition of σ is t_j. Suppose σ' is a sub-word of σ made up of transitions that have an arc to/from p_1.* **Transferring σ' from p_1 to p_2** *means replacing every transition t of σ' (which has an arc to/from p_1 with some weight w) with another transition t' of the same VC-neighbourhood as t which has an arc to/from p_2 with weight w. The sub-word σ' is said to be* **safe for transfer** *from p_1 if for every prefix σ'' of σ', the effect of σ'' on p_1 (i.e., the change in the number of tokens in p_1 as a result of firing all transitions in σ'') is greater than or equal to 0.*

Intuitively, if some sub-word σ' is safe for transfer from p_1, it never removes more tokens from p_1 than it has already added to p_1. So if we transfer σ' from p_1 to p_2, the new transitions will always add tokens to p_2 before removing them from p_2, so there is no chance of number of tokens in p_2 becoming negative due to the transfer. However, the number of tokens in p_1 may become negative due to some old transitions remaining back in the "untransferred" portion of the original firing sequence σ. The following lemma says that if some intermediate marking has very high number of tokens in some place, then a suitable sub-word can be safely transfered without affecting the final marking reached or introducing negative number of tokens in any place, but reducing the maximum number of tokens accumulated in any intermediate marking. The proof is a simple consequence of [61, Lemma 42], which is about one-counter automata. An one-counter automaton is an automaton with a counter that can store natural numbers. Apart from changing its state, the automaton can increment the counter, test it for zero and decrement it when not zero. It is proven in [61, Lemma 42] that if a one-counter automaton can reach from one of its configuration to another, it can do so without increasing the intermediate values of the counter by large numbers.

Lemma 7.4 (Truncation lemma, [61])**.** *Let p_1 and p_2 be places of the same VC-interface. Let $c \in \mathbb{N}$ be any number and σ be a firing sequence. Suppose during the firing of σ, there are intermediate markings M_1 and M_3 such that $M_1(p_1) = c$ and $M_3(p_1) \leq c$. Suppose M_2 is an intermediate marking between M_1 and M_3 such that $M_2(p_1) \geq c + W^2 + W^3$ is the maximum number of tokens in p_1 at any intermediate marking between M_1 and M_3. Then, there is a sub-word σ' of σ that is safe for transfer from p_1 to p_2 such that*

1. *The total effect of σ' on p_1 is 0.*

2. *After transferring σ' to p_2, the number of tokens in p_1 at M_2 is strictly less than the number of tokens in p_1 at M_2 before the transfer.*

3. *No intermediate marking will have negative number of tokens in p_1 after the transfer.*

Proof. Let M_1' be the last intermediate marking before M_2 such that $M_1'(p_1) \leq c + W^2$ (see Fig. 7.2). Let M_3' be the first intermediate marking after M_2 such that $M_3'(p_1) \leq c + W^2$.

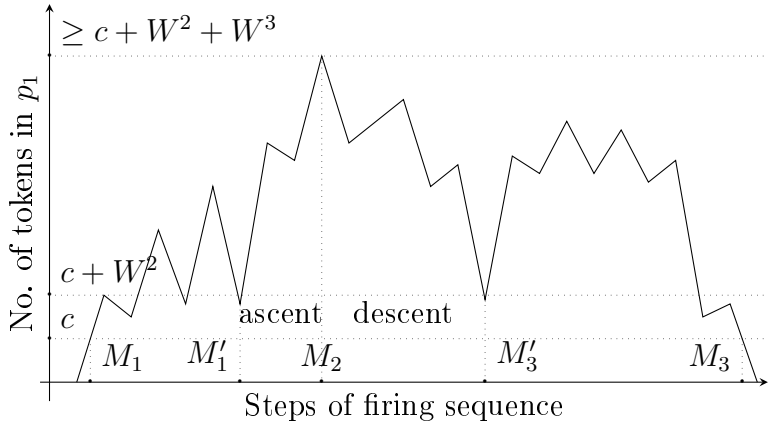


Figure 7.2: Illustration for proof of Lemma 7.4

We will call the subsequence between M_1' and M_2 as *ascent* and the subsequence between M_2 and M_3' as *descent*. During ascent, the number of tokens in p_1 increases by at least W^3. Since each transition can add at most W tokens to p_1, there are at least W^2 transitions adding tokens to p_1 during ascent. There must be at least one number $1 \leq w_1 \leq W$ such that among these W^2 transitions, there are at least W transitions that add exactly w_1 tokens to p_1. Similarly, there is a number $1 \leq w_2 \leq W$ such that at least W transitions remove exactly w_2 tokens from p_1 during descent. The sub-word σ' we need consists of w_2 "adding" transitions from ascent and w_1 "removing" transitions from descent. The total effect of σ' on p_1 is 0 and it is safe for transfer from p_1 to p_2 by construction. Since the first part of σ' removes $w_1 w_2 > 0$ tokens from p_1, the number of tokens $M_2(p_1)$ after transferring σ' to p_2 is strictly less than the number of tokens before the transfer. Before transfer, every intermediate marking between M_1' and M_3' had at least $c + W^2$ tokens. Since the transfer of σ' causes $w_1 w_2 \leq W^2$ fewer tokens, all intermediate markings between M_1' and M_3' will have at least $c \geq 0$ tokens in p_1 after transfer. Intermediate markings before M_1' and after M_3' do not change. □

There can be at most $(2^{2W})^l \leq 2^{2W(W+1)^{2k}}$ VC-interfaces of places that are not in the vertex cover VC, if the number of places in the vertex cover is k. For each VC-interface I, we designate one of the places having I as its VC-interface as special, and use p_I to denote it. We will call $S = VC \cup \{p_I \mid I \text{ is the } VC\text{-interface of a place not in } VC\}$ the set of special places. We will denote the set $P \setminus S$ using I and call the places in I independent places. We will use k' for the cardinality of S and note that $k' \leq k + 2^{2W(W+1)^{2k}}$. If k and W are parameters, then k' is a function of the parameters only. Hence, in the rest of this chapter, we will treat k' as the parameter.

7.2 ParaPspace Algorithm for the Coverability Problem

In this section, we will show that for a Petri net $\mathcal{N}$ with a vertex cover of size k and maximum arc weight W, the coverability problem can be solved in space $\mathcal{O}(ef(k, W)poly(|\mathcal{N}| +$

$\log |M_{cov}|)$). Here, ef is some computable function exponential in k and W while $poly(|\mathcal{N}| + \log |M_{cov}|)$ is some polynomial in the size of the net and the marking to be covered.

Recall vectors and Q-runs from section 5.4. For a transition t and vectors $M, M' : P \to \mathbb{Z}$, we write $M \xrightarrow[Q]{t} M'$ if $M'(p) = M(p) - Pre(p,t) + Post(p,t)$ for all $p \in P$ and $M(q), M'(q) \geq 0$ for all $q \in Q$. The following is a refinement of Definition 5.14 to Petri nets with vertex cover VC.

Definition 7.5. *Let $Q \subseteq P$ be some subset of places such that $I \subseteq Q$. For a vector $M : P \to \mathbb{Z}$, let $lencov(Q, M, M_{cov})$ be the length of the shortest Q-covering run from M. Define $lencov(Q, M, M_{cov})$ to be 0 if there is no such run. Define $\ell(i) = \max\{lencov(Q, M, M_{cov}) \mid I \subseteq Q \subseteq P, |Q \setminus I| = i, M : P \to \mathbb{Z}\}$.*

Let R be the maximum of the range of M_{cov}, the marking to be covered. We will denote $R + W + W^2 + W^3$ by R'. Recall that m is the number of places in the given Petri net. The following lemmas give an upper bound on $\ell(k')$, extending Lemma 5.15.

Lemma 7.6. $\ell(0) \leq mR$.

Proof. $\ell(0)$ is the length of the shortest I-covering run. Recall that all places in I are independent of each other, so if a transition has an arc to one of the places in I, it does not have arcs to any other place in I. Since an I-covering run does not care about places in S, it only has to worry about adding tokens to places in I. If a transition adds a token to some place p in I, it does not remove tokens from any other place in I. Hence, this transition can be repeated R times to add at least R tokens to the place p, which is all that is needed for p. Arguing similarly for other places in I, a total of mR transitions are enough to add all required tokens to all places in I, since there are less than m places in I. □

Lemma 7.7. $\ell(i+1) \leq R'^m(W\ell(i) + R)^{i+1} + \ell(i)$.

Proof. Suppose $I \subseteq Q \subseteq P$ and $|Q \setminus I| = i + 1$. Suppose there is a sequence σ that is Q-covering from some vector M_0. Let p be any place in I of some VC-interface I. Let M be the first intermediate vector such that $M(p) \geq M_{cov}(p)$. We have $M(p) \leq R + W$. We distinguish two cases:

1. For all intermediate vectors M' after M, $M'(p) \geq M(p)$. This means the number of tokens in p never goes below $M(p)$ after the vector M. Let σ' be the sub-word of σ that consists of all transition occurrences after M that has an arc to/from p. The sub-word σ' is safe for transfer from p to p_I. We transfer σ' from p to p_I and note that in the final vector reached after the transfer, p still has $M(p)$ tokens, which is enough to cover M_{cov}.

2. Let M' be the last intermediate vector such that $M'(p) < M(p)$. We invoke the truncation lemma by setting $c = M(p) \leq R+W$, $M_1 = M$ and $M_3 = M'$. We can then transfer the sub-word σ' identified by the truncation lemma to p_I to reduce the number of tokens in p in some intermediate vectors between M and M'. We repeat this process until there are no more than R' tokens in p in any intermediate vector between M and M'. Let M'' be the first intermediate vector after M' such that $M''(p) \geq M_{cov}(p)$. Again, $M''(p) \leq R+W$. If no intermediate vector M_3'' after M'' has $M_3''(p) < M''(p)$, we can transfer all transitions with an arc to/from p occurring after M'' to p_I. Otherwise, we can invoke truncation lemma again to ensure that p has at most R' tokens in any intermediate vector after M''.

Repeating the above case analysis for every independent place $p \in I$, we get a firing sequence π that is Q-covering from M_0 such that in all intermediate vectors, every independent place p has at most R' tokens. If this sequence happens to be $(W\ell(i) + R)$-bounded for Q, then $R'^m(W\ell(i)+R)^{i+1}$ is an upper bound on its length (since all independent places have at most R' tokens and the $i+1$ places in $Q \setminus I$ have at most $(W\ell(i) + R)$ tokens in all intermediate vectors) and we are done.

Otherwise, suppose there is some place $q \in Q \setminus I$ and some intermediate vector M such that $M(q) \geq W\ell(i) + R$. Let M be the first such vector and call the prefix of π up to M as π_1 and the rest of π as π_2. The length of π_1 is at most $R'^m(W\ell(i) + R)^{i+1}$. The sequence π_2 is a $(Q \setminus \{q\})$-covering sequence from M. By definition, there is such a sequence π_2' of length at most $\ell(i)$. The sequence $\pi_1\pi_2'$ is a $(Q \setminus \{q\})$-covering sequence from M_0. Since $M(q) \geq W\ell(i)+R$ and π_2' removes at most $W\ell(i)$ tokens from q, $\pi_1\pi_2'$ is in fact a Q-covering sequence from M_0. Its length is bounded by $R'^m(W\ell(i) + R)^{i+1} + \ell(i)$. □

The following lemma gives an upper bound on $\ell(i)$ using the recurrence relation obtained above.

Lemma 7.8. $\ell(i) \leq (2mWRR')^{m(i+1)!}$.

Proof. By induction on i. For $i = 0$, $\ell(0) \leq mR \leq (2mWRR')^{m1!}$.

$i = 1$:

$$\begin{aligned}
\ell(1) &\leq R'^m(W\ell(0) + R) + \ell(0) \\
&\leq R'^m(WmR + R) + mR \\
&\leq (WRR')^m mR + mR \\
&\leq (mWRR')^{2m} + mR \\
&\leq 2(mWRR')^{2m} \\
&\leq (2mWRR')^{m2!}
\end{aligned}$$

$i \geq 2$:

$$\begin{aligned}
\ell(i+1) &\leq R'^m(W\ell(i) + R)^{i+1} + \ell(i) \\
&\leq R'^m(W(2mWRR')^{m(i+1)!} + R)^{i+1} + (2mWRR')^{m(i+1)!} \\
&\leq (WRR')^{m(i+1)}(2mWRR')^{m(i+1)!(i+1)} + (2mWRR')^{m(i+1)!} \\
&\leq (2mWRR')^{m(i+1)}(2mWRR')^{m(i+1)!(i+1)} + (2mWRR')^{m(i+1)!} \\
&\leq (2mWRR')^{m(i+1)((i+1)!+1)} + (2mWRR')^{m(i+1)!} \\
&\leq 2(2mWRR')^{m(i+1)((i+1)!+1)} \\
&\leq (2mWRR')^{m(i+1)((i+1)!+2)} \\
&\leq (2mWRR')^{m(i+2)!}
\end{aligned}$$

The last step follows since

$$\begin{aligned}
i \geq 2 &\Rightarrow i! \geq 2 \\
&\Rightarrow (i+1)i! \geq 2(i+1) \\
&\Rightarrow (i+1)! \geq 2(i+1) \\
&\Rightarrow (i+1)(i+1)! + (i+1)! \geq (i+1)(i+1)! + 2(i+1) \\
&\Rightarrow (i+2)(i+1)! \geq (i+1)((i+1)! + 2) \\
&\Rightarrow (i+2)! \geq (i+1)((i+1)! + 2)
\end{aligned}$$

□

Theorem 7.9. *With the vertex cover number k and maximum arc weight W as parameters, the Petri net coverability problem can be solved in* ParaPspace.

Proof. From the Lemma 7.8, we get $\ell(k') \leq (2mWRR')^{m(k'+1)!}$. To guess and verify a covering sequence of length at most $\ell(k')$, a non-deterministic Turing machine needs to maintain a counter and intermediate markings, which can be done using memory size $\mathcal{O}(m(k'+1)!(m\log|M_0| + \log m + \log W + \log R + \log R'))$. An application of Savitch's theorem then gives us the ParaPspace algorithm. □

7.3 Model Checking Logic of Counting Properties

The ParaPspace algorithm of the previous section can be extended to the boundedness problem. We however consider the more general problem of model checking the logic of Petri net counting properties defined in section 6.3, which can express boundedness, coverability and some extensions. Following is the main theorem of this section.

Theorem 7.10. *Given a Petri net with an initial marking and a formula ϕ, if the vertex cover number k and the maximum arc weight W of the net are treated as parameters and the nesting depth D of **EF** modality in the formula is treated as a constant, then there is a* ParaPspace *algorithm that checks if the net satisfies the given formula.*

The details of model checking κ formulas is given in Sub-section 7.3.1. Sub-section 7.3.2 gives the details of a ParaPspace algorithm for model checking β formulas. The main ideas and definitions used are same as the ones in the previous chapter, but are repeated here since there are subtle differences due to partitioning the set of places into component places and buffers in the last chapter.

7.3.1 Nested Coverability Properties

As in section 6.4.1, we consider κ formulas of the form $\gamma \wedge \mathbf{EF}(\kappa_1) \wedge \cdots \wedge \mathbf{EF}(\kappa_r)$, with γ being a conjunction of $\tau \geq c$ formulas. The tree generated by such a formula with set of nodes Γ and $ratio(i)$ are as defined in the beginning of section 6.4.1. Recalling Definition 7.5, let $\ell'(M_{cov}) = \max\{lencov(P, M, M_{cov}) \mid M : P \to \mathbb{Z}\}$. The definition of bound function f and guess function h are the same as in Definition 6.12.

To derive an upper bound for $f(i)$ to use in a nondeterministic algorithm, let $R = \max\{ratio(\tau \geq c) \mid \tau \geq c \text{ is a subformula of } \kappa\}$, $R' = R + W + W^2 + W^3$ and $W' = \max\{W, 2\}$. The following lemma uses the result from Lemma 6.14 for the upper bound on $f(D-i-1, p)$.

Lemma 7.11. *Let $q(i) = (2m(k'+1)!)^i$. Then $\ell'(f(D-1)) \leq (2mW'R')^{q(1)}$ and also $\ell'(f(D-i)) \leq \prod_{j=D-i}^{D-1}((D-j+1)2mW'^8R')^{q(i+j+1-D)}$.*

Proof. $\ell'(f(D-1)) \leq (2mW'R')^{q(1)}$ is by Lemma 7.8. Next result is by induction on i.

Base case: $i = 2$. Since $f(D-2, p) \leq 3R'W\ell'(f(D-1))$ and $\ell'(f(D-2)) \leq (2mWr')^{q(1)}$ where $r' = \max\{f(D-2, p) \mid p \in P\} + W + W^2 + W^3$, we get

$$\begin{aligned}\ell'(f(D-2)) &\leq (2mW(3R'W\ell'(f(D-1)) + W + W^2 + W^3))^{q(1)} \\ &\leq (3 * 2mW'^8R')^{q(1)}(2mW'R')^{q(2)}\end{aligned}$$

Induction step: Since $f(D-i-1,p) \leq (i+2)R'W'\ell'(f(D-i))$, we have

$$\begin{aligned}
\ell'(f(D-i-1)) &\leq (2mW((i+2)R'W'\ell'(f(D-i)) + W + W^2 + W^3))^{q(1)} \\
&\leq \left((i+2)2mW'^8R' \prod_{j=D-i}^{D-1} ((D-j+1)2mW'^8R')^{q(i+j+1-D)}\right)^{q(1)} \\
&= \left((i+2)2mW'^8R'\right)^{q(1)} \prod_{j=D-i}^{D-1} \left((D-j+1)2mW'^8R'\right)^{q(i+1+j+1-D)} \\
&= \prod_{j=D-i-1}^{D-1} \left((D-j+1)2mW'^8R'\right)^{q(i+1+j+1-D)}
\end{aligned}$$

□

Theorem 7.12. *Given a Petri net with an initial marking and a κ formula ϕ, if the vertex cover number of the Petri net k and the maximum arc weight W are treated as parameters and the nesting depth D of **EF** modality in the formula is treated as a constant, then there is a* ParaPspace *algorithm that checks if the Petri net satisfies the given formula.*

Proof. First reduce ϕ to the form $\gamma \wedge \mathbf{EF}(\kappa_1) \wedge \cdots \wedge \mathbf{EF}(\kappa_r)$, with γ being a conjunction of $\tau \geq c$ formulas by nondeterministically choosing disjuncts from subformulas of ϕ. By Lemma 6.13, it is enough for a nondeterministic algorithm to guess sequences $\sigma_{\alpha j}$, $\alpha j \in \Gamma$ of lengths at most $\ell'(f(|\alpha j| - 1))$ and verify that they satisfy the formula. Using bounds given by Lemma 7.11 and an argument similar to the one in the proof of Theorem 7.9, it can be shown that the space used is exponential in k' and polynomial in the size of the net and numeric constants in the formula. This gives the ParaPspace algorithm. □

The space requirement of the above algorithm will have terms like m^{2D} and hence it will not be ParaPspace if D is treated as a parameter instead of a constant.

7.3.2 Boundedness Properties

In order to check the truth of β formulas, we adapt the concept of *disjointness sequence* introduced in [19] to our notation, as done in the previous chapter.

Definition 7.13 ([19])**.** *Let $X \subseteq P$ be a non-empty subset of places. If $\sigma = t_1 \cdots t_r$ is a sequence of transitions and p is a place, $\Delta[\sigma](p)$ denotes the total effect of σ on p: $\Delta[\sigma](p) = \sum_{i=1}^{r} Post(p,t_i) - Pre(p,t_i)$. A firing sequence σ enabled at an initial marking $M_0 : P \to \mathbb{N}$ is said to be a X-**pumping sequence** if σ can be decomposed as $\sigma'_1\underline{\sigma_1}\sigma'_2\underline{\sigma_2}\cdots\sigma'_e\underline{\sigma_e}$ such that*

1. *For each $p \in P$, $\Delta[\underline{\sigma_1}](p) \geq 0$ and for each ρ between 2 and e, $\Delta[\underline{\sigma_\rho}](p) < 0$ implies there is a $j \leq \rho - 1$ such that $\Delta[\underline{\sigma_j}](p) > 0$ and*
2. *$X \subseteq \bigcup_{\rho=1}^{e} \{p \in P \mid \Delta[\underline{\sigma_\rho}](p) > 0\}$.*

The subsequences $\underline{\sigma_1}, \ldots, \underline{\sigma_e}$ are called pumping portions of the pumping sequence. They are underlined to distinguish them from non-pumping portions of the sequence.

As shown in Lemma 6.18 in the previous chapter, checking $\mathcal{N}, M_0 \models \{\tau_1, \ldots, \tau_r\} = \omega$ is equivalent to checking the existence of a X-pumping sequence for some $X \subseteq P$ such that for every $j \in \{1, \cdots, r\}$, there is a $p_j \in X$ with $L_{\tau_j}(p_j) \geq 1$.

Lemma 7.14 ([19])**.** *$\mathcal{N}, M_0 \models \{\tau_1, \ldots, \tau_r\} = \omega$ iff there exists a X-pumping sequence for some $X \subseteq P$ such that for every $j \in \{1, \cdots, r\}$, there is a $p_j \in X$ with $L_{\tau_j}(p_j) \geq 1$.*

Proof. Same as proof of Lemma 6.18. □

Model checking β formulas thus reduces to detecting the presence of certain X-pumping sequences. The following definition adapted from [19] is a generalization of Q-enabled sequences.

Definition 7.15 ([19]). *Let $I \subseteq Q \subseteq P$ be a subset of places that contains all independent places, $Y \subseteq P$ a possibly empty subset of places and $X \subseteq P$ a non-empty subset of places. Let $M : P \to \mathbb{Z}$ and $c \in \mathbb{N} \cup \{\omega\}$. A sequence of transitions is said to be a* **Y-neglecting weakly M, Q, c-enabled X-pumping sequence** *if it can be decomposed as $\sigma'_1 \underline{\sigma_1} \sigma'_2 \underline{\sigma_2} \cdots \sigma'_e \underline{\sigma_e}$ such that*

1. *For each $1 \leq \rho \leq e$, for each $p \in P$, $\Delta[\underline{\sigma_\rho}](p) < 0$ implies (there is a $1 \leq j \leq \rho - 1$ such that $\Delta[\underline{\sigma_j}](p) > 0$ or $p \in Y$).*
2. *$X \subseteq \bigcup_{\rho=1}^{e} \{p \in P \mid \Delta[\underline{\sigma_\rho}(p)] > 0\} \setminus Y$.*
3. *For any intermediate vector M' and any place $p \in Q \setminus I$, $M'(Q) < c$.*
4. *For any intermediate vector M' and any place $p \in Q$, $M'(p) < 0$ implies (there is a $\underline{\sigma_j}$ occurring before M' such that $\Delta[\underline{\sigma_j}](p) > 0$ or $p \in Y$).*

Intuitively, a Y-neglecting weakly M, Q, c-enabled X-pumping sequence maintains the number of tokens between 0 and c in all places in Q while in other places, it can become less than 0 or more than c. If a place $p \in Q$ has already been pumped up by some pumping portion $\underline{\sigma_j}$, p may have negative number of tokens in intermediate vectors that occur after $\underline{\sigma_j}$. The following lemma implies that for detecting the presence of pumping sequences, it is enough to detect certain weakly enabled pumping sequences.

Lemma 7.16 ([19]). *Let $X \subseteq P$ be a non-empty subset of places and $M_0 : P \to \mathbb{N}$ be the initial marking. Suppose that $\sigma = \sigma'_1 \underline{\sigma_1} \sigma'_2 \underline{\sigma_2} \cdots \sigma'_e \underline{\sigma_e}$ is a $\emptyset$-neglecting weakly M_0, P, ω-enabled X-pumping sequence. Then, there are $n_1, n_2, \ldots, n_e \in \mathbb{N}$ such that $\sigma'_1 \underline{\sigma_1}^{n_1} \sigma'_2 \underline{\sigma_2}^{n_2} \cdots \sigma'_e \underline{\sigma_e}^{n_e}$ is a X-pumping sequence enabled at M_0.*

Proof. We define $n_e, \ldots, n_1$ in that order as follows:

- $n_e = 1$.
- Suppose $1 \leq \rho < e$ and $n_{\rho+1}, \ldots, n_e$ have already been defined. Define n_ρ to be $(e - \rho)(|\sigma| - 1)W + \sum_{j=\rho+1}^{e} (|\sigma| - 1)W n_j$.

We will prove that $\sigma' = \sigma'_1 \underline{\sigma_1}^{n_1} \sigma'_2 \underline{\sigma_2}^{n_2} \cdots \sigma'_e \underline{\sigma_e}^{n_e}$ satisfies all conditions of Definition 7.13 and that it is enabled at M_0. Condition 2 follows by the fact that σ satisfies condition 2 of Definition 7.15 and that $Y = \emptyset$. Condition 1 of Definition 7.13 follows by the fact that σ satisfies condition 1 of Definition 7.15 and that $Y = \emptyset$. For proving that σ' is enabled at M_0, we will prove the following claim by induction on ρ: for any intermediate marking M' occurring when firing $\sigma'_1 \underline{\sigma_1}^{n_1} \cdots \sigma'_\rho \underline{\sigma_\rho}^{n_\rho}$ from M_0 and any $p \in P$, $M'(p) \geq 0$; and for any intermediate marking M'' occurring while firing σ' from M_0 and any $p' \in \bigcup_{j=1}^{\rho} \{p \in P \mid \Delta[\underline{\sigma_j}](p) > 0\}$, $M''(p) \geq 0$.

Base case $\rho = 1$: Since $Y = \emptyset$ and σ satisfies condition 4 of Definition 7.15, for any intermediate marking M' occurring when firing $\sigma'_1 \underline{\sigma_1}$ from M_0 and any place $p \in P$, $M'(p) \geq 0$. Since σ satisfies condition 1 of Definition 7.15 and $Y = \emptyset$, $\Delta[\underline{\sigma_1}](p) \geq 0$ for any place $p \in P$. Hence, for any intermediate marking M' occurring when firing $\sigma'_1 \underline{\sigma_1}^{n_1}$ from M_0 and any place $p \in P$, $M'(p) \geq 0$. Since $|\sigma'_2 \cdots \sigma'_e| \leq (e - 1)(|\sigma| - 1)$ and $|\underline{\sigma_2}^{n_2} \cdots \underline{\sigma_e}^{n_e}| \leq \sum_{j=2}^{e} (|\sigma| - 1)n_j$,

$\sigma'_2\underline{\sigma_2}^{n_2}\cdots\sigma'_e\underline{\sigma_e}^{n_e}$ can decrease at most $(e-1)(|\sigma|-1)W+\sum_{j=2}^{e}(|\sigma|-1)Wn_j$ tokens from any place. If $M_0 \xrightarrow{\sigma_1\underline{\sigma_1}^{n_1}} M_1$ and $\Delta[\underline{\sigma_1}](p)>0$ for any place p, then $M_1(p)\geq(e-1)(|\sigma|-1)W+\sum_{j=2}^{e}(|\sigma|-1)Wn_j$. Hence, the second part of the claim follows.

Induction step: Assume that $M_0 \xrightarrow{\sigma'_1\underline{\sigma_1}^{n_1}\cdots\sigma'_\rho\underline{\sigma_\rho}^{n_\rho}} M_\rho$. Suppose for some place p' and some intermediate marking M' that occurs while firing $\sigma'_{\rho+1}\underline{\sigma_{\rho+1}}$ from M_ρ, $M'(p)<0$. By induction hypothesis, $p'\notin\bigcup_{j=1}^{\rho}\{p\in P\mid\Delta[\underline{\sigma_j}](p)>0\}$, which contradicts the fact that σ satisfies conditions 1 and 4 of Definition 7.15. Also from condition 1 of Definition 7.15, $\Delta[\underline{\sigma_{\rho+1}}](p)\geq 0$ for any $p\notin\bigcup_{j=1}^{\rho}\{p\in P\mid\Delta[\underline{\sigma_j}](p)>0\}$. Hence, for all $p\in P$ and any intermediate marking M' that occurs while firing $\sigma'_{\rho+1}\underline{\sigma_{\rho+1}}^{n_{\rho+1}}$ from M_ρ, $M'(p)\geq 0$. Suppose $\rho+2\leq e$. Since $|\sigma'_{\rho+2}\cdots\sigma'_e|\leq(e-\rho-1)(|\sigma|-1)$ and $|\underline{\sigma_{\rho+2}}^{n_{\rho+2}}\cdots\underline{\sigma_e}^{n_e}|\leq\sum_{j=\rho+2}^{e}(|\sigma|-1)n_j$, $\sigma'_{\rho+2}\underline{\sigma_{\rho+2}}^{n_{\rho+2}}\cdots\sigma'_e\underline{\sigma_e}^{n_e}$ can decrease at most $(e-\rho-1)(|\sigma|-1)W+\sum_{j=\rho+2}^{e}(|\sigma|-1)Wn_j$ tokens from any place. If $M_\rho \xrightarrow{\sigma'_{\rho+1}\underline{\sigma_{\rho+1}}^{n_{\rho+1}}} M_{\rho+1}$ and $\Delta[\underline{\sigma_{\rho+1}}](p)>0$ for any place p, then $M_{\rho+1}(p)\geq(e-\rho-1)(|\sigma|-1)W+\sum_{j=\rho+2}^{e}(|\sigma|-1)Wn_j$. Hence, second part of the claim follows. □

As is done in section 7.2, we will bound the length of weakly enabled pumping sequences by induction on $|Q|$. The rest of this sub-section is more technical than section 7.2. Here, we try to give an intuitive explanation of the purpose of each lemma in the rest of this section. If we want to check for unboundedness in a specific set of places, the "pumping up" of tokens may happen in several stages, captured by Definition 7.13. See [19, Subsection 3.1] for examples of scenarios where several stages are needed. Lemma 7.17 and Lemma 7.18 provide some basic tools to manipulate pumping sequences, such as repeating certain portions and combining two sequences. In Lemma 7.21, we show how loops can be carefully removed (using linear algebraic techniques) separately from the different stages of a pumping sequence. Lemma 7.22 and Lemma 7.23 give a recurrence relation for the length of a shortest pumping sequence, using the tools we developed earlier for manipulating pumping sequences. Lemma 7.24 gives an upper bound for the recurrence relation. Lemma 7.25 shows how to use the truncation lemma so that the upper bound obtained in Lemma 7.24 is asymptotically smaller.

Lemma 7.17 ([19]). *Suppose $\sigma=\sigma'_1\underline{\sigma_1}\sigma'_2\cdots\sigma'_e\underline{\sigma_e}$ is a Y-neglecting weakly M,Q,ω-enabled X-pumping sequence. Then the sequence $\sigma'=\sigma'_1\sigma_1^{n_1}\underline{\sigma_1}\sigma_1^{n'_1}\sigma'_2\cdots\sigma'_e\sigma_e^{n_e}\underline{\sigma_e}$ is also a Y-neglecting weakly M,Q,ω-enabled X-pumping sequence for any $n_1,n'_1,\ldots,n_e\in\mathbb{N}$ (σ_ρ is same as $\underline{\sigma_\rho}$, except that σ_ρ is not considered a pumping portion while $\underline{\sigma_\rho}$ is considered a pumping portion).*

Proof. We will prove that the new sequence satisfies all the conditions of Definition 7.15. Conditions 1 and 2 are satisfied since the set of pumping portions of the new sequence is equal to that of the old one and occurs in the same order. Condition 3 is trivially satisfied since in this case, $c=\omega$. Suppose for some intermediate vector M' and some place $p\in Q$, $M'(p)<0$. Let j be the maximum number such that $\underline{\sigma_j}$ occurs before M'. Suppose $M \xrightarrow{\sigma'_1\sigma_1^{n_1}\underline{\sigma_1}\sigma_1^{n'_1}\sigma'_2\cdots\sigma'_j\sigma_j^{n_j}\underline{\sigma_j}} M''$ and $M''\xrightarrow{\eta}M'$. If $p\in Y$ or $p\in\bigcup_{j'=1}^{j}\{p'\in P\mid\Delta[\underline{\sigma_{j'}}](p')>0\}$, there is nothing else to prove. Otherwise, $\Delta[\underline{\sigma_{j'}}](p)=0$ for every j' between 1 and j. This implies that if $M \xrightarrow{\sigma'_1\underline{\sigma_1}\sigma'_2\cdots\sigma'_j\underline{\sigma_j}} M_2$ and $M_2\xrightarrow{\eta}M_3$, then $M_3(p)<0$, contradicting the fact that σ satisfies condition 4 of Definition 7.15. □

Lemma 7.18. *Suppose $\sigma=\sigma'_1\underline{\sigma_1}\cdots\sigma'_e\underline{\sigma_e}$ is a Y-neglecting weakly M,Q,ω-enabled X_1-pumping sequence and $\delta=\delta'_1\underline{\delta_1}\cdots\delta'_{e'}\underline{\delta_{e'}}$ is a Y_1-neglecting weakly M_1,Q,ω-enabled X_2-pumping sequence. If $Y_1=Y\cup\{p\in P\mid\Delta[\underline{\sigma_\rho}](p)>0,1\leq\rho\leq e\}$, $M\xrightarrow{\sigma}M_2$ and for*

all $p \in Q \setminus Y_1$, $M_2(p) = M_1(p)$, then $\sigma\delta = \sigma'_1\underline{\sigma_1} \cdots \sigma'_e\underline{\sigma_e}\delta'_1\underline{\delta_1} \cdots \delta'_{e'}\underline{\delta_{e'}}$ is a Y-neglecting weakly M, Q, ω-enabled $(X_1 \cup X_2)$-pumping sequence.

Proof. We will prove that the combined sequence satisfies all conditions of Definition 7.15.

1. This follows since σ and δ individually satisfy condition 1 of Definition 7.15 and $Y_1 = Y \cup \{p \in P \mid \Delta[\underline{\sigma_\rho}](p) > 0, 1 \leq \rho \leq e\}$.

2. This follows from the fact that X_1 and X_2 individually satisfy condition 2 of Definition 7.15.

3. This is trivially satisfied since in this case, $c = \omega$.

4. Suppose M' is some intermediate vector that occurs while firing δ from M_2 with $M'(p) < 0$ for some $p \in Q$. If $p \in Y_1$ or there is some $\underline{\delta_{\rho'}}$ occurring before M' such that $\Delta[\underline{\delta_{\rho'}}](p) > 0$, there is nothing more to prove. Otherwise, the fact that $p \in Q \setminus Y_1$ and $M_2(p) = M_1(p)$ contradicts the fact that δ is a Y_1-neglecting weakly M_1, Q, ω-enabled X_2-pumping sequence, that should have satisfied condition 4 of Definition 7.15.

□

Definition 7.19. *Let $Q, X, Y \subseteq P$ be subsets of places such that $I \subseteq Q$ and X is non-empty. Suppose $\sigma = \sigma'_1\underline{\sigma_1} \cdots \sigma'_e\underline{\sigma_e}$ is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence for some $M : P \to \mathbb{Z}$. For some independent place $p \in I$, if there is a j such that $\Delta[\underline{\sigma_j}] > 0$, we do not care if p has negative number of tokens in some intermediate vector that occurs after $\underline{\sigma_j}$, even if $p \notin Y$. For each $p \in I \setminus Y$, let $j[p]$ be the minimum number such that $\Delta[\underline{\sigma_{j[p]}}](p) > 0$. If $M \xrightarrow{\sigma'_1\underline{\sigma_1}\cdots\sigma'_{j[p]}\underline{\sigma_{j[p]}}} M_1$, then the set of all intermediate vectors occurring between M and M_1 (including M and M_1) is called the* **caring zone of** *p. If there is no $\underline{\sigma_j}$ such that $\Delta[\underline{\sigma_j}](p) > 0$, then the caring zone of p is the set of all intermediate vectors.*

Definition 7.20. *Let $U' \in \mathbb{N}$ be some fixed number. For $j \in \mathbb{N}$, $Q, X, Y \subseteq P$ with $I \subseteq Q$ and X non-empty and a function $M : P \to \mathbb{Z}$, $\lambda(Q, j, M, X, Y)$ is the length of the shortest Y-neglecting weakly M, Q, ω-enabled X-pumping sequence from M if there is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence from M in which every independent place $p \in I \setminus Y$ has at most $U' + jW$ tokens in all intermediate vectors belonging to the caring zone of p. Let $\lambda(Q, j, M, X, Y)$ be 0 if there is no such sequence. Let $\ell_2(i, j) = \max\{\lambda(Q, j, M, X, Y) \mid I \subseteq Q \subseteq P, |Q \setminus I| = i, M : P \to \mathbb{Z}, X, Y \subseteq P, X \neq \emptyset\}$.*

Lemma 7.21. *Let $Q, X, Y \subseteq P$ be subsets of places such that $I \subseteq Q$ and X is non-empty and let $U' \in \mathbb{N}$. Let $c \in \mathbb{N}$. Suppose there is a Y-neglecting weakly M, Q, c-enabled X-pumping sequence $\sigma = \sigma'_1\underline{\sigma_1} \cdots \sigma'_e\underline{\sigma_e}$ for some $M : P \to \mathbb{Z}$ such that every place $p \in I \setminus Y$ has at most U' tokens in all intermediate vectors belonging to the caring zone of p. Then, there is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence of length at most $8ek'(2c)^{c'k'^3}(U'W)^{c'm^4}$ for some constant c'.*

Proof. By induction on e.

Base case $e = 1$: In this case, $\sigma = \sigma'_1\underline{\sigma_1}$. All intermediate vectors occurring as a result of firing σ from M belong to the caring zone of each place $p \in I \setminus Y$. If any two intermediate vectors occurring when σ'_1 is fired from M agree on all places in $Q \setminus Y$, then the subsequence between them can be removed. Hence, we can assume without loss of generality that $|\sigma'_1| \leq U'^m c^{k'}$.

As in Rackoff's proof of Lemma 4.5 in [83], remove $Q \setminus Y$-loops from $\underline{\sigma_1}$ carefully until what remains behind is a sequence σ''_1 of length at most $(U'^m c^{k'} + 1)^2$. Let $\mathbf{b} \in \mathbb{N}^{|S \setminus Y|}$ be

the vector containing a 1 in each coordinate corresponding to a special place in $S \setminus Y$ whose number of tokens is increased by $\underline{\sigma_1}$ and 0 in all other coordinates. If π is a $Q \setminus Y$-loop, its **loop value** is the vector in $\mathbb{Z}^{|S \setminus Y|}$, which contains in each coordinate the total effect of π on the corresponding special place in $S \setminus Y$. Let $\mathbf{L} \subseteq \mathbb{Z}^{|S \setminus Y|}$ be the set of loop values that were removed from $\underline{\sigma_1}$. Let $\mathbf{B}$ be the matrix with $|S \setminus Y|$ rows, whose columns are the members of $\mathbf{L}$. For any sequence π, let $\mathbf{ef}(\pi)$ be the vector in $\mathbb{Z}^{|S \setminus Y|}$, which contains in each coordinate the total effect of π on the corresponding special place in $S \setminus Y$. By definition, $\mathbf{ef}(\underline{\sigma_1}) \geq \mathbf{b}$. The effect of $\underline{\sigma_1}$ can be split into the effect of σ_1'' and the effect of $Q \setminus Y$-loops that were removed from $\underline{\sigma_1}$. If $\mathbf{x}(i)$ is the number of $Q \setminus Y$-loops removed from $\underline{\sigma_1}$ whose loop value is equal to the i^{th} column of $\mathbf{B}$, then we have $\mathbf{Bx} \geq \mathbf{b} - \mathbf{ef}(\sigma_1'')$.

A loop value is just the effect of at most $c^{k'}U'^m$ transitions, and hence each entry of $\mathbf{B}$ is of absolute value at most $c^{k'}U'^m W$. The matrix $\mathbf{B}$ has therefore at most $(2c^{k'}U'^m W + 1)^{k'}$ columns. Each entry of $\mathbf{b} - \mathbf{ef}(\sigma_1'')$ is of absolute value at most $W(c^{k'}U'^m + 1)^2 + 1$. Letting $d_1 = k'$ and $d = \max\{(2c^{k'}U'^m W + 1)^{k'}, c^{k'}U'^m W, W(c^{k'}U'^m + 1)^2 + 1\} \leq (2c)^{3k'}(U'W)^{3m^2}$, we can apply Lemma 4.4 of [83]. The result is that there is a vector $\mathbf{y} \in \mathbb{N}^{|\mathbf{L}|}$ such that the sum of entries of $\mathbf{y}$ is equal to $l_1 \leq d((2c)^{3k'}(U'W)^{3m^2})^{ck'}$ for some constant c. Let c' be a constant such that $l_1 \leq k'(2c)^{c'k'^2}(U'W)^{c'm^3}$.

Now, we will put back l_1 $Q \setminus Y$-loops back to σ_1'', which was of length at most $(c^{k'}U'^m + 1)^2$. Since the length of each $Q \setminus Y$-loop is at most $c^{k'}U'^m$, the total length of the newly constructed pumping portion is at most $(c^{k'}U'^m + 1)^2 + k'(2c)^{c'k'^3}(U'W)^{c'm^4}$. Together with σ_1, whose length is at most $c^{k'}U'^m$, we get a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence of length at most $2(c^{k'}U'^m + 1)^2 + k'(2c)^{c'k'^3}(U'W)^{c'm^4} \leq 8k'(2c)^{c'k'^3}(U'W)^{c'm^4}$.

Induction step: Suppose $\sigma = \sigma_1'\underline{\sigma_1} \cdots \sigma_{e+1}'\underline{\sigma_{e+1}}$. Let $X_1 = \{p \in P \mid \Delta[\underline{\sigma_1}](p) > 0\}$. The sequence $\sigma_1'\underline{\sigma_1}$ is a Y-neglecting weakly M, Q, c-enabled X_1-pumping sequence. Let $M \xrightarrow{\sigma_1'\underline{\sigma_1}} M_1$. As is done in the base case, we can replace $\sigma_1'\underline{\sigma_1}$ by another Y-neglecting weakly M, Q, ω-enabled X_1-pumping sequence σ' of length at most $8k'(2c)^{c'k'^3}(U'W)^{c'm^4}$ ending at some vector M_2 such that for all $p \in Q \setminus Y$, $M_2(p) = M_1(p)$ (this is because we only remove $Q \setminus Y$ loops from $\sigma_1'\underline{\sigma_1}$ to obtain the shorter sequence σ').

The sequence $\sigma_2'\underline{\sigma_2} \cdots \sigma_{e+1}'\underline{\sigma_{e+1}}$ is a $(Y \cup X_1)$-neglecting weakly M_1, Q, c-enabled $(X \setminus X_1)$-pumping sequence. By induction hypothesis, there is another $(Y \cup X_1)$-neglecting weakly M_1, Q, ω-enabled $(X \setminus X_1)$-pumping sequence σ'' of length at most $8k'e(2c)^{c'k'^3}(U'W)^{c'm^4}$. Lemma 7.18 implies that $\sigma'\sigma''$ is a Y-neglecting weakly M, Q, ω-enabled $(X \setminus X_1) \cup X_1$-pumping sequence. The length of $\sigma'\sigma''$ is at most $8k'(e+1)(2c)^{c'k'^3}(U'W)^{c'm^4}$. □

Using the technical lemmas proved above, we will now obtain a recurrence relation for ℓ_2.

Lemma 7.22. $\ell_2(0, j) \leq 8mk'(2W(U' + jW))^{c'm^4}$.

Proof. By Lemma 7.21 after setting $c = 1$ and substituting U' by $U' + jW$. □

Lemma 7.23. $\ell_2(i+1, j) \leq 10mk'(2W\ell_2(i, j+1))^{c'k'^3}((U' + jW)W)^{c'm^4}$.

Proof. Let $Q, X, Y \subseteq P$ be subsets of places such that $I \subseteq Q$, $|Q \setminus I| = i + 1$ and X is non-empty. Let $M : P \to \mathbb{Z}$ be some vector. Suppose there is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence σ such that every independent place $p \in I \setminus Y$ has at most $U' + jW$ tokens in any intermediate vector belonging to the caring zone of p. We will prove that there is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence of length at most $10mk'(2W\ell_2(i, j+1))^{c'k'^3}((U + jW)W)^{c'm^4}$.

Case 1: The sequence σ is a Y-neglecting weakly $M, Q, W\ell_2(i, j+1)$-enabled X-pumping sequence. The required result is a consequence of Lemma 7.21, after substituting $U' + jW$ for U'.

Case 2: The sequence σ decomposes into $\sigma = \sigma'_1\underline{\sigma_1}\cdots\sigma'_e\underline{\sigma_e}$ such that for some $2 \leq \rho \leq e$, $M \xrightarrow{\sigma'_1\underline{\sigma_1}\cdots\underline{\sigma_{\rho-1}}} M_1 \xrightarrow{\sigma'_\rho} M_2$ and there is some intermediate vector M' between M_1 and M_2 and a place $q \in Q \setminus Y$ with $M'(q) \geq W\ell_2(i, j+1)$. Let M' be the earliest such intermediate vector occurring outside of pumping portions. If there is some $\rho > 1$ such that $\{p \in P \mid \Delta[\underline{\sigma_\rho}](p) > 0\} \subseteq \bigcup_{j=1}^{\rho-1}\{p \in P \mid \Delta[\underline{\sigma_j}](p) > 0\}$, then $\underline{\sigma_\rho}$ can be considered as a non-pumping portion and the resulting sequence will still be a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence. Hence, without loss of generality, we can assume that $e \leq m$. Let $M_1 \xrightarrow{\sigma'^{1}_\rho} M' \xrightarrow{\sigma'^{2}_\rho} M_2$. Let $X_1 = \bigcup_{j=1}^{\rho-1}\{p \in P \mid \Delta[\underline{\sigma_j}](p) > 0\}$. The sequence $\sigma'_1\underline{\sigma_1}\cdots\underline{\sigma_{\rho-1}}$ is a Y-neglecting weakly $M, Q, W\ell_2(i, j+1)$-enabled X_1-pumping sequence in which every place $p \in Q \setminus Y$ has at most $U' + jW$ tokens in all intermediate vectors belonging to the caring zone of p. By Lemma 7.21, there is a Y-neglecting weakly M, Q, ω-enabled X_1-pumping sequence δ_1 of length at most $8(\rho-1)k'(2W\ell_2(i, j+1))^{c'k'^3}((U' + jW)W)^{c'm^4}$. We can remove all $(Q \setminus Y \setminus X_1)$-loops from σ'^{1}_ρ to obtain δ'^{1}_ρ of length at most $(W\ell_2(i, j+1))^{k'}(U' + jW)^m$. If $M \xrightarrow{\delta_1} M'_1 \xrightarrow{\delta'^{1}_\rho} M'' \xrightarrow{\sigma'^{2}_\rho} M'_2$, we will have $M''(p) = M'(p)$ for all $p \in (Q \setminus Y \setminus X_1)$.

The sequence $\sigma'^{2}_\rho\underline{\sigma_\rho}\cdots\sigma'_e\underline{\sigma_e}$ is a $(Y \cup X_1)$-neglecting weakly M', Q, ω-enabled $(X \setminus X_1)$-pumping sequence such that every independent place $p \in I \setminus (Y \cup X_1)$ has at most $U' + jW$ tokens in all intermediate vectors belonging to the caring zone of p. By definition, there is a $(Y \cup X_1)$-neglecting weakly $M', Q \setminus \{q\}, \omega$-enabled $(X \setminus X_1)$-pumping sequence δ_2 of length at most $\ell_2(i, j)$. If $q \in X_1$, then δ_2 is also a $(Y \cup X_1)$-neglecting weakly M', Q, ω-enabled $(X \setminus X_1)$-pumping sequence. Otherwise, $M''(q) = M'(q) \geq W\ell_2(i, j)$ and δ_2 can decrease at most $W\ell_2(i, j)$ tokens from q, so again δ_2 is a $(Y \cup X_1)$-neglecting weakly M', Q, ω-enabled $(X \setminus X_1)$-pumping sequence. In either case, Lemma 7.18 implies that $\delta_1\delta'^{1}_\rho\delta_2$ is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence. Its length is at most $8ek'(2W\ell_2(i, j+1))^{c'k'^3}((U' + jW)W)^{c'm^4} + (W\ell_2(i, j+1))^{k'}(U' + jW)^m + \ell_2(i, j)$.

Case 3: The sequence σ decomposes into $\sigma = \sigma'_1\underline{\sigma_1}\cdots\sigma'_e\underline{\sigma_e}$ such that for some intermediate vector M' occurring while firing σ'_1 from M, there is some place $q \in Q \setminus Y$ such that $M'(q) \geq W\ell_2(i, j)$. Let M' be the first such intermediate vector. Let $M \xrightarrow{\sigma'^{1}_1} M' \xrightarrow{\sigma'^{2}_1} M_1$. Remove all $Q \setminus Y$-loops from σ'^{1}_1 to get δ'^{1}_1 of length at most $(W\ell_2(i, j+1))^{k'}(U' + jW)^m$. In addition, $M \xrightarrow{\delta'^{1}_1} M''$ such that $M''(p) = M'(p)$ for all $p \in Q \setminus Y$. The sequence $\sigma'^{2}_1\underline{\sigma_1}\cdots\underline{\sigma_e}$ is a Y-neglecting weakly $M', Q \setminus \{q\}, \omega$-enabled X-pumping sequence such that every independent place $p \in I \setminus Y$ has at most $U' + jW$ tokens in any intermediate vector belonging to the caring zone of p. By definition, there is a Y-neglecting weakly $M', Q \setminus \{q\}, \omega$-enabled X-pumping sequence δ of length at most $\ell_2(i, j)$. Since δ can decrease at most $W\ell_2(i, j)$ tokens from q and $M'(q) = M''(q) \geq W\ell_2(i, j)$, δ is also a Y-neglecting weakly M', Q, ω-enabled X-pumping sequence. Hence, $\sigma'^{1}_1\delta$ is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence.

Case 4: The sequence σ decomposes into $\sigma = \sigma'_1\underline{\sigma_1}\cdots\sigma'_e\underline{\sigma_e}$ such that for some $1 \leq \rho \leq e$, $M \xrightarrow{\sigma'_1\underline{\sigma_1}\cdots\sigma'_\rho} M_1 \xrightarrow{\sigma_\rho} M_2$ and there is some intermediate vector M' between M_1 and M_2 and a place $q \in Q \setminus Y$ with $M'(q) \geq W\ell_2(i, j+1)$. For every independent place $p \in I \setminus Y$, if $\Delta[\underline{\sigma_\rho}](p) > W$, transfer to p_I the last transition in $\underline{\sigma_\rho}$ that adds tokens to p, where I is the $V\overline{C}$-interface of p. Repeat this until for every $p \in \overline{I} \setminus Y$ with $\Delta[\underline{\sigma_\rho}](p) > 0$, no more than W and no less than 1 tokens are added by the new pumping portion after the transfers. By Lemma 7.17, $\sigma'_1\underline{\sigma_1}\cdots\sigma'_\rho\sigma_\rho\underline{\sigma_\rho}\cdots\underline{\sigma_e}$ (obtained by repeating σ_ρ once and treating only the second occurrence as pumping portion) is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence such that every independent place $p \in I \setminus Y$ has at most $U' + (j+1)W$ tokens in any intermediate vector belonging to the caring zone of p. Now, we are back to case 2 or case 3 with $(j+1)$ replacing j. □

We will denote $c'k'^3$ by h.

Lemma 7.24. *$\ell_2(i,j) \leq (10mk')^{(1+h)^i}(2W)^{poly_1(h^i)}(U'+(j+i)W)^{poly_2(h^i)}$ where $poly_1(h^i)$ and $poly_2(h^i)$ are polynomials in h^i, c', k' and m.*

Proof. By induction on i. $\ell_2(0,j) \leq 8mk'(2(U'+jW)W)^{c'm^4}$. We will choose $poly_1$ and $poly_2$ such that $8mk'(2(U'+jW)W)^{c'm^4} \leq 10mk'(2W)^{poly_1(1)}(U'+jW)^{poly_2(1)}$.

$$\begin{aligned}
\ell_2(i+1,j) &\leq 10mk'(2W\ell_2(i,j+1))^h((U'+jW)W)^{c'm^4} \\
&\leq 10mk'\left[2W(10mk')^{(1+h)^i}(2W)^{poly_1(h^i)}(U'+(j+1+i)W)^{poly_2(h^i)}\right]^h \\
&\quad ((U'+jW)W)^{c'm^4} \\
&\leq (10mk')^{1+h(1+h)^i}(2W)^{(1+poly_1(h^i))h+c'm^4}(U'+(j+i+1)W)^{poly_2(h^i)h+c'm^4}
\end{aligned}$$

It is now enough to choose $poly_1$ and $poly_2$ such that $poly_1(h^0) \geq c'm^4$, $poly_1(h^{i+1}) \geq (1+poly_1(h^i))h + c'm^4$, $poly_2(h^0) \geq c'm^4$ and $poly_2(h^{i+1}) \geq poly_2(h^i)h + c'm^4$. These conditions are met by $poly_1(h^i) = h^ic'm^4 + (h+c'm^4)(h^i-1)$ and $poly_2(h^i) = h^ic'm^4 + c'm^4(h^i-1)$, assuming $h \geq 2$. □

For the upper bound obtained in Lemma 7.24 to be useful, we should have a pumping sequence in which independent places have controlled number of tokens in intermediate markings (i.e., U' and j are bounded). The following lemma establishes this with the help of truncation lemma.

Lemma 7.25. *Let $Q, X, Y \subseteq P$ be subsets of places such that $I \subseteq Q$ and X is non-empty. For some $M : P \to \mathbb{Z}$, suppose σ is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence. Let U be the maximum of the range of M and let $U' = U + W + W^2 + W^3$. There is a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence in which every independent place $p \in I \setminus Y$ has at most U' tokens in all intermediate vectors belonging to the caring zone of p.*

Proof. Suppose σ is of the form $\sigma = \sigma'_1\underline{\sigma_1}\sigma'_2\underline{\sigma_2}\cdots\sigma'_e\underline{\sigma_e}$. Ensure that for every independent place $p \in I \setminus Y$ and $1 \leq \rho \leq e$, if $\Delta[\underline{\sigma_\rho}](p) > 0$, then $\Delta[\underline{\sigma_\rho}](p) \geq 2W$. If this is not the case, we can repeat $\underline{\sigma_\rho}$ $2W$ times.

By Lemma 7.17, $\sigma'_1\sigma_1\underline{\sigma_1}\sigma_1\sigma'_2\sigma_2\underline{\sigma_2}\sigma_2\cdots\sigma'_e\sigma_e\underline{\sigma_e}$ is also a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence. Consider some $1 \leq \rho \leq e$ and an independent place $p \in I \setminus Y$ such that $\Delta[\underline{\sigma_\rho}](p) = 0$ and $\underline{\sigma_\rho}$ occurs within the caring zone of p. Let $M \xrightarrow{\sigma'_1\sigma_1\underline{\sigma_1}\sigma_1\cdots\sigma'_{\rho-1}} M_1 \xrightarrow{\sigma_\rho} M_3 \xrightarrow{\underline{\sigma_\rho}} M_4 \xrightarrow{\sigma_\rho} M_6$. Let $c_1 = \min\{M'(p) \mid M' \text{ occurs between } M_1 \text{ and } M_3\}$ be the minimum number of tokens in p among all intermediate vectors occurring between M_1 and M_3. Let M_2 be the first intermediate vector between M_1 and M_3 such that $M_2(p) = c_1$ (see Fig. 7.3). Similarly, let $c_2 = \min\{M'(p) \mid M' \text{ occurs between } M_4 \text{ and } M_6\}$ be the minimum number of tokens in p among all intermediate vectors occurring between M_4 and M_6. Let M_5 be the last intermediate vector occurring between M_4 and M_6 such that $M_5(p) = c_2$. Note that since $\Delta[\underline{\sigma_\rho}](p) = \Delta[\sigma_\rho](p) = 0$, $c_1 = c_2$. Let $M_1 \xrightarrow{\sigma^1_\rho} M_2 \xrightarrow{\sigma^2_\rho} M_3 \xrightarrow{\underline{\sigma_\rho}} M_4 \xrightarrow{\sigma^3_\rho} M_5 \xrightarrow{\sigma^4_\rho} M_6$. Let δ_ρ be the sub-word of $\sigma^2_\rho\underline{\sigma_\rho}\sigma^3_\rho$ consisting of all the transition occurrences having an arc to/from p. Since $M_2(p) = c_1 = c_2 = M_5(p)$ is the minimum number of tokens in p among all intermediate vectors occurring between M_2 and M_5, $\Delta[\delta_\rho](p) = 0$ and δ_ρ is safe for transfer. Transfer δ_ρ from p to p_I, where I is the VC-interface of p. Perform similar transfers for all $1 \leq \rho \leq e$ and independent places $p \in I \setminus Y$ such that $\Delta[\underline{\sigma_\rho}](p) = 0$ and $\underline{\sigma_\rho}$ occurs within the caring zone of p.

Consider some $1 \leq \rho \leq e$ and an independent place $p \in I \setminus Y$ such that $\Delta[\underline{\sigma_\rho}](p) > 0$ and $\underline{\sigma_\rho}$ occurs within the caring zone of p. Let $M \xrightarrow{\sigma'_1\sigma_1\underline{\sigma_1}\sigma_1\cdots\sigma'_{\rho-1}} M_1 \xrightarrow{\sigma_\rho} M_3 \xrightarrow{\underline{\sigma_\rho}} M_4$. Let $c_1 =$

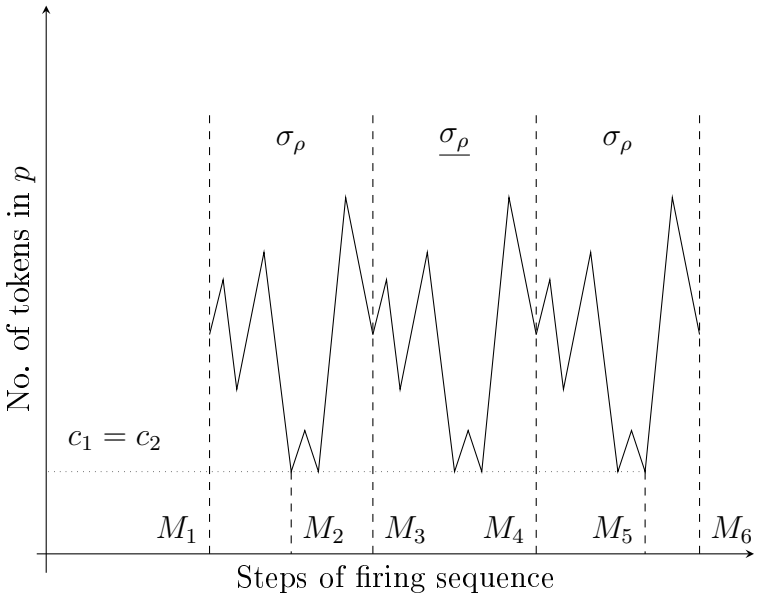


Figure 7.3: Illustration for proof of Lemma 7.25

$\min\{M'(p) \mid M'$ occurs between M_1 and $M_3\}$ be the minimum number of tokens in p among all intermediate vectors occurring between M_1 and M_3. Let M_2 be the first intermediate vector between M_1 and M_3 such that $M_2(p) = c_1$. Let $M_1 \xrightarrow{\sigma_\rho^1} M_2 \xrightarrow{\sigma_\rho^2} M_3 \xrightarrow{\underline{\sigma_\rho}} M_4$. Let δ_ρ be the sub-word of $\sigma_\rho^2\underline{\sigma_\rho}$ consisting of all transition occurrences having an arc to/from p. Since $M_2(p) = c_1$ is the minimum number of tokens in p among all intermediate vectors between M_1 and M_4, δ_ρ is safe for transfer. Transfer δ_ρ to p_I. To ensure that after this transfer, number of tokens in p is pumped up during the pumping portion under consideration, identify the last transition in δ_ρ that adds tokens to p and transfer it back to p. Since $\Delta[\underline{\sigma_\rho}](p) \geq 2W$, this last back transfer will not violate any property of the pumping sequence. Perform this transfer and back transfer for all $1 \leq \rho \leq e$ and independent places $p \in I \setminus Y$ such that $\Delta[\underline{\sigma_\rho}](p) > 0$ and $\underline{\sigma_\rho}$ occurs within the caring zone of p.

Now, we have a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence with the following properties:

1. For all $1 \leq \rho \leq e$ and independent places $p \in I \setminus Y$ such that $\Delta[\underline{\sigma_\rho}](p) = 0$ and $\underline{\sigma_\rho}$ occurs within the caring zone of p, no transition in $\underline{\sigma_\rho}$ has an arc to/from p.

2. For all $1 \leq \rho \leq e$ and independent places $p \in I \setminus Y$ such that $\Delta[\underline{\sigma_\rho}](p) > 0$ and $\underline{\sigma_\rho}$ occurs within the caring zone of p, there is only one transition in $\underline{\sigma_\rho}$ that has an arc to/from p and this transition adds some tokens to p.

Consider an independent place $p \in I \setminus Y$ of some VC-interface I. Let M' be the last intermediate vector in the caring zone of p such that $M'(p)$ is the minimum number of tokens in p among all intermediate vectors in the caring zone of p.

Case 1: $M'(p) \geq M(p)$. In this case, the number of tokens in p does not come below $M(p)$ at all. Let δ_p be the sub-word of the pumping sequence consisting of all transitions occurrences within the caring zone of p that have an arc to/from p, except the last such transition. Transfer δ_p to p_v.

Case 2: $M'(p) < M(p)$. Invoking truncation lemma with $c = M(p) + W$, we identify sub-words between M and M' and transfer them to p_I so that in any intermediate vector within the caring zone of p, p has at most $U + W + W^2 + W^3$ tokens. Note that none of the

sub-words transferred will involve any transition in pumping portions due to the property we have ensured above.

Due to the property we have ensured above, if for some place $p \in I \setminus Y$, there is some $\underline{\sigma_j}$ occurring within the caring zone of p with $\Delta[\underline{\sigma_j}](p) > 0$, it remains so after any of the transfers above. For every independent place $p \in \overline{I} \setminus Y$, we identify and transfer sub-words to p_I based on one of the above two cases. Finally, we end up with a Y-neglecting weakly M, Q, ω-enabled X-pumping sequence such that every independent place $p \in I \setminus Y$ has at most U' tokens in all intermediate vectors belonging to the caring zone of p. □

We will now combine results of previous lemmas to give a ParaPspace upper bound for model checking β formulas.

Theorem 7.26. *With the vertex cover number k and maximum arc weight W as parameters, β formulas of the logic given in the beginning of this section can be model checked in* ParaPspace.

Proof. From Lemma 7.14, model checking β formulas is equivalent to checking the presence of X-pumping sequences for some X. The choice of X can be done non-deterministically in the algorithm. From Lemma 7.16, checking the presence of X-pumping sequences is equivalent to checking the presence of $\emptyset$-neglecting weakly M_0, P, ω-enabled X-pumping sequences. Setting $U' = U + W^2 + W^3$ in Definition 7.20, Lemma 7.25 implies that if there is a $\emptyset$-neglecting weakly M_0, P, ω-enabled X-pumping sequence, there is one of length at most $\ell_2(k', 1)$.

A non-deterministic Turing machine can test for the presence of a weakly enabled pumping sequence by guessing and verifying a sequence of length at most $\ell_2(k', 1)$. By Lemma 7.24, the memory needed by such a Turing machine is $\mathcal{O}(m \log |M_0| + m + \log W + (1+c'k'^3)^{k'} \log k' \log m + poly_1(c'^{k'} k'^{3k'}) \log W + poly_2(c'^{k'} k'^{3k'}) \log(U'k'W))$, or $\mathcal{O}(m \log |M_0| + m + poly(c'^{3k'} k'^{3k'}) \log(U'k'mW))$ for some polynomial $poly$. An application of Savitch's theorem now gives us the required ParaPspace algorithm. □

7.4 Open Problems

As mentioned in the beginning of this chapter, feedback vertex set is a set of vertices whose removal from a graph removes all cycles. Every vertex cover is a feedback vertex set but the converse is not true. The parameterized complexity of problems studied in this chapter with the size of the minimum feedback vertex set is open.

Chapter 8

Conclusion

Following is a list of parameterized complexity results obtained for problems with classical exponential time algorithms (the problems are either PSPACE-complete, NP-complete or exponential time-complete). The entries marked ? are open and for those marked as Not Applicable, the parameter is not well defined for the problem.

	Treewidth	Treewidth and modality depth	Vertex cover number
Reachability, coverability in 1-safe nets	W[1]-hard	Not Applicable	?
LTL model checking on 1-safe Petri nets	W[1]-hard, even when formula size is constant	Not Applicable	FPT with formula size as additional parameter
CTL model checking on 1-safe Petri nets	W[1]-hard, even when formula size is constant	Not Applicable	?
LTL and CTL CNF satisfiability	W[1]-hard	W[1]-hard	?
Modal CNF satisfiability in transitive models	W[1]-hard	W[1]-hard	?
Modal CNF satisfiability in general models	Not FPT unless PTIME = PSPACE	FPT	?
Modal CNF satisfiability in Euclidean models	FPT	FPT	FPT

Following is a list of the parameterized complexity results for EXPSPACE-complete problems, obtained in this thesis.

	Benefit depth	Vertex cover number
Coverability	PARAPSPACE	PARAPSPACE
Boundedness	PARAPSPACE	PARAPSPACE
Model checking logic of counting properties (section 6.3)	PARAPSPACE with nesting depth of **EF** as constant	PARAPSPACE with nesting depth of **EF** as constant

Appendix A

Brief Introduction to Parameterized Complexity

Let Σ be a finite alphabet in which instances $I \in \Sigma^*$ of a problem are specified. A problem is specified through its set $\Pi \subseteq \Sigma^*$ of YES instances, with all other instances assumed to be NO instances. The complexity of a problem is stated in terms of the amount of resources—space, time—needed by any algorithm solving it, measured as a function of the size $|I|$ of the problem instance. In parameterized complexity, introduced by Downey and Fellows [26], the dependence of resources needed is also measured in terms of a parameter $\kappa(I)$ of the input, which is usually less than the input size $|I|$. A parameterized problem is said to be **fixed parameter tractable** (FPT) if it can be solved by an algorithm with running time $f(\kappa(I))poly(|I|)$ where f is some computable function and *poly* is a polynomial. (Similarly, a PARAPSPACE algorithm [36] is one that runs in space $f(\kappa(I))poly(|I|)$.)

For example, consider the following p-AUTOMATON-MODEL-CHECK problem:

p-AUTOMATON-MODEL-CHECK	
Instance:	A finite state automaton $\mathcal{A}$ and a MSO sentence ϕ.
Parameter:	The size $\|\phi\|$ of ϕ.
Problem:	Decide whether all strings accepted by $\mathcal{A}$ satisfy ϕ.

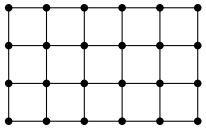
This problem is FPT, by Büchi, Elgot, Trakhtenbrot theorem [12].

There is a parameterized complexity class W[1], lowest in a hierarchy of intractable classes called the W-hierarchy [26] (similar to the polynomial time hierarchy). A parameterized problem complete for W[1] is to decide if there is an accepting computation of at most k steps in a given non-deterministic Turing machine, where the parameter is k [26]. It is widely believed that parameterized problems hard for W[1] are not FPT. To prove that a problem is hard for a parameterized complexity class, we have to give a **parameterized reduction** from a problem already known to be hard to our problem. A parameterized reduction from (Π, κ) to (Π', κ') is an algorithm A that maps problem instances in (resp. outside) Π to problem instances in (resp. outside) Π'. There must be computable functions f and g and a polynomial p such that the algorithm A on input I terminates in time $f(\kappa(I))p(|I|)$ and $\kappa'(A(I)) \leq g(\kappa(I))$, where $A(I)$ is the problem instance output by A. There are many other classes in the intractability hierarchy, the ones used in this thesis are mentioned below.

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \cdots \subseteq \text{AW[1]} \subseteq \text{AW[SAT]} \subseteq \text{AW[P]} \subseteq \text{XP}$$

The class XP contains all problems that can be solved by algorithms with running time $\mathcal{O}(|I|^{f(\kappa(I))})$ for some function f. The inclusion between FPT and XP is known to be strict, while strictness of other inclusions are not known.

From the Definition 1.1, it is clear that pathwidth is at least as large as treewidth and any problem that is W[1]-hard with pathwidth as parameter is also W[1]-hard with treewidth as parameter. An intuitive way to understand the role of treewidth on complexity of problems is through its relation to grids. A grid is a graph of the form shown below.



A large grid has large treewidth. A graph of small treewidth can not have a large grid embedded within it (formally, the graph can not have large grid minors; see [26, Section 7.1] for details). Recall that NP-complete problems can be reduced to coloring problems on grids. A fundamental result by Courcelle [16] shows that graphs of small treewidth are easier to handle algorithmically: checking whether a graph satisfies a MSO sentence is FPT if the graph's treewidth and the MSO sentence's length are parameters. In the context of Chapter 4, the state space of a concurrent system can be considered a graph. However, due to the state explosion problem, the state space can be very large. Instead, we impose treewidth restriction on a compact representation of the large state space — a 1-safe Petri net. Note also that we are not model checking the state space itself but only the language of words generated by the Petri net.

Bibliography

[1] A. Achilleos, M. Lampis, and V. Mitsou. Parameterized modal satisfiability. In *International Colloquium on Automata, Languages and Programming*, volume 6199 of *LNCS*, pages 369–380, 2010.

[2] I. Adler and M. Weyer. Tree-width for first order formulae. In *Computer Science Logic*, volume 5771 of *LNCS*, pages 71–85, 2009.

[3] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12:308–340, 1991.

[4] M. F. Atig and P. Habermehl. On Yen's path logic for Petri nets. In *Workshop on Reachability Problems*, volume 5797 of *LNCS*, pages 51–63, 2009.

[5] M.F. Atig, A. Bouajjani, and S. Qadeer. Context-bounded analysis for concurrent programs with dynamic creation of threads. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 5505 of *LNCS*, pages 107–123, 2009.

[6] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[7] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal of Computing*, 25(6):1305–1317, 1996.

[8] H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996.

[9] M. Bojanczyk, A. Muscholl, T. Schwentick, L. Segoufin, and C. David. Two-variable logic on words with data. In *Logic in Computer Science*, pages 7–16, 2006.

[10] I. Borosh and L. Treybig. Bounds on positive integral solutions of linear diophantine equations. *Proceedings of the American Mathematical Society*, 55(2):299–304, March 1976.

[11] D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, 1983.

[12] J. R. Büchi. On a decision method in restricted second-order arithmetic. In *Logic, Methodology, Philosophy and Science*, pages 1–11. Stanford University Press, 1962.

[13] H. Chen. Quantified constraint satisfaction and bounded treewidth. In *European Conference on Artificial Intelligence*, pages 161–165. IOS Press, 2004.

[14] A. Cheng, J. Esparza, and J. Palsberg. Complexity results for 1-safe nets. *Theoretical Computer Science*, 147(1-2):117–136, 1995.

[15] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[16] B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.

[17] B. Courcelle. The monadic second-order logic of graphs III: tree-decompositions, minors and complexity issues. *Theoretical Informatics and Applications*, 26:257–286, 1992.

[18] V. Dalmau, P. G. Kolaitis, and M. Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *International Conference on Principles and Practice of Constraint Programming*, volume 2470 of *LNCS*, pages 310–326, 2002.

[19] S. Demri. On selective unboundedness of VASS. In *International Workshop on Verification of Infinite-State Systems*, pages 1–15, 2010. http://dx.doi.org/10.4204/EPTCS.39.1.

[20] S. Demri, M. Jurdzinski, O. Lachish, and R. Lazić. The covering and boundedness problems for branching vector addition systems. In *Foundations of Software Technology and Theoretical Computer Science*, volume 4 of *LIPIcs*, pages 181–192, 2009. http://drops.dagstuhl.de/opus/volltexte/2009/2317.

[21] S. Demri, F. Laroussinie, and P. Schnoebelen. A parametric analysis of the state-explosion problem in model checking. *Journal of Computer and System Sciences*, 72(4):547–575, 2006.

[22] S. Demri and R. Lazić. LTL with freeze quantifier and register automata. *ACM Transactions on Computational Logic*, 10(3):1–30, 2009.

[23] J. Desel and J. Esparza. *Free choice Petri nets*. Cambridge University Press, 1995.

[24] J. Desel and W. Reisig. *Place/transition Petri nets*, volume 1491 of *LNCS*, pages 122–173. 1998.

[25] R. Downey. Parameterized complexity for the skeptic. In *IEEE Annual Conference on Computational Complexity*, pages 147–170, 2003.

[26] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer-Verlag, 1999.

[27] R. G. Downey, M. R. Fellows, and U. Stege. Parameterized complexity: A framework for systematically confronting computational intractability. In *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, volume 49 of *DIMACS*, pages 49–100. 1999.

[28] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. In *Symposium on Theory of Computing*, pages 169–180, 1982.

[29] P. Enjalbert and L. F. del Cerro. Modal resolution in clausal form. *Theoretical Computer Science*, 65(1):1–33, 1989.

[30] J. Esparza. *Decidability and complexity of Petri net problems — An introduction*, volume 1491 of *LNCS*, pages 374–428. 1998.

[31] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[32] M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, S. Szeider, and C. Thomassen. On the complexity of some colorful problems parameterized by treewidth. In *International Conference on Combinatorial Optimization and Applications*, volume 4616 of *LNCS*, pages 366–377, 2007.

[33] M. R. Fellows, D. Lokshtanov, N. Misra, F. A. Rosamond, and S. Saurabh. Graph layout problems parameterized by vertex cover. In *International Symposium on Algorithms and Computation*, volume 5369 of *LNCS*, pages 294–305, 2008.

[34] E. Fischer, J.A. Makowsky, and E.V. Ravve. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discrete Applied Mathematics*, 156(4):511–529, 2008.

[35] M. J. Fischer and R. E. Ladner. Propositional modal logic of programs. In *Symposium on Theory of Computing*, pages 286–294, 1977.

[36] J. Flum and M. Grohe. Describing parameterized complexity classes. *Information and Computation*, 187(2):291–319, 2003.

[37] J. Flum and M. Grohe. *Parameterized complexity theory*. Springer, 2006.

[38] A. Frank and E. Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7:49–65, January 1987.

[39] M. Frick and M. Grohe. The complexity of first-order and monadic second-order logic revisited. *Annals of Pure and Applied Logic*, 130(1-3):3–31, 2004.

[40] S. Göller, C. Haase, J. Ouaknine, and J. Worrell. Model checking succinct and parametric one-counter automata. In *International Colloquium on Automata, Languages and Programming*, volume 6199 of *LNCS*, pages 575–586, 2010.

[41] E. Grädel. Why are modal logics so robustly decidable? In *Current trends in theoretical computer science*, pages 393–408. World Scientific Publishing Co., 2001.

[42] E. Grädel and M. Otto. On logics with two variables. *Theoretical Computer Science*, 224:73–113, 1999.

[43] M. Grohe. The structure of tractable constraint satisfaction problems. In *Mathematical Foundations of Computer Science*, volume 4162 of *LNCS*, pages 58–72, 2006.

[44] C. Haase, S. Kreutzer, J. Ouaknine, and J. Worrell. Reachability in succinct and parametric one-counter automata. In *International Conference on Concurrency Theory*, volume 5710 of *LNCS*, pages 369–383, 2009.

[45] P. Habermehl. On the complexity of the linear-time μ-calculus for Petri-nets. In *International Conference on Application and Theory of Petri Nets and Other Models of Concurrency*, volume 1248 of *LNCS*, pages 102–116, 1997.

[46] J. Y. Halpern. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75(2):361–372, 1995.

[47] J. Y. Halpern and Y. O. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992.

[48] J. Y. Halpern and L. C. Rêgo. Characterizing the NP-Pspace gap in the satisfiability problem for modal logic. *Journal of Logic and Computation*, 17(4):795–806, 2007.

[49] E. Hemaspaandra and H. Schnoor. On the complexity of elementary modal logics. In *Symposium on Theoretical Aspects of Computer Science*, volume 1 of *LIPIcs*, pages 349–360, 2008.

[50] A. Herzig and J. Mengin. Uniform interpolation by resolution in modal logic. In *European Conference on Logics in Artificial Intelligence*, volume 5293 of *LNCS*, pages 219–231, 2008.

[51] J.E. Hopcroft and J. Ullman. *Introduction to automata theory, languages and computation.* Addison-Wesley, 1979.

[52] R. Howell, L.E. Rosier, and H.-C. Yen. A taxonomy of fairness and temporal logic problems for petri nets. *Theoretical Computer Science*, 82(2):341–372, 1991.

[53] U. Hustadt and R. A. Schmidt. An empirical analysis of modal theorem provers. *Journal of Applied Non-Classical Logics*, 9(4), 1999.

[54] R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12:415–440, August 1987.

[55] R.M. Karp and R.E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, May 1969.

[56] K. M. Kavi, A. Moshtaghi, and D-J. Chen. Modeling multithreaded applications using Petri nets. *International Journal of Parallel Programming*, 30(5):353–371, 2002.

[57] Y. Kazakov and I. Pratt-Hartmann. A note on the complexity of the satisfiability problem for graded modal logics. In *Logic in Computer Science*, pages 407–416, 2009.

[58] S.R. Kosaraju. Decidability of reachability in vector addition systems. In *Symposium on Theory of Computing*, pages 267–281, 1982.

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

[60] R. E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal of Computing*, 6(3):467–480, 1977.

[61] P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for AC-like equational theories with homomorphisms. In *International Conference on Rewriting Techniques and Applications*, volume 3467 of *LNCS*, pages 308–322, 2005. Full version at http://www.lsv.ens-cachan.fr/Publis/RAPPORTS%5FLSV/PS/rr-lsv-2004-16.rr.ps.

[62] J.L. Lambert. A structure to decide reachability in Petri nets. *Theoretical Computer Science*, 99(1):79–104, June 1992.

[63] R. Lazic. The reachability problem for branching vector addition systems requires doubly-exponential space. *Information Processing Letters*, 110(17):740–745, 2010.

[64] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[65] J. Leroux. Vector addition system reachability problem: a short self-contained proof. In *Principles of Programming Languages*, pages 307–316, 2011.

[66] R. Lipton. The reachability problem requires exponential space. Technical Report 62, 1975. Yale university.

[67] P. Madhusudan and G. Parlato. The tree width of auxiliary storage. In *Principles of Programming Languages*, pages 283–294, 2011.

[68] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems.* Springer-Verlag New York, Inc., 1992.

[69] A. Manuel. Two variables and two successors. In *Mathematical Foundations of Computer Science*, volume 6281 of *LNCS*, pages 513–524, 2010.

[70] D. Marx. Can you beat treewidth? In *Foundations of Computer Science*, pages 169–179, 2007.

[71] E.W. Mayr. An algorithm for the general Petri net reachability problem. In *Symposium on Theory of Computing*, pages 238–246, 1981.

[72] K. McAloon. Petri nets and large finite sets. *Theoretical Computer Science*, 32:173–183, 1984.

[73] R. K. Meyer. *Topics in modal and many-valued logic.* PhD thesis, University of Pittsburgh, Pennsylvania, 1966.

[74] R. K. Meyer and S. Giambrone. R_+ is contained in T_+. *Bulletin of the Section of Logic, Polish Academy of Sciences*, 9:30–32, 1980.

[75] M. Minsky. *Computation: Finite and Infinite Machines.* Prentice Hall, 1967.

[76] L. A. Nguyen. *On the complexity of fragments of modal logics*, volume 5 of *Advances in Modal logic*, pages 249–268. 2005.

[77] G. Pan and M. Y. Vardi. Optimizing a BDD-based modal solver. In *International Conference on Automated Deduction*, volume 2741 of *LNCS*, pages 75–89. 2003.

[78] G. Pan and M. Y. Vardi. Fixed-parameter hierarchies inside pspace. In *Logic in Computer Science*, pages 27–36, 2006.

[79] M. Parigot and E. Pelz. A logical approach of petri net languages. In *Foundations of Software Technology and Theoretical Computer Science*, volume 39 of *Theoretical Computer Science*, pages 155–169, 1985.

[80] C.A. Petri. *Kommunikation mit Automaten.* PhD thesis, Institut für Instrumentelle Mathematik, 1962.

[81] V. R. Pratt. Application of modal logic to programming. *Studia Logica*, 39(2-3):257–274, 1980.

[82] M. Praveen. Complexity of the reachability problem in subclasses of Petri nets. Master's thesis, Homi Bhabha National Institute, 2008.

[83] C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6:223–231, 1978.

[84] J. H. Reif and A. P. Sistla. A multiprocess network logic with temporal and spatial modalities. In *International Colloquium on Automata, Languages and Programming*, volume 154 of *LNCS*, pages 629–639, 1983.

[85] K. Reinhardt. Reachability in Petri nets with inhibitor arcs. In *Workshop on Reachability Problems*, volume 223 of *ENTCS*, pages 239–264, 2008.

[86] C. Reutenauer. *The mathematics of Petri-nets*. Masson and Prentice Hall International (UK) Ltd, 1990. Translated by I. Craig.

[87] L.E. Rosier and H.-C. Yen. A multiparameter analysis of the boundedness problem for vector addition systems. *Journal of Computer and System Sciences*, 32(1):105–135, 1986.

[88] M. Samer and S. Szeider. Constraint satisfaction with bounded treewidth revisited. *Journal of Computer and System Sciences*, 76(2):103–114, 2010.

[89] T. Schwentick and T. Zeume. Two-variable logic with two order relations - (extended abstract). In *Computer Science Logic*, volume 6247 of *LNCS*, pages 499–513, 2010.

[90] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32:733–749, 1985.

[91] B. ten Cate. A note on the expressibility problem for modal logics and star-free regular expressions. *Information Processing Letters*, 109(10):509–513, 2009.

[92] P. B. Thistlewaite, M. A. McRobbie, and R. K. Meyer. *Automated theorem-proving in non-classical logics*. Pitman, 1988.

[93] M. Thorup. All structured programs have small tree width and good register allocation. *Information and Computation*, 142(2):159–181, 1998.

[94] A. S. Troelstra. *Lectures on linear logic*, volume 29 of *CSLI Lectures Notes*. 1992.

[95] A. Urquhart. The complexity of decision procedures in relevance logic. In *Truth or consequences: Essays in honour of Nuel Belnap*, pages 61–76. Kluwer, 1990.

[96] A. Urquhart. The complexity of decision procedures in relevance logic II. *The Journal of Symbolic Logic*, 64(4):1774–1802, 1999.

[97] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Proceedings of the VIII Banff Higher order workshop conference on Logics for concurrency : structure versus automata*, volume 1043 of *LNCS*, pages 238–266, 1996.

[98] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Logic in Computer Science*, pages 332–344, 1986.

[99] H.-C. Yen. A unified approach for deciding the existence of certain petri net paths. *Information and Computation*, 96(1):119–137, 1992.