FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

The Parameterized Complexity of Nonmonotonic Reasoning

PhD THESIS

submitted in partial fulfillment of the requirements of

Doctor of Technical Sciences

within the

Vienna PhD School of Informatics

by

Dipl.-Ing. Stefan Rümmele
Registration Number 0325665

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Univ.Prof. Mag.rer.nat. Dr.techn. Reinhard Pichler
Assistance: Privatdoz. Dipl.-Ing. Dr.techn. Stefan Woltran

Wien, 04.07.2012

(Signature of Author) (Signature of Advisor)

Declaration of Authorship

Dipl.-Ing. Stefan Rümmele
1020 Wien

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

Wien, 04.07.2012

_____

(Signature of Author)

Acknowledgments

First and foremost I want to thank my advisors Reinhard Pichler and Stefan Woltran. I am deeply grateful for their guidance, support and inspiration. I especially want to thank them for giving me the freedom to pursue my own research interests. I feel well prepared for the upcoming challenges of a research career.

Furthermore, I want to thank my coauthors Michael Fellows, Martin Lackner, Michael Morak, Nysret Musliu, Andreas Pfandler, Reinhard Pichler, Frances Rosamond, Stefan Szeider, and Stefan Woltran for the joint work on our papers upon which this thesis is based.

A special thank goes to Mike and Fran for hosting Andreas and myself in Australia, during our collaboration. Working outside the office with nothing but a flip-chart, pen, and paper definitely showed me a new and very fruitful way of doing research. The surfing was great as well!

I want to express my thank to my fellow PhD students at DBAI, Martin Lackner, Andreas Pfandler, Emanuel Sallinger, Sebastian Skritek, and Vadim Savenkov for making my workplace such a great place to be. I am especially in their debt for all the valuable comments and remarks I got on early drafts of this thesis.

Finally, I want to thank my family and friends for their patience and understanding whenever I had to prioritize my work over them. I owe my deepest gratitude to my parents Renate and Herbert Rümmele. Their continuous support made it much easier for me to focus on my research and concentrate on pursuing my goals. To Natasha, you have been vital to me as a source of energy by providing me necessary distraction as well as motivating me to work even harder.

I dedicate this thesis to Natasha and my family.

This work was supported by the Austrian Science Fund (FWF): P20704-N18.

Abstract

In our daily life we encounter situations where already drawn conclusions can turn out to be invalid because new information becomes available which we did not know before. Human reasoning is not only capable of dealing with such nonmonotonic situations, but also is this kind of reasoning permanently performed by humans. Hence, it is no surprise that lots of research in *artificial intelligence* (AI) and *knowledge representation* (KR) is conducted in order to study so-called nonmonotonic reasoning formalisms.

Examples of such formalisms are *belief revision*, *answer-set programming*, and *(propositional) abduction*. A big obstacle that limits the practical applicability of computational nonmonotonic reasoning is the high complexity of these tasks. A promising approach to dealing with high computational complexity is to study what kind of properties those problem instances have that can be solved efficiently. Such tractable fragments are not restricted to syntactical limitations, such as Horn formulas instead of general propositional formulas. We are more interested in structural fragments since they often tell us something about the (hidden) structure of certain problem instances.

The framework within which the search for such structural fragments can be conducted is called *parameterized complexity theory*. Thereby a multivariate complexity analysis of the problem is performed, where the input size is just one dimension. The other dimensions that are studied are called parameters.

To overcome the high complexity of nonmonotonic reasoning problems, we seek parameters or combinations of parameters such that the computational hardness of the problem can be confined in them. This means that if we consider fragments of problem instances with sufficiently small parameter values, we obtain new tractable fragments.

In this thesis we will study the three above mentioned nonmonotonic reasoning formalisms. We initiate the research of parameterized complexity of belief revision as well as propositional abduction. Furthermore, we significantly advance the state-of-the-art of parameterized complexity of answer-set programming.

Abstract

In unserem Alltag erleben wir Situationen, in denen sich bereits getätigte Schlussfolgerungen als ungültig erweisen, weil neue Informationen verfügbar werden, die wir vorher nicht kannten. Menschliches Denken ist nicht nur in der Lage mit solchen nichtmonotonen Situationen umzugehen, sondern tatsächlich wird diese Art des logischen Denkens permanent von Menschen durchgeführt. Daher ist es nicht verwunderlich, dass sich viel Forschung in den Bereichen der *künstlichen Intelligenz* (KI) und der *Wissensrepräsentation* mit Formalismen zu sogenanntem nichtmonotonen Schließen auseinandersetzt.

Beispiele für solche Formalismen sind *Wissensrevision* (Belief Revision), *Antwortmengenprogrammierung* (Answer-Set Programming) und *(aussagenlogische) Abduktion* (propositional Abduction). Eine große Hürde, welche die praktische Anwendbarkeit von maschinellem nichtmonotonen Schließen begrenzt, ist die hohe Komplexität dieser Aufgaben. Ein vielversprechender Ansatz im Umgang mit Problemen hoher Komplexität ist zu untersuchen, welche Eigenschaften jene Probleminstanzen haben, die eine effiziente Berechnung ihrer Lösung erlauben. Solche Fragmente sind nicht auf syntaktische Kriterien beschränkt, wie sie zum Beispiel entstehen wenn nur Horn-Formeln anstatt uneingeschränkte Formeln der Aussagenlogik betrachtet werden. Stattdessen sind wir an strukturellen Fragmenten interessiert, da man aus diesen oft etwas über den strukturellen Aufbau bestimmter Probleminstanzen lernen kann.

Die Methodik, mit welcher die Suche nach solchen strukturellen Fragmenten durchgeführt werden kann, wird als *parametrisierte Komplexitätstheorie* bezeichnet. Diese ermöglicht eine multivariate Analyse der Komplexität des Problems. Dabei stellt die Menge der als Eingabe zur Verfügung gestellten Daten nur eine der Dimensionen dar. Die anderen Dimensionen die untersucht werden, nennt man Parameter.

Um die hohe Komplexität der Probleme aus dem Bereich des nichtmonotonen Schließens zu überwinden, suchen wir Parameter oder Kombinationen von Parametern, so dass die komplette Härte des Problems in diesen beschränkt werden kann. Das bedeutet dann, dass Fragmente von Probleminstanzen mit hinreichend niedrigen Parameterwerten tatsächlich effizient gelöst werden können.

In dieser Arbeit untersuchen wir die drei oben genannten Formalismen für maschinelles nichtmonotones Schließen. Wir initiieren die Erforschung der parametrisierten Komplexität der Wissensrevision sowie der aussagenlogischen Abduktion. Darüber hinaus tragen wir erheblich zur Erweiterung des Wissensstandes bezüglich der parametrisierten Komplexität der Antwortmengenprogrammierung bei.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Darth Vader: Obi-Wan never told you what happened to your father.
Luke Skywalker: He told me enough! He told me you killed him!
Darth Vader: No. I am your father.
– Star Wars Episode V: The Empire Strikes Back

Although the dialog above is a fictive situation, it demonstrates well that *nonmonotonic reasoning* is something that can be observed in the real world. Before talking to Darth Vader, Luke believed that his father is dead. During the conversation he acquires new information that allows him to actually come to the opposite conclusion: his father is still alive. Formalisms to express reasoning in which already drawn conclusions can become invalid later are called nonmonotonic. Human reasoning is not only capable of dealing with such nonmonotonic situations, but also is this kind of reasoning permanently performed by humans. Hence, it is no surprise that a lot of research in *artificial intelligence* (AI) and *knowledge representation* (KR) has been done investigating nonmonotonic reasoning formalisms.

An example of such a formalism is *belief revision* [90]. Thereby a knowledge base is given as a set of propositional formulas and additionally some new information is provided as a propositional formula. This new information may contradict (part of) the knowledge base, and the task is to revise the knowledge in order to incorporate the new information and keep the changes to the existing knowledge base minimal. The above situation can be expressed in this formalism as follows. The original knowledge base of Luke contains

$$\text{father\_is\_dead} \wedge \text{vader\_lives} \wedge (\text{vader\_lives} \wedge \text{vader\_is\_father} \rightarrow \neg\text{father\_is\_dead}).$$

The new information that has to be added is the fact vader_is_father. This leads to a contradiction since Luke can now derive

$$\text{father\_is\_dead} \wedge \neg\text{father\_is\_dead}.$$

One possible solution is to remove the belief $\mathrm{father\_is\_dead}$ from the knowledge base. Generally knowledge is continually evolving. Hence, there is a constant need to be able to revise knowledge bases as new information is received. Belief revision has been applied in a variety of domains, e.g. in software engineering and marketing research [114].

Another example of a nonmonotonic reasoning formalism is *answer-set programming* (also called A-prolog) [15]. It is a form of descriptive logic programming where the problem to be solved is encoded by facts, rules, and constraints. The programs are evaluated under the *stable model semantics* and the outcome, the so-called *answer-sets*, correspond to the solutions of the problem. Answer-set programming allows default negation, that means if we cannot derive some specific fact then we assume that it is false. This is a source of nonmonotonicity since adding more facts to the program at a later point might invalidate previous answers. The following answer-set program models a situation where the spaceship "Millennium Falcon" would be captured by the imperium if its hyperdrive is faulty.

$$\mathrm{jump\_to\_lightspeed} \leftarrow \neg \mathrm{hyperdrive\_faulty}.$$
$$\mathrm{captured\_by\_imperium} \leftarrow \neg \mathrm{jump\_to\_lightspeed}.$$

The sole answer set is $\{\mathrm{jump\_to\_lightspeed}\}$. But if the fact

$$\mathrm{hyperdrive\_faulty}.$$

is added then the previous solution can no longer be derived. Instead, the new answer set is $\{\mathrm{hyperdrive\_faulty}, \mathrm{captured\_by\_imperium}\}$. Answer-set programming is a very prominent form of declarative problem solving and was applied for example in the development of a system for controlling some of the functions of the NASA space shuttle [85].

A third example is (propositional) *abductive reasoning* where propositional formulas describe the knowledge that an agent has about (some fragment of) the world, see for example [31]. The agent is then confronted with some new information, called the *manifestation* and has to choose among a set of so called *hypotheses* those that best explain the manifestation and are consistent with her knowledge. When she receives further manifestations she might be forced to drop old hypotheses in favor of others. Consider for example a knowledge base about spaceship battles that contains

$$\mathrm{surrounded\_by\_enemies} \rightarrow \mathrm{its\_a\_trap}.$$

Assume one hears now the information $\mathrm{its\_a\_trap}$, called the manifestation. In order to search for the cause of this, one has to select a plausible hypothesis. A solution would be to select $\mathrm{surrounded\_by\_enemies}$. In contrast to deductive reasoning, abductive reasoning is a method for reverse inference. This means one is interested in explaining observed behavior by finding appropriate causes. It is widely believed that humans use abduction in their reasoning when searching for diagnostic explanations. Abductive reasoning has applications in areas such as system diagnosis [78] and medical diagnosis [95].

Nonmonotonic reasoning is natural and unavoidable in the real world, and the formalisms presented above as well as many more frameworks bring the necessary tools to model this kind of reasoning in a computational setting. But there is a big obstacle that limits the practical applicability of computational nonmonotonic reasoning. It is the high complexity of most of

these tasks. Many of the corresponding computational problems are in general hard for classes on the second level of the polynomial hierarchy [30, 31, 32]. That means they are presumably even harder than NP-complete problems. But even if these formalisms are restricted to certain fragments of propositional logic, like Horn formulas, many of the computational problems are still intractable. For example deciding whether a Horn abduction instance has a solution is NP-complete [103], and belief revision for Horn knowledge bases using Satoh's revision operator [102] is coNP-complete [30].

However, this is not the end of the story. In practice there are many very successful solvers for answer-set programs. This is consistent with the situation of SAT solvers. Nowadays the latter are able to find solutions to propositional satisfiability instances with hundreds of thousands of variables and millions of clauses within a few minutes [54] despite the fact that the problem is NP-complete. So it seems as if there exists a gap between what theory tells us and what one is able to do in practice. A promising approach to close this gap is to study what properties those instances have that that can be solved efficiently. From that, one might be able to identify certain fragments of the problem that are tractable. Those fragments are not restricted to syntactical limitations, like Horn formulas mentioned above. Instead we will call them structural fragments since they often tell us something about the (hidden) structure of certain problem instances.

The framework within which the search for such structural fragments can be conducted is called *parameterized complexity theory* [24]. It is based on the observation that many computationally hard problems become tractable if some parameter that represents a structural aspect of the problem instance is small. Classical complexity theory studies the hardness of a problem only with respect to the input size and ignores all other properties. In contrast, parameterized complexity theory performs a multivariate analysis of the problem where the input size is just one dimension. The other studied dimensions are the so-called parameters. Informally speaking, a parameter can be anything quantifiable.

Let us for example take a look at the problem of deciding the satisfiability of a propositional formula. Possible parameters are the number of variables, the number of clauses, the maximum number of occurrences of a single variable, the size of the largest clause, and many more. Of course some parameters might also turn out to be not useful for identifying tractable fragments. For example, the satisfiability problem is still NP-complete even when restricted to clauses of size at most 3. On the other hand Yamamoto [116] proposed an algorithm which needs $\mathcal{O}(1.234^m \cdot p(n))$ time, where m is the number of clauses and $p(n)$ is a polynomial in the input size. This leads to the tractable fragment of instances with few clauses but possibly many variables.

While the values for the parameters mentioned above can be very easily computed from the problem instances, it is also possible to consider parameters that are more difficult to compute. One might invest time into computing such parameters in order to later save time in the actual computation of the problem. For example, one might transform a given formula into a graph by adding each clause and each variable as a node. If a variable occurs in a clause, we add an edge between the corresponding nodes. Now one can consider various properties of this graph as a parameter of our original problem. One possibility is the *treewidth* of this graph. This is, roughly speaking, a measure of the "tree-likeness" of a graph.

Therefore, to overcome the high complexity of the nonmonotonic reasoning problems men-

tioned above, we want to seek parameters or combinations of parameters that confine the computational hardness of the problem. That means, we will be able to solve these problems efficiently as long as the values of the corresponding parameters are sufficiently small. Formally, we are interested in parameters for which the problems become *fixed-parameter tractable*. A problem is called fixed-parameter tractable if there exists an algorithm with running time of at most

$$f(k) \cdot p(n),$$

where n is the input size, p is some polynomial, k is the parameter value, and f is an arbitrary computable function. Since function f only depends on parameter k, the expression $f(k)$ is constant for any fixed value of k. This means that if we only consider problem instances of parameter values that are bounded by a constant c, we obtain a polynomial time algorithm for arbitrary large input sizes where the degree of this polynomial time algorithm does not depend on c. For details see Chapter 2.

The ultimate goal is to find more and more parameters for which the computationally hard problems become fixed-parameter tractable. The more such tractable fragments are known, the more instances can be solved exactly and efficiently. Note that the parameterized complexity approach gives us exact algorithms. Exact solutions in the area of reasoning are important since in such settings it is not even clear what an approximate solution could be. Another approach for dealing with computational hard problems are heuristics. But there it is either not guaranteed that they will find the optimal solution or it is not guaranteed that they will find it efficiently.

1.2 Problem Statement

In this thesis we will study three different formalisms for nonmonotonic reasoning. We initiate the research of parameterized complexity of belief revision as well as propositional abduction. Furthermore, we significantly advance the state-of-the-art regarding the parameterized complexity of answer-set programming.

The first goal is to identify new tractable fragments for the above-mentioned reasoning problems. To this end we will search for parameters such that these problems become fixed-parameter tractable. This parameterized complexity approach does not exclude the study of structural fragments as well. Indeed, for problems such as abduction that remain intractable even when restricted to syntactical fragments of propositional logic such as Horn formulas, it makes sense to search for tractable structural fragments inside these syntactical fragments.

The second goal is to develop faster exact algorithms for these problems. For example, for the parameter treewidth there exists a meta-theorem due to Courcelle [17] that states for a whole class of problems that they become fixed-parameter tractable when parameterized by treewidth. But often the running time of the algorithm derived from this theorem can be improved drastically when searching for problem specific solutions.

Similar to classical complexity theory, parameterized complexity theory provides a framework to prove that a problem is not fixed-parameter tractable under some complexity theoretic assumptions. These tools come in form of a variety of complexity classes as well as suitable reductions. The most important parameterized intractability classes are arranged in the so-called W-hierarchy. Therefore, the third goal is to prove parameterized intractability for the remaining

parameters of our problems. This will help us to understand the source of hardness of the studied problems. If a problem is still hard when parameterized by a certain parameter then this means that this parameter alone does not confine its hardness.

A technique of parameterized complexity theory, called *kernelization* [24], studies efficient and effective preprocessing of problems. Indeed, such a preprocessing exists if and only if the problem is fixed-parameter tractable. But it turned out that there are problems that can be preprocessed in polynomial time to an instance size that is only polynomial in the parameter value. One says that these problems admit a *polynomial kernel*. In contrast, for other problems it seems that they can only be preprocessed to an instance size that is exponential in the parameter value. Therefore kernelization helps to differentiate between fixed-parameter tractable problems even further. Hence, the fourth goal is to study whether our problems admit a polynomial kernel and can therefore be efficiently preprocessed.

Very recently tools have been developed to show that certain problems do not admit a polynomial kernel under standard complexity theoretic assumptions. As mentioned above, these problems can only be efficiently preprocessed to obtain instance sizes that are exponential in the parameter value. The fifth goal is to prove for the remaining parameterizations of our problems that they cannot be efficiently preprocessed.

1.3 Main Results

The main contributions of this thesis are grouped according to the three studied problems: belief revision, answer-set programming, and abduction.

1.3.1 Belief Revision

We initiate the study of parameterized complexity of belief revision. More precisely, we study the decision problem which asks whether something holds in the revised knowledge base, as well as the enumeration problem where one has to enumerate all possible models of the revised knowledge base. There is no single answer to the question of how to measure minimality of changing a knowledge base. Therefore, many different revision operators have been proposed in the literature. We consider here the ones according to Dalal [20], Satoh [102], and Winslett [115]. We study these problems when parameterized by the treewidth of an associated graph encoding. The main contributions are as follows:

- A novel algorithm based on dynamic programming for deciding the reasoning problem of Dalal's operator in *fixed-parameter linear time* when parameterized by the treewidth of the formulas. Fixed-parameter linear means fixed-parameter tractable with only linear dependency on the input size.

- An extension of the algorithm for the decision problem, such that also the set of all models of the revised knowledge base is computed. In particular, our algorithm works with linear delay if the formulas have bounded treewidth. Linear delay means that the time for outputting the first model as well as the time between outputting two consecutive models is only linear in the input size.

- We show that the reasoning problems for Satoh's and Winslett's operators are definable in monadic second-order logic (MSO) and that the reasoning problem for Dalal's operator can be defined in an extension of MSO following Arnborg, Lagergren, and Seese [4]. We can then conclude from Courcelle's Theorem [17] that Satoh's and Winslett's operators are fixed-parameter tractable with respect to the parameter treewidth. In case of Dalal's operator we obtain an alternative proof of its fixed-parameter tractability, which follows of course also from our dedicated algorithm via dynamic programming.

1.3.2 Answer-Set Programming

For answer-set programming there exists already some basic parameterized complexity analysis, see for example [57, 71]. We extend this research by considering programs with weight constraints (PWCs) and programs with cardinality constraints (PCCs) [82]. PWCs are an extension of answer-set programs where it is possible to assign weights to the occurring literals. This formalism then allows to express constraints that restrict which literals are allowed to be set to true respecting certain weight bounds. PCCs are a special case of PWCs where every literal has weight 1.

We investigate mainly the consistency problem, asking if a given program has a solution. Moreover, we show how these results can be extended to the problems of credulous and skeptical reasoning. Credulous reasoning asks whether a fact holds in some answer-set while skeptical reasoning asks whether a fact holds in every answer-set. We study these problems when parameterized by treewidth together with the largest occurring constraint bounds in the program.

- It is shown in [92] that the consistency problem of PWCs remains NP-complete even if the treewidth of the considered programs is bounded by a constant. Actually this holds even for the constant 1 which means the program is acyclic. Hence, further restrictions on the PWCs are needed to ensure tractability.

  One possibility is to consider the largest integer occurring in (lower or upper) bounds of the constraints in the PWC. We call this parameter constraint-width. If the constraint-width is taken as an additional parameter together with treewidth, then the consistency problem of PWCs becomes *fixed-parameter linear*. This means the running time is linear in the input size and only exponential in the treewidth and the constraint-width.

- For PCCs (i.e., PWCs where all weights are equal to 1) we design a new dynamic programming algorithm with a running time that can be called *non-uniform polynomial time* tractable. Let w denote the treewidth of a PCC Π and let n denote the size of Π. Then our algorithm works in time $f(w) \cdot n^{2w}$ for some function f that only depends on the treewidth, but not on the size n of the program. The term "non-uniform" refers to the factor n^{2w} in the time bound, where the size n of the program is raised to the power of an expression that depends on the treewidth w and is therefore not independent of the parameter.

  We shall also discuss further extensions of this dynamic programming algorithm for PCCs. For example, it can be used to solve in non-uniform polynomial time the consistency problem of PWCs if the weights are given in unary representation.

- Of course, an algorithm for the PCC consistency problem that is fixed-parameter tractable, i.e., that operates in time $f(w) \cdot n^{O(1)}$ would be preferable. There the parameter w would not occur in the exponent of the program size n. Unfortunately, it is shown in [92] that no such algorithm exists under common complexity theoretical assumptions. Technically, it is proven that the consistency problem of PCCs parameterized by treewidth is hard for the parameterized complexity class W[1] and therefore the problem is *fixed-parameter intractable*. In other words, a non-uniform polynomial-time running time of our dynamic programming algorithm is the best that one can expect.

- For answer-set programs without weight or cardinality constraints we present an implementation of an algorithm based on dynamic programming over tree decompositions. We evaluate this algorithm against a previously implemented one and show how algorithm selection can be used to decide for a given problem instance which of the two algorithms can be expected to perform better.

1.3.3 Abduction

We initiate the study of parameterized complexity of propositional abduction. More precisely, we study the decision problem that asks whether a given abduction instance has a solution. We also introduce two new versions of the abduction problem. One asks if a given instance has a solution that has a certain size. The other one asks if a given instance has a solution that is smaller than or equal to a certain size.

We study these three problems in the general framework of propositional logic as well as in restricted fragments thereof. Those fragments are Horn formulas, definite Horn formulas, as well as Krom formulas. The main contributions are as follows:

- We perform a classical complexity analysis of the new abduction problems asking for solutions of certain size. By proving that these problems are intractable as well, we justify a parameterized complexity analysis.

- We analyze the parameterized complexity by considering these problems with a number of different parameterizations: number of manifestations, number of hypotheses, desired size bound on the solution, treewidth, vertex cover number, and number of variables. We also study the parameterized complexity when parameterizing our problems by a combination of the above-mentioned parameters.

- We present several new fixed-parameter tractability results and even a polynomial kernel result. For example the abduction problem restricted to Krom formulas admits a polynomial kernel when parameterized by the number of hypothesis plus the number of manifestations.

- For the remaining fixed-parameter tractable problems we prove that no polynomial kernel exists unless the Polynomial Hierarchy collapses to the third level.

- For parameterizations where we do not show fixed-parameter tractability, we present parameterized intractability results by proving completeness in the W-hierarchy. For some

parameters we show that not even this is possible. In these cases we prove that the parameters do not help at all by showing NP-hardness even if the parameter values are bounded by some constant.

1.4 Structure of the Work

Chapter 2 introduces the basic notions and notations which we use throughout the thesis. This includes elementary results upon which the contributions of this thesis are based.

In Chapter 3 we discuss the state-of-the-art regarding the parameterized complexity of problems in the areas of AI and KR. We will also recall some state-of-the-art with respect to the three nonmonotonic reasoning formalisms which are studied in this work, namely belief revision, answer-set programming, and abduction.

The main part of the thesis starts with Chapter 4 which presents the study of the belief revision problem. We first introduce the problem formally and present some problem specific notation. Then we show that the covered problems can be expressed in monadic second-order logic or in one of its extensions. Hence, fixed-parameter tractability of these problems follows by Courcelle's Theorem. Finally, we present a dynamic programming algorithm with improved running time compared to the MSO based approach above.

Chapter 5 contains the study of answer-set programming. We first introduce the problem formally and present our problem specific notations. Then we present an NP-completeness result that shows that the parameter treewidth is not enough for programs with weight constraints. Next, fixed-parameter tractability for such programs is shown when considering the additional parameter constraint-width. This is done by encoding the problem in monadic second-order logic. We then present a dynamic programming approach for programs with cardinality constraints that runs in non-uniform polynomial time. Next, we present extensions of this dynamic programming algorithm to cover variations of this problem. In order to show that a non-uniform polynomial running time is optimal for our problem and its extensions we present a hardness result that shows that one cannot hope to significantly improve this time bound. Finally, we discuss an implementation of an answer-set programming solver that is based on dynamic programming over tree decompositions. We also discuss the evaluation of two similar algorithms and show how one can decide which one to use for a given problem instance.

Chapter 6 contains the study of the abduction problem. We first introduce the problem formally and present our problem specific notations. Then we perform a classical complexity analysis. By showing intractability of the studied problems we motivate a parameterized complexity analysis which is the main part of this chapter. There we focus first on the parameterized complexity of abduction restricted to Horn and definite Horn formulas. Finally, we show parameterized complexity results for abduction on Krom formulas.

Conclusions are given in Chapter 7. This contains a summary, the discussion of the results, and a comparison with related work. Finally, we discuss open issues and give an outlook towards future work.

This thesis is based on the following publications:

- Michael R. Fellows, Andreas Pfandler, Frances A. Rosamond, and Stefan Rümmele. The parameterized complexity of abduction. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2012, Toronto, Ontario, Canada, July 22-26, 2012*. AAAI Press, 2012. To appear.

- Michael Morak, Nysret Musliu, Reinhard Pichler, Stefan Rümmele, and Stefan Woltran. A new tree-decomposition based algorithm for answer set programming. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2011, Boca Raton, FL, USA, November 7-9, 2011*, pages 916–918. IEEE, 2011.

- Michael Morak, Nysret Musliu, Reinhard Pichler, Stefan Rümmele, and Stefan Woltran. Evaluating tree-decomposition based algorithms for answer set programming. In *Proceedings of the 6th International Conference on Learning and Intelligent Optimization, LION 6, Paris, France, January 16-20, 2012*, Lecture Notes in Computer Science. Springer, 2012. To appear.

- Michael Morak, Reinhard Pichler, Stefan Rümmele, and Stefan Woltran. A dynamic-programming based asp-solver. In Tomi Janhunen and Ilkka Niemelä, editors, *Proceedings of the 12th European Conference on Logics in Artificial Intelligence, JELIA 2010, Helsinki, Finland, September 13-15, 2010*, volume 6341 of *Lecture Notes in Computer Science*, pages 369–372. Springer, 2010.

- Reinhard Pichler, Stefan Rümmele, Stefan Szeider, and Stefan Woltran. Tractable answer-set programming with weight constraints: Bounded treewidth is not enough. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*, pages 508–517. AAAI Press, 2010.

- Reinhard Pichler, Stefan Rümmele, Stefan Szeider, and Stefan Woltran. Tractable answer-set programming with weight constraints: Bounded treewidth is not enough. *TPLP*, 2012. To appear.

- Reinhard Pichler, Stefan Rümmele, and Stefan Woltran. Belief revision with bounded treewidth. In Esra Erdem, Fangzhen Lin, and Torsten Schaub, editors, *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2009, Potsdam, Germany, September 14-18, 2009*, volume 5753 of *Lecture Notes in Computer Science*, pages 250–263. Springer, 2009.

CHAPTER 2

Preliminaries

2.1 Parameterized Complexity Theory

This section gives a very brief introduction to *parameterized complexity theory*. The framework of parameterized complexity theory was developed by Downey and Fellows throughout a series of papers in the early 1990s and a seminal monograph [24]. Since its foundation the field has grown rapidly. This resulted in two additional books that give an overview and an introduction to the area. One by Niedermeier [79], which focuses more on the algorithmic aspect of fixed-parameter algorithms, and one by Flum and Grohe [42], which studies parameterized complexity from the perspective of logic.

Classical complexity theory studies the computational costs of problems as a function of the input size. Many problems turned out to be NP-hard. This means the best known deterministic algorithms for these problems need time that is exponential in the input size. Indeed, under usual complexity theoretic assumption (the exponential time hypothesis), there exists no polynomial time algorithm for these problems. In order to deal with this seemingly inevitable combinatorial explosion, methods such as heuristics or approximation algorithms are used.

Another relatively new approach which is able to find exact/optimal solutions for NP-hard problems is coming from the area of parameterized complexity theory. The idea is to to confine the combinatorial explosion to one aspect of the problem, the parameter. More precisely, an instance of a parameterized problem is a pair (x, k), where x is the main part and k (usually a non-negative integer) is the parameter.

Definition 1. A parameterized problem is *fixed-parameter tractable* with respect to parameter k if there exists an algorithm solving any problem instance (x, k) of size n in time $f(k) \cdot n^c$ where f is some computable function and c is a constant independent of k. If $c = 1$ then we speak of *fixed-parameter linear* time.

FPT denotes the class of all fixed-parameter tractable decision problems.

Analogously to classical complexity theory, Downey and Fellows [24] developed a framework providing reducibility and completeness notions.

Definition 2. Let f and g be arbitrary computable functions, and let c be a constant. An *fpt-reduction* from a parameterized decision problem Π to a parameterized decision problem Ψ is a transformation that maps an instance (x, k) of Π of size n to an instance (y, l) of Ψ such that

1. the transformation is fixed-parameter tractable, i.e., it can be computed in time $f(k) \cdot n^c$,
2. $l \leq g(k)$, and
3. (x, k) is a yes-instance of Π if and only if (y, l) is a yes-instance of Ψ.

Note that because of the condition $l \leq g(k)$, the parameter value of the new problem only depends on the parameter value of the original problem.

A parameterized complexity class $\mathcal{C}$ is a class of parameterized problems. A parameterized problem Π is $\mathcal{C}$-hard if every problem in $\mathcal{C}$ is fpt-reducible to Π. Problem Π is called $\mathcal{C}$-complete if it is additionally contained in $\mathcal{C}$.

This notion leads to a variety of complexity classes. The only ones that are used in this thesis are FPT, the W-hierarchy, XP, and para-NP. Thereby XP is the class of parameterized problems solvable in time $\mathcal{O}(n^{g(k)})$ for some computable function g. A parameterized problem Π is para-NP-hard if there is some fixed k for which Π restricted to instances (x, k) is NP-hard [40].

A problem parameterized by k is in W[P] if there exists a non-deterministic algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$ using at most $f'(k) \cdot \log n$ many non deterministic steps, where f and f' are computable functions.

To show membership in the W-hierarchy we will use first-order model checking problems. This approach was first proposed by Downey, Fellows, and Regan [25], and extended by Flum and Grohe [41]. MC[$\Sigma_{t,u}$] denotes the model-checking problem over $\Sigma_{t,u}$ formulas. The class $\Sigma_{t,u}$ contains all first-order formulas of the form

$$\exists \overline{x}_1 \forall \overline{x}_2 \exists \overline{x}_3 \ldots Q \overline{x}_t \varphi(\overline{x}_1, \ldots, \overline{x}_t),$$

where φ is quantifier free and Q is an $\exists$ if t is odd and a $\forall$ if t is even, and the quantifier blocks – with the exception of the first $\exists$ block – are of length at most u. Given a finite structure $\mathcal{A}$ and a formula $\varphi \in \Sigma_{t,u}$, MC[$\Sigma_{t,u}$] asks whether $\mathcal{A}$ is a model of φ. When parameterized by $|\varphi|$, MC[$\Sigma_{t,u}$] is W[t]-complete for $t \geq 1$, $u \geq 1$ [25]. For the case $t = 1$ we will write Σ_1 instead of $\Sigma_{1,u}$ since the length of the first quantifier block is not restricted.

The following relations between parameterized complexity classes are known, see for example [24] and [42].

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \cdots \subseteq \text{W[P]} \subseteq \text{XP}$$
$$\text{W[P]} \subseteq \text{para-NP}$$

Furthermore, it is known that the following inclusion is strict:

$$\text{FPT} \subset \text{XP}$$

Of particular interest is the class W[1], which can be considered as parameterized counterpart to NP. For example, the CLIQUE problem (given a graph G and an integer k, decide whether G

contains a complete subgraph of k vertices), parameterized by k, is a well-known W[1]-complete problem. It is believed that FPT $\neq$ W[1]. There is strong theoretical evidence that supports this assumption, for example, FPT $=$ W[1] would imply that the Exponential Time Hypothesis fails, see Flum and Grohe [42]. Hence, showing that a parameterized problem is W[1]-hard is a proof that this problem does not admit an algorithm with a fixed-parameter tractable running time unless Exponential Time Hypothesis fails.

Next we will take a look at efficient preprocessing. A common method in parameterized complexity theory is to provide polynomial-time executable data-reduction rules [24]. An algorithm applying such rules is called a *kernelization algorithm*.

Definition 3. Let f and g be arbitrary computable functions. A *kernelization algorithm* is a transformation of a problem instance (x, k) of Π into an instance (y, l) of Π such that

1. (x, k) is a yes-instance if and only if (y, l) is a yes-instance,
2. $l \leq f(k)$, and
3. y has size $\leq g(k)$.

It is easily shown that a parameterized problem Π is FPT if and only if it admits a kernelization algorithm. Although in general the kernelization bound $g(k)$ may be exponential in k, it has been shown that many FPT problems admit *polynomial kernels*, that is, polynomial time kernelization algorithms where the kernelization bound is a polynomial function of k, $g(k) = k^{\mathcal{O}(1)}$.

Recently, the subfield of kernelization has received increased interest with the development of methods for proving lower bounds [10, 14]. These methods allow to show that for some problems there cannot exist polynomial kernels unless the Polynomial Hierarchy (PH) collapses.

Many of these results are using the following notion.

Definition 4. A parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is *compositional* if there exists an algorithm that computes, given a sequence $(x_1, k), \ldots, (x_t, k) \in \Sigma^* \times \mathbb{N}$, a new instance $(x', k') \subseteq \Sigma^* \times \mathbb{N}$ such that the following properties hold:

1. The algorithm only requires time polynomial in $\sum_{i=1}^{t} |x_i| + k$,
2. (x', k') is a yes-instance if and only if there is some $1 \leq i \leq t$ such that (x_i, k) is a yes-instance, and
3. $k' \leq k^{\mathcal{O}(1)}$.

Given a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$, we call the problem

$$\{x\#1^k \mid (x, k) \in \Pi\}$$

the unparameterized version of Π, where $\#$ is a new symbol not included in Σ and 1 is an arbitrary symbol. Bodlaender, Downey, Fellows, and Hermelin [10] showed that if a problem is hard and it is compositional, then it does not admit a polynomial kernel.

Theorem 5 ([10]). *Let Π be a parameterized problem such that the unparameterized version of Π is* NP*-complete. If Π is compositional, then it does not admit a polynomial kernel unless the Polynomial Hierarchy collapses to the third level (*$\mathsf{PH} = \Sigma_3^{\mathsf{P}}$*).*

A polynomial parameter and time (PPT) reduction is a polynomial time reduction increasing the parameter only polynomially. Such a reduction preserves polynomial kernels:

Theorem 6 ([14]). *Let Π and Ψ be parameterized problems, where the unparameterized version of Π is* NP*-complete and the unparameterized version of Ψ is in* NP*. Then a PPT reduction from Π to Ψ implies that if Ψ has a polynomial kernel, Π has a polynomial kernel.*

Theorem 6 can be used to show that a parameterized problem Ψ does not admit a polynomial kernel. Assume we know that a parameterized problem Π does not admit a polynomial kernel. By showing that there exists a PPT reduction from Π to Ψ we can exclude a polynomial kernel for Ψ as well since a polynomial kernel for Ψ would imply a polynomial kernel for Π. But this would contradict our assumption.

Using PPT reductions for proving kernel lower bounds is often more convenient than showing that the problem is compositional.

2.2 Graphs and Treewidth

In this work we deal only with *undirected, simple graphs*, that means graphs without self-loops and with not more than one edge between two vertices. We denote a graph by a pair (V, E), where V denotes the set of *vertices* and E is the set of *edges*. An edge is a subset of V of cardinality 2.

The treewidth of a graph measures, intuitively speaking, its "tree-likeness". A formal definition is given below. Nowadays treewidth is often used as a parameter in the study of parameterized complexity. It was originally studied in graph theory. Robertson and Seymour [97, 98] introduced the notions of *treewidth* and *tree-decomposition* as part of their graph minors research.

Definition 7. A *tree decomposition* of a graph $G = (V, E)$ is a pair (T, χ), where T is a tree and χ maps each node t of T (we use $t \in T$ as a shorthand) to a *bag* $\chi(t) \subseteq V$, such that

1. for each $v \in V$, there is a $t \in T$, such that $v \in \chi(t)$,
2. for each $\{v, w\} \in E$, there is a $t \in T$, such that $\{v, w\} \subseteq \chi(t)$, and
3. for each $t_1, t_2, t_3 \in T$, such that t_2 lies on the path from t_1 to t_3, we have $\chi(t_1) \cap \chi(t_3) \subseteq \chi(t_2)$.

Note that we refer to the elements in the tree of a tree decomposition as nodes while we called the elements of a graph vertices. This is done to make the distinction between the original graph and its tree decomposition more clear.

A tree decomposition (T, χ) is *normalized* (or *nice*) [60] if T is a rooted tree and the following conditions hold:

1. each $t \in T$ has at most 2 child nodes,
2. for each $t \in T$ with two child nodes r and s, $\chi(t) = \chi(r) = \chi(s)$, and
3. for each $t \in T$ with one child node s, $\chi(t)$ and $\chi(s)$ differ in exactly one element.

The *width* of a tree decomposition (T, χ) is the number

$$\max\{|\chi(t)| \mid t \in T\} - 1.$$

It is known that every tree decomposition can be normalized in linear time without increasing the width [60]. The *treewidth* of a graph G, denoted as $tw(G)$, is the minimum of the widths of all tree decompositions of G.

As mentioned above, intuitively one might say that $tw(G)$ measures the "tree-likeness" of graph G. For example, the treewidth of trees is 1 and the treewidth of complete graphs with n vertices is $n - 1$.

For arbitrary but fixed $w \geq 1$, it is possible in linear time to decide if a graph has treewidth $\leq w$ and, if so, to compute a tree decomposition of width w [7]. In other words, the problem of deciding if a graph has certain treewidth is fixed-parameter linear. Recently, Bodlaender, Jansen, and Kratsch [11] showed that this decision problem admits a polynomial kernel when parameterized either by vertex cover number or by feedback vertex set. They also studied kernelization lower bounds. See Kloks [60] for an introduction into the area of computing or approximating treewidth. For an overview on treewidth see for example the surveys by Bodlaender [8, 9].

2.3 Propositional Logic

This section briefly recalls some basic notions and definitions from propositional logic. We assume a countable infinite set of *(propositional) variables*. Variables are sometimes called *atoms*. In this work we will use both terms synonymously. Let PROP denote the class of *(propositional) formulas*. These formulas are defined recursively as follows:

1. If x is a variable, then $x \in$ PROP.
2. If $\varphi \in$ PROP, then its negation $\neg\varphi \in$ PROP.
3. If $\varphi, \psi \in$ PROP, then their conjunction $(\varphi \wedge \psi) \in$ PROP and disjunction $(\varphi \vee \psi) \in$ PROP.

As usual, we omit unnecessary parentheses and write for example $(\varphi \vee \psi \vee \chi)$ instead of $((\varphi \vee \psi) \vee \chi)$. We denote by $var(\varphi)$ the set of propositional variables occurring in a formula φ. A *literal* is either a variable x or its negation $\neg x$. A *clause* is a disjunction of literals $(l_1 \vee l_2 \vee \cdots \vee l_m)$. A formula in *conjunctive normal form* (CNF) is a conjunction of clauses $c_1 \wedge c_2 \wedge \cdots \wedge c_n$. It is sometimes convenient to represent clauses and formulas in CNF as sets. Thereby the clause $(l_1 \vee l_2 \vee \cdots \vee l_m)$ is represented by $\{l_1, \ldots, l_m\}$ and the formula $c_1 \wedge c_2 \wedge \cdots \wedge c_n$ is represented by $\{c_1, \ldots, c_n\}$.

An *interpretation* (or *assignment*) I is a subset of variables. The intended meaning is that I contains exactly those variables that are set to true. The remaining ones are set to false.

In the obvious way one defines what it means for an interpretation to *satisfy* a formula (or a subformula). If interpretation I satisfies formula φ, I is called a *model* of φ. The set of all models of φ is denoted by $Mod(\varphi)$. Let φ and ψ be formulas. We say φ entails ψ, denoted by $\varphi \models \psi$, if each model of φ is also a model of ψ.

In the context of dynamic programming we will talk about models with respect to a given *universe*. Thereby a *universe* U is a set of variables. The models of a clause c with respect to U are given by

$$Mod_U(c) := \{I \subseteq U \mid I \cap c \neq \emptyset\} \cup \{I \subseteq U \mid \exists x \in U \setminus I, \neg x \in c\}.$$

The models of a set C of clauses with respect to a universe U are given by

$$Mod_U(C) := \bigcap_{c \in C} Mod_U(c).$$

For the universe U of all variables in $var(C)$, it holds that $Mod_U(C) = Mod(C)$.

The class of formulas in CNF is denoted by CNF. The subclass of CNF of formulas having clause size at most 2 is denoted by KROM. *Horn* formulas are formulas in CNF with at most one positive literal per clause. *Definite Horn* formulas have exactly one positive literal per clause. We denote the corresponding classes of formulas by HORN respectively DEFHORN.

Let $Res(\varphi)$ be the operator extending $\varphi \in$ CNF by iteratively applying *resolution* and dropping tautological clauses until a fixed-point is reached. Applying resolution adds the clause $c_1 \cup c_2$ to φ if there exists a variable x such that $c_1 \cup \{x\} \in \varphi$ and $c_2 \cup \{\neg x\} \in \varphi$. Resolution on Krom formulas will always yield a Krom formula. In that case $Res(\varphi)$ can be computed in polynomial time. Let c be a non-tautological clause and let $var(\{c\}) \subseteq var(\varphi)$ then $c \in Res(\varphi)$ if and only if $\varphi \models c$. For details, see for example [65].

In order to study reasoning problems parameterized by treewidth, we represent CNF formulas by *incidence graphs* or *primal graphs*. For CNF formula $\varphi \in$ CNF, the incidence graph is given by $\mathcal{G}_I = (V_I, E_I)$ where V_I contains a vertex for each variable and each clause in φ, and E_I is the set of all edges $\{x, c\}$ such that variable x occurs in clause c. The primal graph of φ is given by $\mathcal{G}_P = (V_P, E_P)$ where V_P contains a vertex for each variable and E_P is the set of all edges $\{x, y\}$ such that variables x and y occur together in a clause.

A *Boolean circuit* is a directed acyclic graph where exactly one of the nodes has out-degree 0. This node is called the output gate. Nodes with in-degree 0 are called input gates. All nodes except the input gates are of one of the following types: AND-gate, OR-gate, or NEGATION-gate. An *assignment* to a circuit is a function that sets each input gate to true or false. An assignment is a *satisfying assignment*, if the computed value of the output gate is true. The *weight* of an assignment is the number of input gates set to true.

2.4 Monadic Second Order Logic and Courcelle's Theorem

Monadic second-order logic (MSO) extends first-order logic by the use of *set variables* (denoted by upper case letters), which range over sets of domain elements.

An important tool for establishing fixed-parameter tractability of a decision problem on graphs when parameterized by treewidth is *Courcelle's Theorem* [17]. Courcelle's Theorem

states that any property of finite structures, that is definable in *monadic second-order logic* (MSO), can be decided in linear time over graphs with bounded treewidth. In other words, decision problems that are expressible in MSO are fixed-parameter linear over finite graphs when parameterized by the treewidth of the graph. An alternative proof of this result can be found in the book of Downey and Fellow [24].

Arnborg, Lagergren, Seese [4] showed an extension of Courcelle's Theorem. They defined a logic called *extended monadic second order logic*. This logic provides the opportunity to not only express decision problems, but also to express optimization (extremum) problems and counting problems as well. They showed that problems expressible in extended MSO are fixed-parameter linear over finite graphs when parameterized by the treewidth of the graph.

Szeider [109] proposed another extension of Courcelle's Theorem. Thereby, the vertices of a graph are labeled by a set of integers. Each vertex v of the selected solution has to be adjacent to α other vertices in the solution where α occurs in the label of v. Additionally to these constraints on the vertices, the solution has to satisfy a property that is expressed in MSO. Problems that can be expressed in this framework are fixed-parameter tractable when parameterized by treewidth.

2.5 Some Parameterized Problems

We briefly discuss some decision problems together with their complexity with respect to typical parameters. These problems will be used throughout the thesis in various (parameterized) reductions in order to prove hardness results.

VERTEX COVER	
Instance:	A graph G and an integer k.
Problem:	Does G have a vertex cover of size k?

A *vertex cover* of a graph $G = (V, E)$ is a set of vertices $V' \subseteq V$, such that each edge $e \in E$ is incident to at least one vertex contained in V'.

VERTEX COVER parameterized by k is FPT.

The *vertex cover number* $\tau(G)$ of a graph G is the size of the smallest vertex cover of G. A well known property is that for every graph G, $\tau(G) \geq tw(G)$. Therefore parameterized hardness results with respect to parameter vertex cover number carry over to parameterizing by treewidth.

INDEPENDENT SET	
Instance:	A graph G and an integer k.
Problem:	Does G have an independent set of size k?

An *independent set* of a graph $G = (V, E)$ is a set of vertices $V' \subseteq V$, such that for every edge $e \in E$ at least one of its incident vertices is not contained in V'.

INDEPENDENT SET parameterized by k is W[1]-complete.

WEIGHTED MONOTONE CIRCUIT SAT$_{\leq/=}$	
Instance:	A monotone circuit C and an integer k.
Problem:	Is there a satisfying assignment for C having weight of at most / exactly k?

A *monotone circuit* is a Boolean circuit that does not contain negations. WEIGHTED MONOTONE CIRCUIT SAT$_{\leq/=}$ parameterized by k is W[P]-complete, even when all AND-gates and all OR-gates are binary.

MULTICOLORED INDEPENDENT SET	
Instance:	A graph $G = (V, E)$, an integer k, and a k-coloring $\mathsf{c} : V \to \{c_1, \ldots, c_k\}$.
Problem:	Does G have an independent set V' of size k, such that for all $\{x, y\} \subseteq V'$: $\mathsf{c}(x) \neq \mathsf{c}(y)$?

Recall that problems the INDEPENDENT SET and CLIQUE are closely related. An independent set of size k of a graph G is a clique of size k of the complement of G. The complement of G a graph consists of exactly those edges that are missing from G. It is easy to see that this relationship yields an fpt-reduction in both direction and that this reduction also preserves the k-coloring. In [35] it was shown that MULTICOLORED CLIQUE parameterized by k is W[1]-complete. By the observation above it follows that MULTICOLORED INDEPENDENT SET parameterized by k is W[1]-complete as well.

RED-BLUE DOMINATING SET	
Instance:	A bipartite graph $G = (V_{\text{red}} \cup V_{\text{blue}}, E)$ and an integer k.
Problem:	Is there a set $S \subseteq V_{\text{red}}$ with $\lvert S\rvert \leq k$ such that each vertex in V_{blue} is adjacent to a vertex in S?

Fernau [37] showed that this problem is W[2]-complete when parameterized by k.

CHAPTER 3

State of the Art

3.1 Belief Revision

The area of *belief revision* addresses the problem of *consistently* incorporating new information into an existing knowledge base. Problems arising from this area have been studied since the early 1980s. An extensive introduction into the area is for example the handbook article by Peppas [90].

Alchourrón, Gärdenfors, and Makinson proposed the famous *AGM postulates* [2]. The framework for belief revision which they proposed is described on a very high level. They just assume a logic language that is closed under Boolean connectives together with a consequence operator. The postulates themselves try to express the notion of "rational belief revision". The intuition is to change the existing knowledge base as little as possible.

It was observed by Katsuno and Mendelzon [59] that the AGM postulates in some cases would lead to non-intuitive solutions. These cases are related to situations where some agent performs some change in its environment and then needs to update his beliefs accordingly. The subarea that emerged from their findings is called *belief update*. Following the spirit of the AGM postulates, Katsuno and Mendelzon proposed their own set of postulates which capture these update scenarios [59].

Since the above-mentioned postulates define belief revision and belief update in a very general way, there is still plenty of room for defining actual realizations of revision and update. In the literature many different so-called *revision operators* and *update operators*, have thus been proposed. These operators take as arguments the knowledge base and the new information and compute a new knowledge base. Four of the most fundamental approaches are due to Dalal [20], Satoh [102], Winslett [115], and Forbus [43].

Eiter and Gottlob [30] studied the complexity of various update and revision operators. For the problem of deciding whether a given formula holds in the revised knowledge base, their results include Π_2^P-completeness for Satoh's, Winslett's, and Forbus' operator as well as Θ_2^P-completeness for Dalal's operator. They also considered special cases where the theory is not an arbitrary propositional formula, but is restricted to certain fragments, like Horn formulas. For

Horn formulas the operators due to Satoh and Winslett are coNP-complete while the operators due to Forbus and Dalal remain Π_2^P-complete and Θ_2^P-complete respectively.

Liberatore and Schaerf [68] analyzed the complexity of the following model checking problem. Given an interpretation, decide if it is a model of the revised knowledge base. Darwiche [21] did some work on belief revision with bounded treewidth formulas. He relies on compilation techniques to transforms such formulas in linear time into an efficient representation for belief revision.

3.2 Answer-Set Programming

Answer-set programming (ASP) (also known as A-Prolog) has evolved as a paradigm that allows for very elegant solutions to many combinatorial problems. An introduction to ASP can for example be found in the recent survey paper by Brewka, Eiter, and Truszczyński [15].

An answer-set program consists of facts, rules, and constraints. The basic idea is to encode a computational problem as an answer-set program such that the models of the program correspond to the solutions of the considered problem. The semantics according to which these models are computed is called the *stable model semantics* and was introduced by Gelfond and Lifschitz [45].

The answer-set programming approach for declarative problem solving was proposed by Niemelä [80] and Marek and Truszczyński [73]. The term "answer-set programming" was later introduced by Lifschitz [69].

The basic form of answer-set programs are called *normal programs*. It was shown by Marek and Truszczyński [72] that it is NP-complete to decide whether a normal program has a solution. *Disjunctive logic programs* allow to express disjunctions in the head of rules. For these programs Eiter and Gottlob [32] showed that it is Σ_2^{P}-complete to decide whether a solution exists. Truszczyński [111] proved a trichotomy result that classifies disjunctive programs into classes that are either in P, NP-complete, or Σ_2^{P}-complete.

Niemelä, Simons and Soininen [82] extended logic programs with cardinality or, more generally, weight constraints. This allows an even larger class of problems to be accessible by answer-set programming. For instance, in the product configuration domain, we need to express cardinality, cost, and resource constraints, which are very difficult to capture using logic programs without weights.

Some applications of ASP include the construction of phylogenies in bioinformatics [34], product configuration [104], decision support systems like the one for the space shuttle [85], data integration [66], and semantic web [33].

There exist many system implementations of the ASP paradigm, for example CLASP [44], DLV [67], SMODELS [81], and CMODELS [46].

The already existing work on the parameterized complexity of answer-set programming is reviewed in Section 3.4.

3.3 Abduction

Abduction as a formalism of reasoning was studied by Peirce [89]. The term abduction was introduced by him as well. Abductive reasoning was introduced into the area of artificial intelligence by Morgan [77] and Pople [95].

Since then abduction has been applied in many areas inside and outside of AI. Examples include system diagnosis [78], medical diagnosis [95], and text interpretation [55].

Early work on the computational complexity of propositional abduction was done by Selman and Levesque [103]. Among others they showed that abduction is still NP-complete when restricted to Horn formulas. A systematic complexity analysis was done by Eiter and Gottlob [31]. Their results include that abduction for general propositional formulas is Σ_2^{P}-complete and that abduction restricted to definite Horn formulas can be solved in polynomial time. The classical complexity of abduction has been extensively studied in other papers as well, for example [18, 19, 86, 94]. The abduction problem for Krom formulas was shown to be NP-complete by Nordh and Zanuttini [86]. Creignou and Zanuttini [19] showed that the same problem can be computed in polynomial time if there exists only a single manifestation.

3.4 Parameterized Complexity of Reasoning

The study of the parameterized complexity of nonmonotonic reasoning was initiated by Gottlob, Scarcello, and Sideri [48] at a time when the field of parameterized complexity theory itself was very young. They showed fixed-parameter tractability of a variety of problems and parameterizations. Among these are constraint satisfaction parameterized by treewidth and domain size, and propositional logic programming under the stable model semantics parameterized by the feedback width of the dependency graph. They also showed that circumscription parameterized by the model size is fixed-parameter intractable.

Despite this early work, in the following years there was not much research conducted on the parameterized complexity of nonmonotonic reasoning. Most of the work on parameterized complexity focused on problems from graph theory. This has changed recently and nowadays the study of parameterized complexity of problems from other areas is gaining increased attention.

The parameterized complexity of the propositional satisfiability (SAT) problem and constraint satisfaction problems (CSPs) are two notable exceptions. Both of them are well studied.

Papadimitriou and Yannakakis [88] showed that CSP is W[1]-complete when parameterized by the number of variables or the sum of the arities of the constraints. Note that they actually studied Boolean conjunctive queries but this problem is closely related to CSP [61]. As mentioned above, CSP is fixed-parameter tractable when parameterized by treewidth of the primal graph plus the domain size [48]. But it is W[1]-hard if one takes the treewidth of the incidence graph instead [101]. Grohe [52] showed that for the problem of deciding whether a CSP instance admits a solution, the class of instances of bounded treewidth is exactly the class of structural restricted instances that lead to tractability. Greco and Scarcello [50, 51] extended this result to enumeration and optimization problems. Grohe and Marx [53] studied CSP for instances of bounded fractional hypertree width.

SAT as well as its counting variant is not only fixed-parameter tractable when parameterized by treewidth of the primal graph but also for treewidth of the incidence graph [105, 39, 100]. Alekhnovich and Razborov [3] studied SAT parameterized by branchwidth. Fischer, Makowsky, and Ravve [39] showed that SAT is fixed-parameter tractable when parameterized by directed clique-width. The parameter clique-width can be approximated by an fpt algorithm [56]. A lot of research investigated so-called *backdoor sets*, see for example [106, 84]. The size of such sets of variables measures the distance to "easy" instances. For example a deletion Horn backdoor set contains those variables when removed from the clauses the remaining formula is Horn. Another form of backdoor sets are so-called strong backdoor sets. A strong backdoor set for formula class C is a set of variables, such that for all possible truth assignments to those variables the remaining formula is contained in C. Nishimura, Ragde and Szeider [83] showed that detecting strong backdoor sets for Horn or Krom is fixed-parameter tractable. Parameterizing SAT by these backdoor sets leads to fixed-parameter tractability as well. The notion of backdoor sets has been generalized to quantified Boolean formulas by Samer and Szeider [99].

For answer-set programming the solution size has been considered as a parameter. Lonc and Truszczyński [71] showed that finding stable models of size at most k is W[2]-complete. Truszczyński [110] showed that finding stable models containing all except for at most k facts is fixed-parameter tractable. As mentioned above, Gottlob, Scarcello, and Sideri [48] considered the feedback width of the dependency graph as a parameter and proved fixed-parameter tractability. Gottlob, Pichler, and Wei [47] showed with the help of an MSO encoding that the parameter treewidth leads to fixed-parameter tractability as well. This also holds for the brave and cautious reasoning problems associated with ASP. A dynamic programming algorithm for evaluating answer-set programs of bounded treewidth was proposed by Jakl, Pichler, and Woltran [57]. Fichte and Szeider [38] studied backdoor sets for answer-set programming.

Besides the mentioned MSO encoding for ASP, [47] shows how to express several other problems from the area of AI in monadic second-order logic. From Courcelle's Theorem it follows that they all are fixed-parameter tractable when parameterized by treewidth. For example encodings for closed world reasoning as well as propositional abduction are contained.

At the core of many nonmonotonic reasoning formalisms lies the task of computing minimal models. Recently, Lackner and Pfandler [63] did a parameterized complexity analysis of this problem using a variety of parameters and discovered several tractable fragments. This analysis has been extended to the more general circumscription formalism in [64].

Ordyniak and Szeider [87] studied Bayesian structure learning parameterized by treewidth. The parameterized complexity of planning has been studied by Downey, Fellows, and Stege [26] and Bäckström, Chen, Jonsson, Ordyniak, and Szeider [5].

Furthermore, there exists some work on the parameterized complexity of abstract argumentation. Dunne [27] as well as Dvorák, Pichler, and Woltran [29] considered the parameter treewidth. Dvorák, Ordyniak, and Szeider [28] studied argumentation problems parameterized by distance measures to tractable fragments.

Szeider [108] showed for a number of parameterized problems in AI, including answer-set programming parameterized by the feedback width, that they do not admit a polynomial kernel.

Further references can be found in the recent survey of Gottlob and Szeider [49] on parameterized complexity of problems in AI and database theory.

CHAPTER 4

Belief Revision

This chapter is based on joint work with Reinhard Pichler and Stefan Woltran, and appeared in the proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2009) [93].

In this chapter we initiate a parameterized complexity analysis of belief revision problems. Recall from Chapter 1 that the area of belief revision studies how to revise a given knowledge base when new information is obtained that is contradicting some old knowledge.

Here we restrict ourselves to *propositional knowledge bases*, i.e., knowledge bases that are given by propositional formulas. Problems arising from the revision of propositional knowledge bases have been intensively studied for two decades. In the literature many different realizations of revision, called *revision operators*, have thus been proposed. These operators take as an argument the knowledge base and the new information and compute a new knowledge base. Three of the most fundamental approaches are due to Dalal [20], Satoh [102], and Winslett [115].

We will consider these operators in combination with two computational problems. The *reasoning* problem asks whether a given formula will be logically entailed by the revised knowledge base. For the *enumeration* problem one has to compute all models of the revised knowledge base. The parameter which we study thereby is treewidth.

We show fixed-parameter tractability for the reasoning problem with Satoh's operator and Winslett's operator by giving an MSO definition of this problem. This means, we will consider finite graphs that represent the formulas of the old knowledge base and the formula expressing the new information, and define for them a property in MSO that is true if and only if the answer to the reasoning problem is yes. In order to prove an analogous result for Dalal's operator, we have to make use of an extension of Courcelle's Theorem due to Arnborg, Lagergren, and Seese [4].

The proof of Courcelle's Theorem and its extension in [4] is "constructive": It works by transforming the MSO evaluation problem into a tree language recognition problem, which is then solved via a finite tree automaton (FTA). However, the "algorithms" resulting from such an MSO-to-FTA transformation are usually not practical due to excessively large constants. Consequently, Niedermeier states that MSO "is a very elegant and powerful tool for quickly

deciding about FPT, but it is far from any efficient implementation" [79]. We therefore present a novel algorithm for the reasoning problem with Dalal's operator. This algorithm is based on dynamic programming and builds upon an algorithm of [100] for the #SAT problem (i.e., the problem of counting all models of a given propositional formula). Moreover, we extend our reasoning algorithm to an algorithm for the enumeration problem of Dalal's operator. As far as the complexity is concerned, for bounded treewidth our algorithms work in linear time for the reasoning problem and with linear delay for the enumeration problem. Linear delay means that the time needed for computing the first model and for computing any further model of the revised knowledge base is linear in the input size.

4.1 Belief Revision Background

Formally, the problem of belief revision is usually specified as follows: Given a knowledge base (i.e., a formula) α and a formula β, find a revised knowledge base $\alpha \circ \beta$, such that β is true in all models of $\alpha \circ \beta$ and the change compared to the models of α is minimal. The following problems are of great interest:

- *Reasoning*. Given formulas α, β, and γ, decide if $\alpha \circ \beta \models \gamma$ holds.
- *Enumeration*. Given formulas α and β, compute the models of $\alpha \circ \beta$.

Several realizations for $\circ$ have been proposed in the literature. The desired properties for these have been formulated by Alchourrón, Gärdenfors, and Makinson in the famous AGM-Postulates [2], or in terms of propositional logic and finite knowledge bases, by Katsuno and Mendelzon [59]. Three of the most fundamental approaches are due to Dalal [20], Satoh [102], and Winslett [115]. Complexity results for the reasoning problem with respect to different operators $\circ$ are provided in [30] including Θ_2^P-completeness for the operator due to Dalal and Π_2^P-completeness for Satoh's and Winslett's operator.

4.1.1 Revision Operators

The approaches of revision we deal with in this work rely on so-called model-based change operators. Such operators usually utilize a model distance $M \Delta M'$ which yields the set of atoms differently assigned in interpretations M and M', i.e. in our notation, $M \Delta M' = (M \setminus M') \cup (M' \setminus M)$. Assuming that α is consistent (we tacitly make this assumption throughout this work), the operators due to Satoh [102] ("$\circ_S$"), Dalal [20] ("$\circ_D$"), and Winslett [115] ("$\circ_W$") can be defined as follows:

$$
\begin{aligned}
Mod(\alpha \circ_D \beta) &= \{J \in Mod(\beta) \mid \exists I \in Mod(\alpha) \text{ such that } |I \Delta J| = |\Delta|^{min}(\alpha, \beta)\}, \\
Mod(\alpha \circ_S \beta) &= \{J \in Mod(\beta) \mid \exists I \in Mod(\alpha) \text{ such that } I \Delta J \in \Delta^{min}(\alpha, \beta)\}, \\
Mod(\alpha \circ_W \beta) &= \bigcup_{I \in Mod(\alpha)} \{J \in Mod(\beta) \mid I \Delta J \in \Delta_I^{min}(\beta)\},
\end{aligned}
$$

where

$$\begin{aligned} |\Delta|^{min}(\alpha,\beta) &= min(\{|I\Delta J| \mid I \in Mod(\alpha), J \in Mod(\beta)\}), \\ \Delta^{min}(\alpha,\beta) &= min_{\subseteq}(\{I\Delta J \mid I \in Mod(\alpha), J \in Mod(\beta)\}), \\ \Delta_I^{min}(\beta) &= min_{\subseteq}(\{I\Delta J \mid J \in Mod(\beta)\}), \end{aligned}$$

with $min_{\subseteq}$ selecting elements which are minimal with respect to set inclusion. The intuition behind these operators is the following. Dalal uses the criteria of cardinality to select those models of β with a minimal change compared to those of α. This means that those models of β are selected that differ in the least variables from any model of α. Satoh's operator is very similar to the one due to Dalal. Again only those models of β are selected that are closest to any model of α. But here closeness is defined via minimality with respect to set inclusion instead of cardinality. Winslett's operator is an update operator instead of a revision operator. The difference is that every model is changed individually. For every single model of α the closest model(s) of β are selected. The outcome of these individual selections is then combined. Winslett defines closeness similar to Satoh via minimality with respect to set inclusion.

Subsequently, we refer to a *revision scenario* as either a pair of formulas (α, β) (in case of the enumeration problem) or a triple (α, β, γ) (in case of the reasoning problem).

As shown in [30], given formulas α, β, γ, deciding $\alpha \circ \beta \models \gamma$ with $\circ \in \{\circ_S, \circ_W\}$ is Π_2^P-complete while deciding $\alpha \circ_D \beta \models \gamma$ is Θ_2^P-complete. For these results, hardness holds even in case γ is a single atom.

4.1.2 Tree Decompositions for Revision Scenarios

To build tree decompositions for revision scenarios (α, β, γ), we use incidence graphs[1] over $\Gamma = \alpha \cup \beta \cup \gamma$. Recall from Section 2.3 that for formulas α,β, and γ, such a graph G is given by vertices $\Gamma \cup \mathit{var}(\Gamma)$ and has as edges all unordered pairs $\{a, c\}$ with an atom $a \in \mathit{var}(\Gamma)$ appearing in a clause $c \in \Gamma$. In case of normalized tree decompositions, we distinguish between six types of nodes: atom introduction (AI), clause introduction (CI), atom removal (AR), clause removal (CR), branch (B), and leaf (L) nodes. The first four types will be often augmented with the element e (either an atom or clauses) which is removed or added compared to the bag of the child node.

Example 8. The revision scenario $A \circ B$, which is used as a running example throughout this chapter, is given by the following formulas:

$$\begin{aligned} A &= \underbrace{(x \vee y)}_{c_1} \wedge \underbrace{(\neg y)}_{c_2} \wedge \underbrace{(\neg z)}_{c_3}, \\ B &= \underbrace{(\neg x \vee y)}_{c_4} \wedge \underbrace{(\neg y \vee z)}_{c_5}. \end{aligned}$$

These have models $Mod(A) = \{\{x\}\}$ and $Mod(B) = \{\{\}, \{z\}, \{y, z\}, \{x, y, z\}\}$. Therefore, $|\Delta|^{min}(A, B) = 1$ and $\Delta^{min}(A, B) = \Delta_{\{x\}}^{min}(B) = \{\{x\}, \{y, z\}\}$. Hence, the results are

[1] See [100] for justifications why incidence graphs are favorable over other types of graphs.

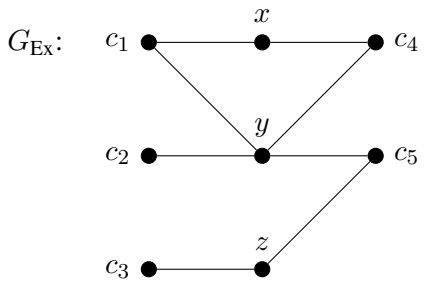

Figure 4.1: The incidence graph G_{Ex} of the revision scenario $A \circ B$ from Example 8.

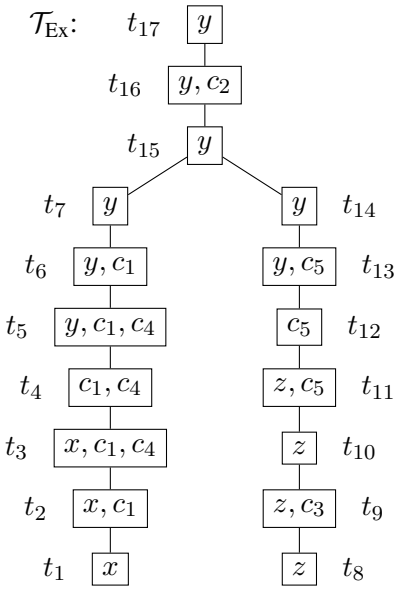

Figure 4.2: A normalized tree decomposition $\mathcal{T}_{\text{Ex}}$ of G_{Ex} from Example 8.

$Mod(A \circ_D B) = \{\{\}\}$ and $Mod(A \circ_S B) = Mod(A \circ_W B) = \{\{\}, \{x, y, z\}\}$. Figure 4.1 shows the incidence graph G_{Ex} of this scenario. A normalized tree decomposition $\mathcal{T}_{\text{Ex}}$ of G_{Ex} having width 2 is shown in Figure 4.2. Actually, G_{Ex} cannot have a tree decomposition of width < 2, since only trees have treewidth = 1 and G_{Ex} contains a cycle. Hence, the tree decomposition in Figure 4.2 is optimal and we have $tw(G_{\text{Ex}}) = 2$. Examples for different node types are t_1 as (L) node, t_2 as (c_2-CI) node, t_4 as (x-AR) node, t_5 as (y-AI) node, t_6 as (c_4-CR) node, and t_{15} as (B) node. ⊣

4.2 Applying Courcelle's Theorem

In order to show the fixed-parameter tractability of the aforementioned belief revision problems using Courcelle's Theorem, we first have to define how to model the instances as finite structures.

Let formulas α, β, and γ be given by a structure $\mathcal{A}$ with signature

$$\sigma = \{\mathit{atom}(\cdot), \mathit{clause}_\alpha(\cdot), \mathit{clause}_\beta(\cdot), \mathit{clause}_\gamma(\cdot), \mathit{pos}(\cdot,\cdot), \mathit{neg}(\cdot,\cdot)\}.$$

Structure $\mathcal{A}$ has domain $A = \Gamma \cup \mathit{var}(\Gamma)$, where $\Gamma = \alpha \cup \beta \cup \gamma$. Moreover, for each relation symbol in σ, a relation over A is contained in $\mathcal{A}$ with the following intended meaning: *atom* designates the set of atoms, clause_α, clause_β and clause_γ denote the set of clauses of α, β, and γ respectively. Furthermore, $\mathit{pos}(a, c)$ denotes that atom a occurs positively in clause c. Negative literals are described by $\mathit{neg}(a, c)$.

The treewidth of a structure $\mathcal{A}$ is defined as the treewidth of the graph that we get by taking the set of domain elements (in our case, $A = \Gamma \cup \mathit{var}(\Gamma)$) as vertices and by considering two vertices (i.e., domain elements) as adjacent if these domain elements jointly occur in some tuple of the structure, i.e., the edges of this graph are of the form (a, c) where either $\mathit{pos}(a, c)$ or $\mathit{neg}(a, c)$ is contained in the structure. Hence, the treewidth of $\mathcal{A}$ is precisely the treewidth defined via the *incidence graph* of Γ as described in the previous section.

The fact that set I is a model of a formula φ can then be stated by the following MSO property (see also [47]):

$$\begin{aligned} \mathit{mod}_\varphi(I) \quad \equiv \quad & \forall x[x \in I \rightarrow \mathit{atom}(x)] \wedge \\ & \forall c[\mathit{clause}_\varphi(c) \rightarrow \exists a((\mathit{pos}(a, c) \wedge a \in I) \vee (\mathit{neg}(a, c) \wedge a \notin I))]. \end{aligned}$$

Towards an MSO-encoding for $\alpha \circ \beta$ we define two more helper formulas. The first one characterizes valid triples $I \Delta J = K$ and the second one characterizes proper subsets $X \subset Y$.

$$\begin{aligned} \mathit{diff}(I, J, K) \quad &\equiv \quad \forall a[a \in K \leftrightarrow ((a \in I \wedge a \notin J) \vee (a \notin I \wedge a \in J))], \\ \mathit{sub}(X, Y) \quad &\equiv \quad \forall a(a \in X \rightarrow a \in Y) \wedge \exists b(b \in Y \wedge b \notin X). \end{aligned}$$

We put things together to characterize the models of $\alpha \circ_S \beta$:

$$\mathit{rev}^S_{\alpha,\beta}(J) \equiv \exists K[\mathit{modD}_{\alpha,\beta}(J, K) \wedge \forall J' \forall K'(\mathit{sub}(K', K) \rightarrow \neg \mathit{modD}_{\alpha,\beta}(J', K'))], \text{ with}$$
$$\mathit{modD}_{\alpha,\beta}(J, K) \equiv \mathit{mod}_\beta(J) \wedge \exists I[\mathit{mod}_\alpha(I) \wedge \mathit{diff}(I, J, K)].$$

It is now easy to see that the MSO formula $\forall J(\mathit{rev}^S_{\alpha,\beta}(J) \rightarrow \mathit{Mod}_\gamma(J))$ characterizes the reasoning problem $\alpha \circ_S \beta \models \gamma$ for Satoh's revision operator. Using the same helper formulas we can express Winslett's operator $\circ_W$ as well:

$$\begin{aligned} \mathit{rev}^W_{\alpha,\beta}(J) \equiv \mathit{mod}_\beta(J) \wedge \exists I \exists K \big[& \mathit{mod}_\alpha(I) \wedge \mathit{diff}(I, J, K) \wedge \\ & \neg \exists J' \exists K' \big(\mathit{mod}_\beta(J') \wedge \mathit{diff}(I, J', K') \wedge \mathit{sub}(K', K)\big)\big]. \end{aligned}$$

We thus obtain via Courcelle's Theorem the following result.

Theorem 9. *The reasoning problems $\alpha \circ_S \beta \models \gamma$ and $\alpha \circ_W \beta \models \gamma$ are fixed-parameter linear with respect to the treewidth. This means they are solvable in time*

$$\mathcal{O}(f(w) \cdot \|\alpha \cup \beta \cup \gamma\|),$$

where f is a function depending only on the treewidth w of the revision scenario (α, β, γ) and $\|\alpha \cup \beta \cup \gamma\|$ denotes the size of an appropriate encoding of $\alpha \cup \beta \cup \gamma$.

For Dalal's operator, we need additional machinery. Let $\mathit{modD}_{\alpha,\beta}(J,K)$ be the MSO-formula given above. Following the notation of [4], $|\Delta|^{min}(\alpha,\beta)$ can be described by a *linear extended monadic second-order extremum problem* $\min_{\psi}|K|$, where $\psi \equiv \exists J\, \mathit{modD}_{\alpha,\beta}(J,K)$. But in order to express the reasoning problem $\alpha \circ_D \beta \models \gamma$ in a straight forward way we would need to take $\min_{\psi}|K|$ as a subformula and make sure that all models of $\alpha \circ_D \beta$ with minimal cardinality are a model of γ as well. However, this is not possible in linear extended monadic second-order logic. In order to still be able to use the extension of Courcelle's Theorem of [4], we decide this problem in two steps. Therefore, we require the following lemma.

Lemma 10. *The reasoning problem $\alpha \circ_D \beta \models \gamma$ holds if and only if $|\Delta|^{min}(\alpha,\beta\wedge\gamma) < |\Delta|^{min}(\alpha,\beta\wedge\neg\gamma)$.*

Proof. Since $\mathit{Mod}(\beta) = \mathit{Mod}(\beta\wedge\gamma) \cup \mathit{Mod}(\beta\wedge\neg\gamma)$, we can express $|\Delta|^{min}(\alpha,\beta)$ as

$$|\Delta|^{min}(\alpha,\beta) = \min(|\Delta|^{min}(\alpha,\beta\wedge\gamma), |\Delta|^{min}(\alpha,\beta\wedge\neg\gamma)).$$

Now assume $\alpha \circ_D \beta \not\models \gamma$. That means, there exists a $J \in \mathit{Mod}(\alpha \circ_D \beta)$ such that $J \notin \mathit{Mod}(\gamma)$. From this and the definition of $\alpha \circ_D \beta$ it follows that $J \in \mathit{Mod}(\beta\wedge\neg\gamma)$. Therefore, there exists $I \in \mathit{Mod}(\alpha)$ such that $|I\Delta J| = |\Delta|^{min}(\alpha,\beta) = |\Delta|^{min}(\alpha,\beta\wedge\neg\gamma)$. Hence, $|\Delta|^{min}(\alpha,\beta\wedge\neg\gamma) \leq |\Delta|^{min}(\alpha,\beta\wedge\gamma)$.

For the other direction assume that $|\Delta|^{min}(\alpha,\beta\wedge\neg\gamma) \leq |\Delta|^{min}(\alpha,\beta\wedge\gamma)$. Therefore, $|\Delta|^{min}(\alpha,\beta) = |\Delta|^{min}(\alpha,\beta\wedge\neg\gamma)$. In other words, there exists $J \in \mathit{Mod}(\alpha \circ_D \beta)$ with $J \in \mathit{Mod}(\neg\gamma)$ and hence $\alpha \circ_D \beta \not\models \gamma$. □

With the help of this lemma and the extension of Courcelle's Theorem by [4], we thus get fixed-parameter tractability.

Theorem 11. *Assuming unit cost for arithmetic operations, the reasoning problem $\alpha \circ_D \beta \models \gamma$ is fixed-parameter linear with respect to the treewidth, i.e., it is solvable in time*

$$\mathcal{O}(f(w)\cdot\|\alpha\cup\beta\cup\gamma\|),$$

where f is a function depending only on the treewidth w of the revision scenario (α,β,γ) and $\|\alpha\cup\beta\cup\gamma\|$ denotes the size of an appropriate encoding of $\alpha\cup\beta\cup\gamma$.

Proof. We use the characterization of $\alpha \circ_D \beta \models \gamma$ given by Lemma 10. Both expressions $|\Delta|^{min}(\alpha,\beta\wedge\gamma)$ and $|\Delta|^{min}(\alpha,\beta\wedge\neg\gamma)$ can be characterized by linear extended MSO extremum problems $\min_{\psi_1}|K|$, respectively $\min_{\psi_2}|K|$. Thereby,

$$\psi_1 \equiv \exists J[\mathit{mod}_{\gamma}(J) \wedge \mathit{modD}_{\alpha,\beta}(J,K)],$$
$$\psi_2 \equiv \exists J[\neg\mathit{mod}_{\gamma}(J) \wedge \mathit{modD}_{\alpha,\beta}(J,K)].$$

By Theorem 5.6 of [4], those can be evaluated in linear time if we assume unit cost for arithmetic operations and if the treewidth of $\alpha\cup\beta\cup\gamma$ is bounded by a fixed constant. The answer to the reasoning problem $\alpha \circ_D \beta \models \gamma$ can now be retrieved by comparing the two outcomes according to Lemma 10. □

4.3 Dynamic Programming Approach for Dalal's Operator

In this section, we show how the theoretical results from Section 4.2 can be put to practice by dynamic programming. We discuss here only the realization for the Dalal-revision operator $\circ_D$ in detail.

We start with an algorithm to decide $\alpha \circ_D \beta \models \gamma$. The very idea of such an algorithm is to associate certain objects (so-called bag assignments) to each node t of a tree decomposition for this problem, such that certain information about the subproblem represented by the subtree rooted at t remains available. Consequently, results for the entire problem can be read off the root of the tree decomposition.

We then make use of our algorithm also for the enumeration problem. Hereby, we traverse the tree decomposition a second time, but starting from the root, where we already have identified certain objects which will allow us to compute the models of the revised knowledge base. However, to guarantee that the enumeration does not provide duplicate models, some additional adjustments in the data structure will be necessary.

4.3.1 Reasoning Problem

For the problem $\alpha \circ_D \beta \models \gamma$, we restrict ourselves here to scenarios where γ is a single atom occurring in $\mathit{var}(\alpha \cup \beta)$ in order to keep the presentation simple. In what follows, we fix $\mathcal{T} = (T, \chi)$ to be a normalized tree decomposition of the incidence graph for $\alpha \cup \beta$. We refer to the root node of T as t_{root}, and we require that the bags of t_{root} and of all leaf nodes of T do not contain any clauses. Such a tree decomposition is easily obtained from a normalized one by suitably adding (CI)- and (CR)-nodes. Additionally, we require that γ appears in $\chi(t_{\mathit{root}})$. Finally, we assume $\alpha \cap \beta = \emptyset$ holds, thus for any clause $c \in \alpha \cup \beta$, its origin $o(c)$ is either α or β. We fix the universe of all atoms occurring in the involved formulas $U = \mathit{var}(\alpha \cup \beta)$.

For a node $t \in T$, we denote by T_t the subtree of T rooted at t. For a set S of elements (either atoms or clauses), $t|_S$ is a shorthand for $\chi(t) \cap S$; moreover, $t{\downarrow}_S$ is defined as $\bigcup_{m \in T_t} m|_S$, and $t{\Downarrow}_S$ abbreviates $t{\downarrow}_S \setminus t|_S$.

Definition 12. A tuple $\vartheta = (t, M_\alpha, M_\beta, C)$, where $t \in T$, $M_\alpha, M_\beta \subseteq t|_U$, and $C \subseteq t|_{\alpha \cup \beta}$ is called a *bag assignment (for node t)*.

Bag assignments for a node t implicitly talk about interpretations over $t{\downarrow}_U$. The following definition makes this more precise.

Definition 13. For a bag assignment $\vartheta = (t, M_\alpha, M_\beta, C)$ and $\varphi \in \{\alpha, \beta\}$, define

$$E_\varphi(\vartheta) = \big\{ K \subseteq t{\downarrow}_U \mid \; K \setminus (t{\Downarrow}_U) = M_\varphi; \; (C \cap \varphi) \cup (t{\Downarrow}_\varphi) = \{ c \in t{\downarrow}_\varphi \mid K \in \mathit{Mod}_{t{\downarrow}_U}(c) \} \big\}.$$

In other words, we associate with a bag assignment $(t, M_\alpha, M_\beta, C)$ all interpretations K that extend M_α in such a way, that all clauses from α appearing in C and in bags below node t are satisfied. The same is done for β. Bag assignments for which such extended interpretations exist for both α and β are of particular interest for us.

Definition 14. A bag assignments ϑ is called *bag model* if and only if $E_\alpha(\vartheta) \neq \emptyset \neq E_\beta(\vartheta)$.

We next rephrase the main features of the definition of $\circ_D$ in terms of bag models and then show that bag models for the root node capture $\circ_D$ as expected.

Definition 15. For any bag model $\vartheta = (t, M_\alpha, M_\beta, C)$, define

$$\begin{aligned} \delta(\vartheta) &= \min\big\{\, |I_\alpha \Delta I_\beta| \mid I_\alpha \in E_\alpha(\vartheta), I_\beta \in E_\beta(\vartheta)\big\}; \quad \text{and} \\ \mathcal{E}(\vartheta) &= \big\{I_\beta \in E_\beta(\vartheta) \mid \exists I_\alpha \in E_\alpha(\vartheta), |I_\alpha \Delta I_\beta| = \delta(\vartheta)\big\}. \end{aligned}$$

Theorem 16. *Let Θ be the set of all bag models ϑ for t_{root}, such that no bag model ϑ' for t_{root} with $\delta(\vartheta') < \delta(\vartheta)$ exists. Then, $Mod(\alpha \circ_D \beta) = \bigcup_{\vartheta \in \Theta} \mathcal{E}(\vartheta)$.*

Proof. ($\subseteq$): Let $J \in Mod(\alpha \circ_D \beta)$. Hence, $J \in Mod(\beta)$ and there exists an $I \in Mod(\alpha)$, such that $|I \Delta J| = |\Delta|^{min}(\alpha, \beta) = k$. Consider $\vartheta = (t_{root}, M_\alpha, M_\beta, \emptyset)$ where $M_\alpha = t_{root}|_I$ and $M_\beta = t_{root}|_J$. Since we assumed that no clauses are stored in $\chi(t_{root})$ and $t_{root}\!\downarrow_U = U$, $J \in Mod(\beta)$ yields that $J \in Mod_{t\downarrow_U}(c)$ holds for each $c \in t_{root}\!\downarrow_\beta = t_{root} \Downarrow_\beta$. The same argumentation applies to I and α. Hence, $I \in E_\alpha(\vartheta)$, $J \in E_\beta(\vartheta)$, and thus ϑ is a bag-model. To show $\vartheta \in \Theta$, it remains to show that no other $\vartheta' \in \Theta$ exists with $\delta(\vartheta') < \delta(\vartheta) \leq k$. Towards a contradiction, suppose such a $\vartheta' = (t_{root}, M'_\alpha, M'_\beta, C')$ exists. By definition, then there exists an $I'_\alpha \in E_\alpha(\vartheta')$ and an $I'_\beta \in E_\beta(\vartheta')$ with $|I'_\alpha \Delta I'_\beta| = \delta(\vartheta')$. Let $\varphi \in \{\alpha, \beta\}$. By definition of $E_\varphi(\cdot)$, we obtain $I'_\varphi \in Mod_{t_{root}\downarrow_U}((C' \cap \varphi) \cup (t_{root} \Downarrow_\varphi))$. Again $C' = \emptyset$ by our assumption for t_{root}, and thus $t_{root} \Downarrow_\varphi = \varphi$. We also know $U = t_{root}\!\downarrow_U$. $I'_\varphi \in Mod(\varphi)$ follows. Hence, we have found models I'_α, I'_β for α, and respectively β, such that $|I'_\alpha \Delta I'_\beta| < k$. A contradiction to our assumption that $|\Delta|^{min}(\alpha, \beta) = k$. The other direction holds by essentially the same arguments.

($\supseteq$): Let $J \in \mathcal{E}(\vartheta)$ for some $\vartheta \in \Theta$. Then, $J \in E_\beta(\vartheta)$ and there exists an $I \in E_\alpha(\vartheta)$, such that $|I \Delta J| = \delta(\vartheta)$. Similar as above, we can conclude from $J \in E_\beta(\vartheta)$ that $J \in Mod(\beta)$ and from $I \in E_\alpha(\vartheta)$ that $I \in Mod(\alpha)$. To show $J \in Mod(\alpha \circ_D \beta)$, suppose, towards a contradiction, this claim does not hold, i.e. there exist $I' \in Mod(\alpha)$, $J' \in Mod(\beta)$, such that $|I' \Delta J'| < \delta(\vartheta)$. One can show that then $\vartheta' = (t_{root}, M, N, \emptyset)$, where $M = t_{root}|_{I'}$, $N = t_{root}|_{J'}$, is a bag model that satisfies $\delta(\vartheta') \leq |I' \Delta J'| < \delta(\vartheta)$. But this is in contradiction to our assumption $\vartheta \in \Theta$. □

We now put our concept of bag models to work also below the root node. Our goal is to characterize bag models ϑ without an explicit computation of $E_\varphi(\vartheta)$. To this end, first note that bag models for leaf nodes t are easily built from all pairs of interpretations over the atoms in the bag $\chi(t)$; also recall that we assumed that no clause is in $\chi(t)$. Thus, formally, the set of all bag models for a leaf node t is given by $\{(t, M, N, \emptyset) \mid M, N \subseteq t|_U\}$. For each such bag model $\vartheta = (t, M, N, \emptyset)$, $\delta(\vartheta) = |M \Delta N|$ is clear. Next, we define a relation $\prec_\mathcal{T}$ between bag assignments, such that all bag models of a node are accordingly linked to bag models of the child(ren) node(s). We thus can propagate, starting from the leaves, bag models upwards the tree decomposition. Afterwards, we will show how $\delta(\vartheta)$ can be treated accordingly.

Definition 17. For bag assignments $\vartheta = (t, M_\alpha, M_\beta, C)$ and $\vartheta' = (t', M'_\alpha, M'_\beta, C')$, we have $\vartheta' \prec_\mathcal{T} \vartheta$ if and only if t has a single child t', and the following properties are satisfied, depending on the node type of t:

1. (c-CR): $M_\alpha = M'_\alpha$, $M_\beta = M'_\beta$, $C = C' \setminus \{c\}$, $c \in C'$;

2. (c-CI): $M_\alpha = M'_\alpha$, $M_\beta = M'_\beta$, and $C = C' \cup \{c\}$ if $M_{o(c)} \in Mod_{t|_U}(c)$; and $C = C'$ otherwise;

3. (a-AR): $M_\alpha = M'_\alpha \setminus \{a\}$, $M_\beta = M'_\beta \setminus \{a\}$, $C = C'$;

4. (a-AI): one of the following cases applies

   - $M_\alpha = M'_\alpha \cup \{a\}$, $N = M'_\beta \cup \{a\}$, $C = C' \cup \{c \in t|_{\alpha\cup\beta} \mid a \in c\}$;
   - $M_\alpha = M'_\alpha \cup \{a\}$, $M_\beta = M'_\beta$, $C = C' \cup \{c \in t|_\alpha : a \in c\} \cup \{d \in t|_\beta : \neg a \in d\}$;
   - $M_\alpha = M'_\alpha$, $M_\beta = M'_\beta \cup \{a\}$, $C = C' \cup \{c \in t|_\alpha : \neg a \in c\} \cup \{d \in t|_\beta : a \in d\}$;
   - $M_\alpha = M'_\alpha$, $M_\beta = M'_\beta$, $C = C' \cup \{c \in t|_{\alpha\cup\beta} \mid \neg a \in c\}$.

For branch nodes, we extend (with slight abuse of notation) $\prec_\mathcal{T}$ to a ternary relation.

Definition 18. Let $\vartheta = (t, M_\alpha, M_\beta, C)$, $\vartheta' = (t', M'_\alpha, M'_\beta, C')$ and $\vartheta'' = (t'', M''_\alpha, M''_\beta, C'')$ be three bag assignments. We relate $(\vartheta', \vartheta'') \prec_\mathcal{T} \vartheta$ if t has two children t' and t'', $M_\alpha = M'_\alpha = M''_\alpha$, $M_\beta = M'_\beta = M''_\beta$, and $C = C' \cup C''$.

Lemma 19. *Let $\vartheta, \vartheta', \vartheta''$ be bag assignments, such that $\vartheta' \prec_\mathcal{T} \vartheta$ (respectively $(\vartheta', \vartheta'') \prec_\mathcal{T} \vartheta$). Then, ϑ is a bag model if and only if ϑ' is a bag model (respectively both ϑ' and ϑ'' are bag models).*

Proof. For the proof, one has to distinguish between the node types. Here, we only show the case where ϑ is a bag assignment for a (c-CI) node t with child t'. In this case, $\vartheta' \prec_\mathcal{T} \vartheta$ holds exactly for assignments of the form $\vartheta = (t, M_\alpha, M_\beta, C)$ and $\vartheta' = (t', M_\alpha, M_\beta, C')$, where $C = C' \cup \{c\}$ if c appears in $\varphi \in \{\alpha, \beta\}$ and M_φ is a partial model of c (i.e., $M_\varphi \in Mod_{t|_U}(c)$); and $C = C'$ otherwise. Consider the case $c \in \alpha$ (the other case is symmetric). We show $E_\alpha(\vartheta) = E_\alpha(\vartheta')$ and $E_\beta(\vartheta) = E_\beta(\vartheta')$. The assertion then follows. We will only sketch a proof for $E_\alpha(\vartheta) = E_\alpha(\vartheta')$.

We have $V = t{\downarrow_U} = t'{\downarrow_U}$, $W = t \Downarrow_U = t' \Downarrow_U$. It is sufficient to show $(C \cap \alpha) \cup t \Downarrow_\alpha = \{d \in t{\downarrow_\alpha} \mid K \in Mod_V(d)\}$ if and only if $(C' \cap \alpha) \cup t' \Downarrow_\alpha = \{d \in t'{\downarrow_\alpha} \mid K \in Mod_V(d)\}$ for each $K \subseteq V$ such that $K \setminus W = M_\alpha$. Fix such a K and note that $t \Downarrow_\alpha = t' \Downarrow_\alpha$. One can show $(C \cap \alpha) = \{d \in t|_\alpha \mid K \in Mod_V(d)\}$ if and only if $(C' \cap \alpha) = \{d \in t'|_\alpha \mid K \in Mod_V(d)\}$ by observing that $K \in Mod_V(c)$ if and only if $M_\alpha \in Mod_{t|_U}(c)$. □

Next, we define recursively a number assigned to bag models ϑ and show that this number in fact matches the minimal distance $\delta(\vartheta)$ defined above.

Definition 20. Let $\vartheta = (t, M_\alpha, M_\beta, C)$ be a bag model. We define

$$\rho(\vartheta) = \begin{cases} |M_\alpha \Delta M_\beta| & \text{if } t \text{ is a leaf node} \\ \rho(\vartheta') \text{ with } \vartheta' \prec_\mathcal{T} \vartheta & \text{if } t \text{ is type (CI) or (CR)} \\ \min\left\{\rho(\vartheta') \mid \vartheta' \prec_\mathcal{T} \vartheta\right\} & \text{if } t \text{ is type (AR)} \\ \min\left\{\rho(\vartheta') \mid \vartheta' \prec_\mathcal{T} \vartheta\right\} & \text{if } t \text{ is type } (a\text{-AI}) \text{ and } a \notin M_\alpha \Delta M_\beta \\ \min\left\{\rho(\vartheta') \mid \vartheta' \prec_\mathcal{T} \vartheta\right\} + 1 & \text{if } t \text{ is type } (a\text{-AI}) \text{ and } a \in M_\alpha \Delta M_\beta \\ \min\left\{\vartheta' \times \vartheta'' \mid (\vartheta', \vartheta'') \prec_\mathcal{T} \vartheta\right\} & \text{if } t \text{ is type (B)} \end{cases}$$

where $\vartheta' \times \vartheta''$ stands for $\rho(\vartheta') + \rho(\vartheta'') - |M_\alpha \Delta M_\beta|$.

Lemma 21. *For any bag model ϑ, $\delta(\vartheta) = \rho(\vartheta)$.*

Example 22. In Figure 4.3, we list all bag models ϑ of the tree decomposition from Example 8 together with values $\rho(\vartheta)$. For instance, leaf node t_8 has bag models for all pairs of interpretations over $\{z\}$. If we go upwards the tree, we observe that bag models for t_9 may additionally contain clause from c_3 satisfied by the respective assignments. In the next node t_{10} only those bag models survive where c_3 was contained, since t_{10} is a (c_3-CR) node. Note that for the root, the bag model $\vartheta = (t_{17}, \emptyset, \emptyset, \emptyset)$ is the one with minimal $\rho(\vartheta)$. It can be checked that $\mathcal{E}(\vartheta) = \{\{\}\}$ as expected (recall that $Mod(\alpha \circ_D \beta) = \{\{\}\}$). ⊣

Theorem 23. *Assuming unit cost for arithmetic operations, $\alpha \circ_D \beta \models \gamma$ can be decided in time $O(f(w) \cdot \|\alpha \cup \beta\|)$, where f is a function depending only on the treewidth w of (α, β) and $\|\alpha \cup \beta\|$ denotes the size of an appropriate encoding of $\alpha \cup \beta$.*

Proof. Lemma 19 suggests the following algorithm: first, we establish the bag models ϑ for leaf nodes together with their value for $\delta(\vartheta) = \rho(\vartheta)$; then we compute all remaining bag models via $\prec_\mathcal{T}$ in a bottom-up manner, and keep track of $\delta(\cdot)$ using the definition of ρ, which is indeed feasible thanks to Lemma 21. As soon as we have the bag models for the root node together with their δ-values we know that bag models in Θ as defined in Theorem 16 characterize the models of $\alpha \circ_D \beta$. Due to our assumption that γ is just a single atom occurring in $\chi(t_{root})$, it remains to check whether for each $(t_{root}, M_\alpha, M_\beta, C) \in \Theta$, $\gamma \in M_\beta$ holds.

The effort needed for processing a leaf node as well as the transition from child to parent nodes only depends on the treewidth but not on $\|\alpha \cup \beta\|$. The size of $\mathcal{T}$ is linear bounded by the size of $\alpha \cup \beta$, thus the desired time bound for our algorithm follows. □

4.3.2 Enumeration Problem

Our reasoning algorithm from Section 4.3.1 gathers the following information along the bottom-up traversal of $\mathcal{T}$.

1. all bag models $\vartheta = (t, M_\alpha, M_\beta, C)$ for all nodes t in $\mathcal{T}$,
2. the minimal distance $\delta(\vartheta)$ between the models $I_\alpha \in E_\alpha(\vartheta)$ and $I_\beta \in E_\beta(\vartheta)$, and

3. the relation $\prec_{\mathcal{T}}$ indicating which bag model(s) ϑ' at the child node t' (respectively at the two child nodes t' and t'') give rise to which bag model ϑ at a node t in $\mathcal{T}$.

In principle, this is all the information needed to enumerate the models in $\alpha \circ_D \beta$ by starting with the bag models ϑ in Θ from Theorem 16 (i.e., the bag models ϑ at the root node t_{root}, such that no other bag model ϑ' at t_{root} with smaller value of $\delta(\cdot)$ exists) and determining $\mathcal{E}(\vartheta)$ for every $\vartheta \in \Theta$ by traversing $\mathcal{T}$ in top-down direction following the $\prec_{\mathcal{T}}$ relation in reversed direction. However, such an enumeration algorithm faces two problems:

1. In Definition 20 (with $\rho(\vartheta) = \delta(\vartheta)$, by Lemma 21) we computed the minimum value attainable by $\rho(\vartheta)$ over all possible bag models ϑ' (respectively pairs $(\vartheta', \vartheta'')$) with $\vartheta' \prec_{\mathcal{T}} \vartheta$ (respectively $(\vartheta', \vartheta'') \prec_{\mathcal{T}} \vartheta$). Hence, when we now follow the $\prec_{\mathcal{T}}$ relation in the reversed direction, we have to make sure that, from any ϑ, we only continue with bag models ϑ' (respectively with pairs $(\vartheta', \vartheta'')$) that actually lead to the minimal value of $\rho(\vartheta)$.

2. For distinct bag models ϑ, ϑ' for any node t, $\mathcal{E}(\vartheta) \cap \mathcal{E}(\vartheta') = \emptyset$ is *not* guaranteed. More precisely, suppose that two bag models $\vartheta = (t, M_\alpha, M_\beta, C)$ and $\vartheta' = (t, M'_\alpha, M'_\beta, C')$ fulfill the condition $M_\beta = M'_\beta$. Then it may well happen that some model I_β is contained both in $E_\beta(\vartheta)$ and $E_\beta(\vartheta')$, such that I_β has minimal distance $\delta(\vartheta)$ from some $I_\alpha \in E_\alpha(\vartheta)$ and also minimal distance $\delta(\vartheta')$ from some $I'_\alpha \in E_\alpha(\vartheta')$. However, for our enumeration algorithm we want to avoid the computation of duplicates since this would, in general, destroy the linear time upper bound on the delay.

The first problem is dealt with below by restricting the relation $\prec_{\mathcal{T}}$ to a subset $\ll_{\mathcal{T}}$ of $\prec_{\mathcal{T}}$. For the second problem, we shall extend the relation $\ll_{\mathcal{T}}$ on bag models to a relation on sets of bag models. We start with the definition $\ll_{\mathcal{T}}$ on bag models. Let us introduce some additional notation first: We identify the components of a bag assignment $\vartheta = (t, M_\alpha, M_\beta, C)$ as $\vartheta_{node} = t$; $\vartheta_\alpha = M_\alpha$; $\vartheta_\beta = M_\beta$; and $\vartheta_{clause} = C$. If Θ is a set of bag assignments such that ϑ_β is identical for all $\vartheta \in \Theta$, then we write Θ_β to denote ϑ_β for any $\vartheta \in \Theta$. Finally, we write $\mathcal{E}(\Theta)$ as a short-hand for $\bigcup_{\vartheta \in \Theta} \mathcal{E}(\vartheta)$.

Definition 24. Let ϑ, ϑ', and optionally, ϑ'' be bag models, such that ϑ'_{node} (and, optionally, also ϑ''_{node}) is a child of $t = \vartheta_{node}$. We define $\vartheta' \ll_{\mathcal{T}} \vartheta$ (respectively $(\vartheta', \vartheta'') \ll_{\mathcal{T}} \vartheta$) if and only if $\vartheta' \prec_{\mathcal{T}} \vartheta$ (respectively $(\vartheta', \vartheta'') \prec_{\mathcal{T}} \vartheta$) and one of the following conditions is fulfilled:

(i) t is of type (CI) or (CR);

(ii) t is of type (AR) and $\delta(\vartheta) = \delta(\vartheta')$;

(iii) t is of type (a-AI), $a \in \vartheta_\alpha$ if and only if $a \in \vartheta_\beta$, and $\delta(\vartheta) = \delta(\vartheta')$;

(iv) t is of type (a-AI), $a \notin \vartheta_\alpha$ if and only if $a \in \vartheta_\beta$, and $\delta(\vartheta) = \delta(\vartheta') + 1$; or

(v) t is of type (B) and $\delta(\vartheta) = \delta(\vartheta') + \delta(\vartheta'') - |\vartheta_\alpha \Delta \vartheta_\beta|$.

We now extend $\ll_{\mathcal{T}}$ from a relation on bag models to a relation on sets Θ of bag models ϑ (with identical component ϑ_β). By slight abuse of notation, we reuse the same symbol $\ll_{\mathcal{T}}$.

Definition 25. Let $\Theta \neq \emptyset$ be a set of bag models for $t \in \mathcal{T}$, such that for all $\vartheta, \vartheta' \in \Theta$, $\vartheta_\beta = \vartheta'_\beta$.

(i) Suppose that t is either of type (CI), (CR) or of type (a-AI) with $a \notin \Theta_\beta$. Then we define $\Theta' \ll_{\mathcal{T}} \Theta$ for $\Theta' = \{\vartheta' \mid \vartheta'_\beta = \Theta_\beta$ and $\vartheta' \ll_{\mathcal{T}} \vartheta$ for some $\vartheta \in \Theta\}$.

(ii) Suppose that t is of type (a-AI) with $a \in \Theta_\beta$. Then we define $\Theta' \ll_{\mathcal{T}} \Theta$ for $\Theta' = \{\vartheta' \mid \vartheta'_\beta = \Theta_\beta \setminus \{a\}$ and $\vartheta' \ll_{\mathcal{T}} \vartheta$ for some $\vartheta \in \Theta\}$.

(iii) Suppose that t is of type (a-AR). Then we define $\Theta'_1 \ll_{\mathcal{T}} \Theta$ and $\Theta'_2 \ll_{\mathcal{T}} \Theta$ for $\Theta'_1 = \{\vartheta' \mid \vartheta'_\beta = \Theta_\beta$ and $\vartheta' \ll_{\mathcal{T}} \vartheta$ for some $\vartheta \in \Theta\}$ and $\Theta'_2 = \{\vartheta' \mid \vartheta'_\beta = \Theta_\beta \cup \{a\}$ and $\vartheta' \ll_{\mathcal{T}} \vartheta$ for some $\vartheta \in \Theta\}$.

(iv) Suppose that t is of type (B). Then we define $\Theta' \ll_{\mathcal{T}} \Theta$ for $\Theta' = \{\vartheta' \mid \exists \vartheta''$ with $(\vartheta', \vartheta'') \ll_{\mathcal{T}} \vartheta$ for some $\vartheta \in \Theta\}$.

Moreover, for every $\hat{\Theta} \subseteq \Theta'$ with $\Theta' \ll_{\mathcal{T}} \Theta$, we define $(\hat{\Theta}, \Theta'') \ll_{\mathcal{T}} \Theta$, where $\Theta'' = \{\vartheta'' \mid \exists \vartheta \in \Theta$ and $\exists \vartheta' \in \hat{\Theta}$ with $(\vartheta', \vartheta'') \ll_{\mathcal{T}} \vartheta\}$.

The following lemma states that every model in $\mathcal{E}(\Theta)$ at node t can be computed via $\mathcal{E}(\Theta')$ (and optionally $\mathcal{E}(\Theta'')$) at the child node(s) of t and, conversely, that every element in $\mathcal{E}(\Theta')$ (and optionally $\mathcal{E}(\Theta'')$) can indeed be extended to a model of $\mathcal{E}(\Theta)$.

Lemma 26. *Let $t \in \mathcal{T}$ and let Θ be a non-empty set of bag models for t, such that for all $\vartheta, \vartheta' \in \Theta$, $\vartheta_\beta = \vartheta'_\beta$. Then the following properties hold:*

(i) Suppose t is of type (CI), (CR), (AR), or of type (a-AI), such that $a \notin \Theta_\beta$. Then $I \in \mathcal{E}(\Theta)$ if and only if $I \in \mathcal{E}(\Theta')$, such that $\Theta' \ll_{\mathcal{T}} \Theta$.

(ii) Suppose t is of type (a-AI), such that $a \notin \Theta_\beta$. Then $I \in \mathcal{E}(\Theta)$ if and only if $(I \setminus \{a\}) \in \mathcal{E}(\Theta')$, such that $\Theta' \ll_{\mathcal{T}} \Theta$.

(iii) Suppose t is of type (B). Then $I \in \mathcal{E}(\Theta)$ if and only if $I = I' \cup I''$ for some $I' \in \mathcal{E}(\hat{\Theta})$ and $I'' \in \mathcal{E}(\Theta'')$, where $\hat{\Theta} \subseteq \Theta'$, $\Theta' \ll_{\mathcal{T}} \Theta$, and $(\hat{\Theta}, \Theta'') \ll_{\mathcal{T}} \Theta$.

For our enumeration algorithm, we start at the root node of $\mathcal{T}$ and first partition the relevant bag models ϑ according to ϑ_β. Formally, let Θ be the set of all bag models ϑ for t_{root}, such that no bag model ϑ' for t_{root} with $\delta(\vartheta') < \delta(\vartheta)$ exists. Then we partition Θ into $\Theta_1, \ldots, \Theta_t$, such that for each $\vartheta, \vartheta' \in \Theta_i$, $\vartheta_\beta = \vartheta'_\beta$, and for each $\vartheta \in \Theta_i$, $\vartheta' \in \Theta_j$ with $i \neq j$, $\vartheta_\beta \neq \vartheta'_\beta$. Clearly, the sets $\mathcal{E}(\Theta_i)$ are pairwise disjoint. Hence, no duplicates will be computed when we compute $\mathcal{E}(\Theta_1), \ldots, \mathcal{E}(\Theta_t)$ separately.

Given a set Θ of bag models, we compute $\mathcal{E}(\Theta)$ by implementing an appropriate *iterator* for every node t in $\mathcal{T}$. The iterator provides functions `open`, `get_current`, and `get_next`. In addition, other functions like `close` (to deallocate state information) are needed which we do not discuss here.

The open *function.* The open function serves to initialize the state information at each node of a given subtree of $\mathcal{T}$. For instance, it is convenient (in particular, for branch nodes) to store in a Boolean flag first whether get_next() has not yet been called since the initialization with the call of function open. Moreover, for an (AR) node, there can exist two sets Θ_1 and Θ_2 with $\Theta_i \ll_{\mathcal{T}} \Theta$. We have to store in the state of the (AR) node, which one of these two sets is currently being processed at the child node.

The open function takes as input a set Θ of bag models ϑ with identical ϑ_β and recursively calls open(Θ') with $\Theta' \ll_{\mathcal{T}} \Theta$. If the current node is of type (CI), (CR), or (AI), then Θ' is unique. Likewise, Θ' is unique for the first child of a branch node. In case of an (AR) node, Θ' corresponds to Θ_1 from Definition 25, case (iii), provided that it is non-empty. Otherwise, $\Theta' = \Theta_2$ is chosen. The children of a branch node are treated asymmetrically by the $\ll_{\mathcal{T}}$-relation and, hence, also by the open function. In the first place, we only compute Θ' with $\Theta' \ll_{\mathcal{T}} \Theta$ for the *first child* of every branch node. As we shall explain below, the function get_next() computes the set of assignments $\mathcal{E}(\Theta)$, returning one such assignment per call. For branch nodes, we thus compute for the first child the set $\mathcal{E}(\Theta')$ with $\Theta' \ll_{\mathcal{T}} \Theta$. For each assignment I' thus returned, get_next() also yields $\Gamma' = \{\hat{\vartheta} \mid I' \in \mathcal{E}(\hat{\vartheta})\} \subseteq \Theta'$. Then the subtree rooted at the second child node is processed with Θ'', such that $(\Gamma', \Theta'') \ll_{\mathcal{T}} \Theta$. Hence, for every assignment I', we have to compute Θ'' (which is uniquely determined by Γ' and Θ) and call open(Θ''), before we can retrieve the assignments I'' in $\mathcal{E}(\Theta'')$ with get_next().

The get_next *and* get_current *function.* Suppose that a node t in $\mathcal{T}$ has been initialized by a call of open(Θ) with $\vartheta_{node} = t$ for every $\vartheta \in \Theta$. Then we can call get_next() for this node in order to retrieve the first respectively the next assignment I in $\mathcal{E}(\Theta)$. In addition to the assignment I, the get_next function also provides a set $\Gamma \subseteq \Theta$ as output, such that $\Gamma = \{\hat{\vartheta} \mid I \in \mathcal{E}(\hat{\vartheta})\}$. As we have already seen, this set Γ of bag assignments is needed when we encounter a branch node on our way back to the root, in order to determine the set Θ'' for the second child. The get_current function is called (for the first child of a branch node) to retrieve once again the result from the previous call to get_next. If no next assignment exists, then get_next returns the value "Done".

In order to compute the first respectively next assignment I, we traverse $\mathcal{T}$ downwards by recursive calls of get_next() until the leaves are reached. In the leaves, we start with the assignment $I = \Theta_\beta$ and also set $\Gamma = \Theta$. This assignment I and the set Γ are now updated on the way back to the root. The only modifications to I are in fact done when we are at an (a-AI) node or at a (B) node. For (AI) nodes, we add a in case a is added to the respective Θ_β. For (B) nodes, we set $I = I' \cup I''$, where I' (respectively I'') is the assignment returned by the call of get_next() for the first (respectively second) child node. In Figure 4.4 we give the pseudo-code of the get_next function in case of a branch node. For the remaining node types, the implementation of get_next is even simpler.

Theorem 27. *Given formulas α and β, the models in $Mod(\alpha \circ_D \beta)$ can be computed with delay $O(f(w) \cdot |\alpha \cup \beta|)$, where f is a function depending only on the treewidth w of (α, β).*

Proof. By our definition of $\ll_{\mathcal{T}}$ on sets and by Lemma 26, we can be sure that (1) every assignment in $Mod(\alpha \circ_D \beta)$ is eventually computed by our iterator-based implementation via recursive calls of get_next and (2) no assignment is computed twice. Indeed, our set-based definition of

the $\ll_{\mathcal{T}}$-relation groups together bag models ϑ with identical ϑ_β and, for any bag models ϑ' with $\vartheta_\beta \neq \vartheta'_\beta$, we trivially have $\mathcal{E}(\vartheta) \cap \mathcal{E}(\vartheta') = \emptyset$.

As far as the complexity is concerned, note that the recursive calls of the open function come down to a top-down traversal of $\mathcal{T}$. (In fact, by the asymmetric treatment of the children of a branch node, open is only called for the nodes along the left-most path in $\mathcal{T}$.) Similarly, each call of get_next and get_current leads to a single traversal of the subtree below the current node t. The work actually carried out inside each call is independent of the size of $\mathcal{T}$. Hence, in total, we end up with a time bound that is linear in the size of $\mathcal{T}$ and, hence, in the size of α and β. □

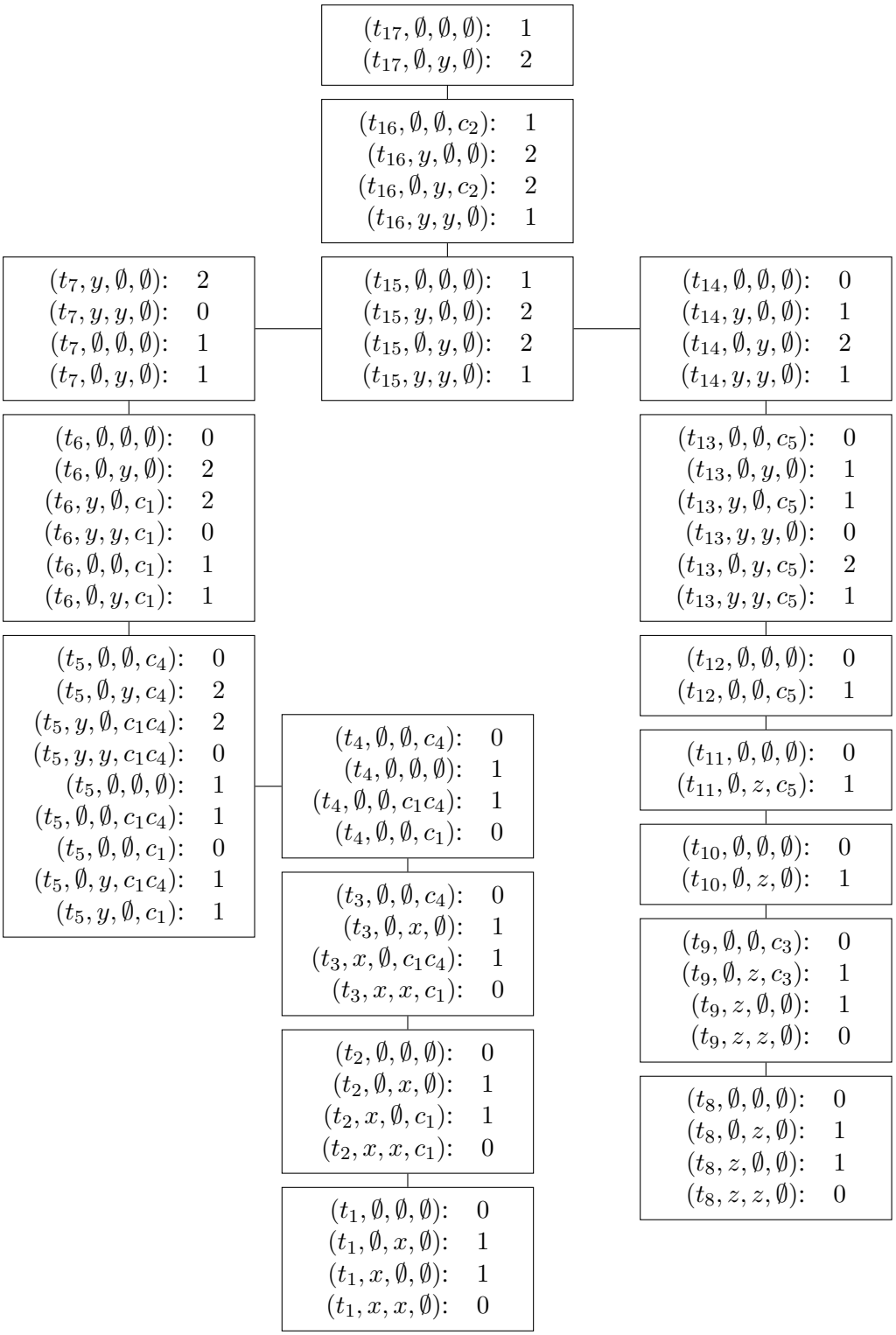

Figure 4.3: All bag models for the tree decomposition from Example 8.

Function get_next for a branch node t with child nodes t', t''
Let Θ be the input parameter of the previous call of function open
if first *then*
first = False;
$(I', \Gamma') = t'$.get_next();
Let Θ'' such that $(\Gamma', \Theta'') \ll_{\mathcal{T}} \Theta$;
t''.open(Θ'')
$(I'', \Gamma'') = t''$.get_next()
else
$(I', \Gamma') = t'$.get_current()
$(I'', \Gamma'') = t''$.get_next()
if (I'', Γ'') = undefined (i.e., the call of t''.get_next() returned "Done") *then*
$(I', \Gamma') = t'$.get_next();
if (I', Γ') = undefined (i.e., the call of t'.get_next() returned "Done") *then*
return "Done"
endif
Let Θ'' such that $(\Gamma', \Theta'') \ll_{\mathcal{T}} \Theta$;
t''.open(Θ'')
$(I'', \Gamma'') = t''$.get_next()
endif
endif
return $(I' \cup I'', \{\vartheta \in \Theta \mid \exists \gamma' \in \Gamma' \text{ and } \exists \gamma'' \in \Gamma'', \text{ such that } (\gamma', \gamma'') \ll_{\mathcal{T}} \vartheta\})$

Figure 4.4: Function get_next() for a branch node.

CHAPTER 5

Answer-Set Programming

This chapter is mainly based on joint work with Reinhard Pichler, Stefan Szeider, and Stefan Woltran, and appeared in the proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning (KR 2010) [91] as well as in the journal Theory and Practice of Logic Programming (TPLP) [92].

The implementation and evaluation presented in Section 5.7 is based on joint work with Michael Morak, Nysret Musliu, Reinhard Pichler, and Stefan Woltran. It appeared partly in the proceedings of the Twelfth European Conference on Logics in Artificial Intelligence (JELIA 2010) [76], in the proceedings of the Twenty-Third International Conference on Tools with Artificial Intelligence (ICTAI 2011) [74], as well as in the proceedings of the Sixth International Conference on Learning and Intelligent Optimization (LION 6) [75].

The first part of this chapter is devoted to applying the notion of treewidth to answer-set programs with cardinality or weight constraints and to identify tractable fragments. It will turn out that the straightforward application of treewidth to this class of programs does not suffice to obtain tractability. However, by imposing further restrictions, tractability can be achieved.

In the main part of this chapter, we restrict ourselves to *normal logic programs with cardinality constraints* (PCCs) or *weight constraints* (PWCs). Clearly, all common algorithmic tasks related to PCCs and PWCs – like checking the consistency of a program – are intractable, since intractability even holds without such constraints.

5.1 Answer-Set Programming Background

5.1.1 Disjunctive Logic Programs

A (propositional) *disjunctive logic program* (program, for short) is a pair $\Pi = (\mathcal{A}, \mathcal{R})$, where $\mathcal{A}$ is a set of propositional atoms and $\mathcal{R}$ is a set of rules of the form:

$$a_1 \vee \cdots \vee a_l \leftarrow a_{l+1}, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n \tag{5.1}$$

where $n \geq 1$, $n \geq m \geq l$ and $a_i \in \mathcal{A}$ for all $1 \leq i \leq n$. Thereby "$\neg$" symbolizes default negation. We omit strong negation as considered in [6] but our results easily extend to programs with strong negation. A rule $r \in \mathcal{R}$ of the form (5.1) consists of a head $H(r) = \{a_1, \ldots, a_l\}$ and a body $B(r) = B^+(r) \cup B^-(r)$, given by $B^+(r) = \{a_{l+1}, \ldots, a_m\}$ and $B^-(r) = \{a_{m+1}, \ldots, a_n\}$. A set $M \subseteq \mathcal{A}$ is a called a model of r, if $B^+(r) \subseteq M \wedge B^-(r) \cap M = \emptyset$ implies that $H(r) \cap M \neq \emptyset$. We denote the set of models of r by $Mod(r)$ and the models of a program $\Pi = (\mathcal{A}, \mathcal{R})$ are given by $Mod(\Pi) = \bigcap_{r \in \mathcal{R}} Mod(r)$.

The reduct Π^I of a program Π w.r.t. an interpretation $I \subseteq \mathcal{A}$ is given by $(\mathcal{A}, \{r^I : r \in \mathcal{R}, B^-(r) \cap I = \emptyset)\})$, where r^I is r without the negative body, i.e., $H(r^I) = H(r)$, $B^+(r^I) = B^+(r)$, and $B^-(r^I) = \emptyset$. Following [45], $M \subseteq \mathcal{A}$ is an *answer set* of a program $\Pi = (\mathcal{A}, \mathcal{R})$ if $M \in Mod(\Pi)$ and for no $N \subset M$, $N \in Mod(\Pi^M)$.

We consider here the class of *head-cycle free programs* (HCFPs) as introduced in [6]. We first recall the concept of *(positive) dependency graphs*. A dependency graph of a program $\Pi = (\mathcal{A}, \mathcal{R})$ is given by $\mathcal{G} = (V, E)$, where $V = \mathcal{A}$ and $E = \{(p, q) \mid r \in \mathcal{R}, p \in B^+(r), q \in H(r)\}$. A program $\Pi = (\mathcal{A}, \mathcal{R})$ is called head-cycle free if its dependency graph does not contain a directed cycle going through two different atoms which jointly occur in the head of a rule in $\mathcal{R}$.

Example 28. The following disjunctive program solves the 3-colorability problem for the graph that is represented by the following facts: $\mathrm{vertex}(a)$, $\mathrm{vertex}(b)$ and $\mathrm{edge}(a, b)$.

$$\begin{aligned}
\mathrm{red}(a) \vee \mathrm{green}(a) \vee \mathrm{blue}(a) &\leftarrow \top. \\
\mathrm{red}(b) \vee \mathrm{green}(b) \vee \mathrm{blue}(b) &\leftarrow \top. \\
\bot &\leftarrow \mathrm{red}(a), \mathrm{red}(b). \\
\bot &\leftarrow \mathrm{green}(a), \mathrm{green}(b). \\
\bot &\leftarrow \mathrm{blue}(a), \mathrm{blue}(b).
\end{aligned}$$

⊣

To build tree decompositions for programs, we use incidence graphs. For program $\Pi = (\mathcal{A}, \mathcal{R})$, such a graph is given by $\mathcal{G} = (V, E)$, where $V = \mathcal{A} \cup \mathcal{R}$ and E is the set of all pairs (a, r) with an atom $a \in \mathcal{A}$ appearing in a rule $r \in \mathcal{R}$.

5.1.2 Weight Constraint Programs

A *program with weight constraints* (PWC) is a triple $\Pi = (A, \mathcal{C}, \mathcal{R})$, where A is a set of *atoms*, $\mathcal{C}$ is a set of *weight constraints* (or *constraints* for short), and $\mathcal{R}$ is a set of *rules*. Each constraint $c \in \mathcal{C}$ is a triple (S, l, u) where S is a set of *weight literals* over A representing a clause and $l \leq u$ are nonnegative integers, the lower and upper bound. A weight literal over A is a pair (a, j) or $(\neg a, j)$ for $a \in A$ and $1 \leq j \leq u + 1$, the weight of the literal. Unless stated otherwise, we assume that the bounds and weights are given in binary representation. For a constraint $c = (S, l, u) \in \mathcal{C}$, we write $Cl(c) := S$, $l(c) := l$, and $u(c) := u$. Moreover, we use $a \in Cl(c)$ and $\neg a \in Cl(c)$ as an abbreviation for $(a, j) \in Cl(c)$ and $(\neg a, j) \in Cl(c)$, respectively, for an arbitrary j. A rule $r \in \mathcal{R}$ is a pair (h, b), where $h \in \mathcal{C}$ is the head and $b \subseteq \mathcal{C}$ is the body. We write $H(r) := h$ and $B(r) := b$.

We denote the size of a reasonable encoding of program Π by $\|\Pi\|$ and call it the size of Π. Unless otherwise stated, weights are assumed to be encoded in binary notation. For instance, the following would do.

$$\|\Pi\| = |A| + \sum_{(S,l,u)\in\mathcal{C}} (1 + \log l + \log u + \sum_{(lit,j)\in S} (1 + \log j)) + \sum_{(h,b)\in\mathcal{R}} (1 + |b|).$$

Given a constraint $c \in \mathcal{C}$ and an interpretation $I \subseteq A$ over atoms A, we denote the weight of c in I by

$$W(c, I) = \sum_{\substack{(a,j)\in Cl(c)\\ a\in I}} j \; + \sum_{\substack{(\neg a,j)\in Cl(c)\\ a\notin I}} j.$$

I is a model of c, denoted by $I \models c$, if $l(c) \leq W(c, I) \leq u(c)$. For a set $C \subseteq \mathcal{C}$, $I \models C$ if $I \models c$ for all $c \in C$. Moreover, C is a model of a rule $r \in \mathcal{R}$, denoted by $C \models r$, if $H(r) \in C$ or $B(r) \not\subseteq C$. I is a model of program Π (denoted by $I \models \Pi$) if $\{c \in \mathcal{C} \mid I \models c\} \models r$ for all $r \in \mathcal{R}$. If the lower bound of a constraint $c \in \mathcal{C}$ is missing, we assume $l(c) = 0$. If the upper bound is missing, $I \models c$ if $l(c) \leq W(c, I)$. A *program with cardinality constraints* (PCC) can be seen as a special case of a PWC, where each literal has weight 1.

Example 29. Consider the following system configuration problem, where one has to choose among the given parts: p_1 : 4000\$, p_2 : 2000\$, and p_3 : 1000\$ such that the total cost is $\leq$ 5000\$. Thereby one of $\{p_1, p_2\}$ has to be selected and p_3 requires p_2.

This scenario can be represented by the PWC

$$\Pi_{Ex} = (\{p_1, p_2, p_3\}, \{c_1, c_2, c_3, c_4\}, \{r_1, r_2, r_3\})$$

with weight constraints

$$\begin{aligned} c_1 &= (\{(p_1, 4), (p_2, 2), (p_3, 1)\}, 0, 5) \\ c_2 &= (\{(p_1, 1), (p_2, 1)\}, 1, 2) \\ c_3 &= (\{(p_2, 1)\}, 1, 1) \\ c_4 &= (\{(p_3, 1)\}, 1, 1) \end{aligned}$$

and rules

$$\begin{aligned} r_1 &= (c_1, \emptyset) \\ r_2 &= (c_2, \emptyset) \\ r_3 &= (c_3, \{c_4\}). \end{aligned}$$

⊣

Stable model semantics for programs with weight constraints. Let $\Pi = (A, \mathcal{C}, \mathcal{R})$ be a PWC and let $I \subseteq A$ be an interpretation. Following Niemelä et al. [82], the reduct c^I of a

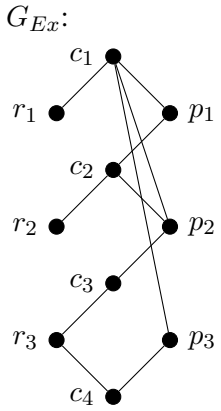

Figure 5.1: Incidence graph G_{Ex} of Example 30.

constraint $c \in \mathcal{C}$ with respect to I is obtained by removing all negative literals and the upper bound from c, and replacing the lower bound by

$$l' = \max(0,\ l(c) - \sum_{\substack{(\neg a, j) \in Cl(c) \\ a \notin I}} j).$$

The reduct Π^I of program Π with respect to I can be obtained by first removing each rule $r \in \mathcal{R}$ which contains a constraint $c \in B(r)$ with $W(c, I) > u(c)$. Afterwards, each remaining rule r is replaced by the set of rules[1] (h, b), where $h \in I \cap Cl(H(r))$ and $b = \{c^I \mid c \in B(r)\}$, i.e., the head of the new rules is an atom instead of a constraint. Interpretation I is called a *stable model* (or *answer set*) of Π if I is a model of Π and there exists no $J \subset I$ such that J is a model of Π^I. The set of all answer sets of Π is denoted by $\mathcal{AS}(\Pi)$. The *consistency problem* for PWCs is to decide whether $\mathcal{AS}(\Pi) \neq \emptyset$.

Treewidth and constraint-width of PWCs. To build tree decompositions for programs, we use *incidence graphs*. Recall the definition of incidence graphs for propositional formulas from Section 2.3. For a PWC $\Pi = (A, \mathcal{C}, \mathcal{R})$, such a graph has vertex set $A \cup \mathcal{C} \cup \mathcal{R}$. There is an edge between $a \in A$ and $c \in \mathcal{C}$ if $a \in Cl(c)$ or $\neg a \in Cl(c)$, and there is an edge between $c \in \mathcal{C}$ and $r \in \mathcal{R}$ if $c \in \{H(r)\} \cup B(r)$. The treewidth of Π, denoted by $tw(\Pi)$, is the treewidth of its incidence graph. The *constraint-width* of Π, denoted by $cw(\Pi)$, is the largest (lower or upper) bound occurring in the constraints of $\mathcal{C}$ (or 0 if there are no bounds).

Example 30. Recall the PWC Π_{Ex} from Example 29. The incidence graph G_{Ex} of Π_{Ex} as well as a normalized tree decomposition $\mathcal{T}_{Ex}$ for Π_{Ex} of width 2 are depicted in Figures 5.1 and 5.2. ⊣

[1] With some abuse of notation, we sometimes write for an atom h, (h, b) as a shorthand for the rule $((\{(h, 1)\}, 1, 1), b)$.

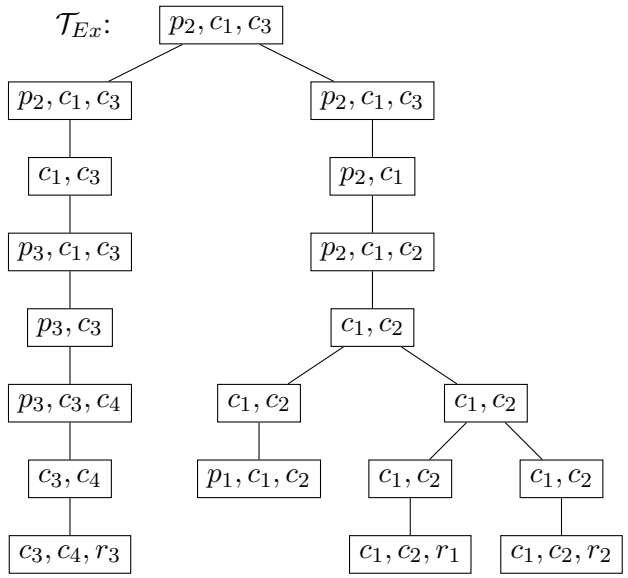

Figure 5.2: Tree decomposition $\mathcal{T}_{Ex}$ of Example 30.

5.2 NP-Completeness

In [92] it was shown that the parameter treewidth alone is not enough to lead to fixed-parameter tractability for the consistency problem of programs with weight constraints. Indeed, this parameterization only leads to para-NP-hardness.

Theorem 31 ([92])**.** *The consistency problem for PWCs is NP-complete already for programs having treewidth 1.*

The NP-hardness is shown by reduction from the NP-complete problem PARTITION. An instance of PARTITION is a collection of positive integers $X = \{x_1, \ldots, x_n\}$ encoded in binary and the question is whether there exists a set $I \subseteq \{1, \ldots, n\}$ such that

$$\sum_{i \in I} x_i = \sum_{i \notin I} x_i.$$

The problem PARTITION is only "weakly NP-hard". This means, its NP-hardness depends on the binary encoding of the given integers. For unary encoding this no longer holds. Accordingly, the reduction provides only weak NP-hardness for the consistency problem of PWCs of bounded treewidth. In fact, we will prove in Section 5.5 that if we assume the weights to be given in unary the consistency problem is feasible in (non-uniform) polynomial time for PWCs of bounded treewidth. The non-uniformity refers to the fact that the degree of the polynomial will depend on the treewidth.

5.3 Linear-Time Tractability

Since the parameter treewidth alone is not sufficient to achieve fixed-parameter tractability we consider as an additional parameter the value of the largest (lower or upper) bound occurring in the constraints of the program. We call this parameter *constraint-width*.

Indeed, it turns out that the combination of the two parameters leads to fixed-parameter tractability. We showed the following result in [92].

Theorem 32 ([92])**.** *The consistency problem for PWCs can be solved in linear time for instances whose treewidth and constraint-width are bounded by constants.*

We proved this result by considering the incidence graph with additional labels for edges that indicate the polarity of literals and labels to distinguish between the head and the body of rules. Additional labels encode the weight of the literals. Since the constraint-width is bounded, the number of such labels is bounded as well. Furthermore, we used vertex labels to encode the bounds of constraints as well as three distinguished labels to differentiate between vertices that represent atoms, clauses, and rules. This graph can be constructed in linear time.

The consistency problem of PWCs is then encoded as a monadic second-order property over such graphs. Theorem 32 now follows directly by Courcelle's Theorem.

5.4 Dynamic Programming Approach

Recently, Jakl, Pichler, and Woltran [57] presented an algorithm for answer-set programming that is based on dynamic programming. It works for programs without cardinality or weight constraints, but possibly with disjunction in the head of the rules. One way to obtain a dynamic programming algorithm for PCCs is to try to extend the algorithm of Jakl et al. [57] by methods to handle the cardinality constraints. In principle, this should be feasible. However, computationally, this approach has a serious drawback, namely, the aforementioned algorithm is tractable for bounded treewidth, but it is *double exponential* with respect to the treewidth (basically this is due to the handling of disjunctions).

Here our goal is to present an algorithm that is only *single exponential* with respect to the treewidth. In order to achieve this goal, we have to manipulate a slightly more complicated data structure along the bottom-up traversal of the tree decomposition. In particular, we have to deal with orderings on the atoms in a model.

To this end, we need an alternative characterization of stable models. By slightly rephrasing the result by Liu [70], we can characterize answer sets of PCCs as follows:

Proposition 33. *Given a PCC* $\Pi = (A, \mathcal{C}, \mathcal{R})$*,* $M \subseteq A$ *is an answer set (stable model) of* Π *if and only if the following conditions are jointly satisfied:*

- M *is a model of* Π*, i.e.,* $M \models \Pi$*,*
- *there exists a strict linear order* $<$ *over* M *such that for each atom* $a \in M$*, there exists a rule* $r \in \mathcal{R}$ *with*

  (R1) $a \in Cl(H(r))$*,*

(R2) $M \models B(r)$,

(R3) for each $c \in B(r)$,

$$l(c) \leq |\{b \in Cl(c) | b < a\} \cup \{\neg b \in Cl(c) | b \in A \setminus M\}|\,.$$

Since the handling of linear orders is crucial for utilizing the above characterization, we will fix some notation first. We denote by $[x_1, x_2, \ldots, x_n]$ a (strict) linear order $x_1 < x_2 < \ldots < x_n$ on a set $X = \{x_1, \ldots, x_n\}$. Moreover, $[\![X]\!]$ denotes the set of all possible linear orders over X. Two linear orders $[x_1, \ldots, x_n]$ and $[y_1, \ldots, y_m]$ are called *inconsistent* if there are x_i, x_j, y_k, y_l such that $x_i < x_j$, $y_k < y_l$, $x_i = y_l$ and $x_j = y_k$. Otherwise, we call them *consistent*.

Given two consistent linear orders $[x_1, \ldots, x_n] \in [\![X]\!]$ and $[y_1, \ldots, y_m] \in [\![Y]\!]$, we denote by $[x_1, \ldots, x_n] + [y_1, \ldots, y_m] = S$ the set of their possible *combinations*. S contains those linear orders $[z_1, \ldots, z_p] \in [\![X \cup Y]\!]$ such that for every pair $x_i < x_j$ (respectively $y_i < y_j$), there exists $z_k < z_l$ with $z_k = x_i$ and $z_l = x_j$ (respectively $z_k = y_i$ and $z_l = y_j$). Note that in general there exists more than one possible combination. Furthermore, we denote by $[x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n] - [x_i]$ the linear order $[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$.

Throughout the whole section, let $\mathcal{T} = (T, \chi)$ be a normalized tree decomposition of a PCC $\Pi = (A, \mathcal{C}, \mathcal{R})$. We present a dynamic programming algorithm, traversing $\mathcal{T}$ in bottom-up direction in order to compute whether Π admits an answer set. Ultimately, we will state properties about subtrees of $\mathcal{T}$ and inductively add more and more nodes until we get a statement about the whole tree. To this end, the following notions, which are similar to the ones used in Section 4.3.1, become handy. Given a node $t \in T$, we denote by T_t the subtree of T rooted at t. For a set $S \subseteq A \cup \mathcal{C} \cup \mathcal{R}$, $t|_S$ is a shorthand for $\chi(t) \cap S$. Moreover, $t{\downarrow_S} := \bigcup_{m \in T_t} m|_S$ and $t{\Downarrow_S} := t{\downarrow_S} \setminus t|_S$.

Since the scope of a solution will always be limited to a subtree of the whole tree decomposition, the notion of a model has to be refined with respect to a universe $U = t{\downarrow_A}$. To this end, the cardinality of a constraint $c \in \mathcal{C}$ with respect to an interpretation $I \subseteq U$ is given by

$$\Gamma(c, I, U) = |\{b \in Cl(c) \mid b \in I\}| + |\{\neg b \in Cl(c) \mid b \in U \setminus I\}|\,.$$

Then I is a model of c under universe U (denoted by $I \models_U c$) if $l(c) \leq \Gamma(c, I, U) \leq u(c)$. Note that $\models_U$ and $\models$ coincide for $U = A$. Similarly, for a subset of constraints $\mathcal{C}' \subseteq \mathcal{C}$, set $C \subseteq \mathcal{C}'$ is a model of rule $r \in \mathcal{R}$ under restriction $\mathcal{C}'$, denoted by $C \models_{\mathcal{C}'} r$ if $H(r) \in C$ or $B(r) \cap \mathcal{C}' \not\subseteq C$.

In order to facilitate the discussion below, we define the following sum for constraint $c \in \mathcal{C}$, interpretation $I \subseteq U$ over a set of atoms $U \subseteq A$ and linear order $L_<$ containing at least $I \cup \{c\}$:

$$\begin{aligned}\Gamma_<(c, I, U, L_<) \;=\; & |\{b \in Cl(c) \mid b \in I \wedge b < c\}| + \\ & |\{\neg b \in Cl(c) \mid b \in U \setminus I\}|\,.\end{aligned}$$

The following definition helps us to find partial answer sets, limited to the scope of a subtree of $\mathcal{T}$.

Definition 34. A *partial solution* (for node $t \in T$) is a tuple $\hat{\vartheta} = (t, \hat{M}, \hat{C}, \hat{R}, \hat{L}_<, \hat{\gamma}, \hat{\gamma}_<, \hat{\Delta})$ with interpretation $\hat{M} \subseteq t{\downarrow_A}$, satisfied constraints $\hat{C} \subseteq t{\downarrow_{\mathcal{C}}}$, satisfied rules $\hat{R} \subseteq t{\downarrow_{\mathcal{R}}}$, linear order $\hat{L}_< \in [\![\hat{M} \cup \hat{C} \cup t{\downarrow_{\mathcal{R}}}]\!]$, cardinality functions $\hat{\gamma} : t{\downarrow_{\mathcal{C}}} \to \mathbb{N}$ and $\hat{\gamma}_< : \hat{C} \to \mathbb{N}$, and derivation

witness $\hat{\Delta} = (\hat{\delta}_R, \hat{\delta}_M, \hat{\delta}_h, \hat{\delta}_b, \hat{\sigma})$ with derivation rules $\hat{\delta}_R \subseteq t{\downarrow_{\mathcal{R}}}$, derived atoms $\hat{\delta}_M \subseteq \hat{M}$, derivation head constraints $\hat{\delta}_h \subseteq \hat{C}$, derivation body constraints $\hat{\delta}_b \subseteq \hat{C}$, and check function $\hat{\sigma} : \hat{\delta}_h \to \{0, 1\}$ such that the following conditions are jointly satisfied:

1. $\hat{C} \cap t{\Downarrow_{\mathcal{C}}} = \{c \in t{\Downarrow_{\mathcal{C}}} |\ \hat{M} \models_{t{\downarrow_A}} c\}$
2. $\hat{R} = \{r \in t{\downarrow_{\mathcal{R}}} |\ \hat{C} \models_{t{\downarrow_{\mathcal{C}}}} r\}$ and $t{\Downarrow_{\mathcal{R}}} \subseteq \hat{R}$
3. $\hat{\gamma}(c) = \Gamma(c, \hat{M}, t{\downarrow_A})$ for all $c \in t{\downarrow_{\mathcal{C}}}$
4. $\hat{\gamma}_<(c) = \Gamma_<(c, \hat{M}, t{\downarrow_A}, \hat{L}_<)$ for all $c \in \hat{C}$
5. $\hat{\delta}_M = \{a \in \hat{M} \mid c \in \hat{\delta}_h, a \in Cl(c), a > c\}$ and $\hat{M} \cap t{\Downarrow_A} \subseteq \hat{\delta}_M$
6. $\hat{\delta}_b = \bigcup_{r \in \hat{\delta}_R} B(r)$ and $\hat{\delta}_b \subseteq \hat{C}$
7. $c \in B(r) \Rightarrow r > c$ for all $c \in \hat{\delta}_b$ and $r \in \hat{\delta}_R$
8. $l(c) \leq \hat{\gamma}_<(c)$ for all $c \in \hat{\delta}_b \cap t{\Downarrow_{\mathcal{C}}}$
9. $\hat{\sigma}(c) = 1 \Leftrightarrow \exists r \in \hat{\delta}_R$ with $H(r) = c$ and $c > r$
10. $\hat{\sigma}(c) = 1$ for all $c \in \hat{\delta}_h \cap t{\Downarrow_{\mathcal{C}}}$

The idea of this data structure is that, for some atom, clause, or rule that is no longer "visible" in the current bag but was included in the subtree, the containment in one of the sets of $\hat{\vartheta}$ is strictly what one would expect from an answer set. While for elements that are still visible, this containment does not have to fulfill that many conditions and can be seen as some sort of "guess".

For example, $\hat{C} \cap t{\Downarrow_{\mathcal{C}}}$, the set of constraints in $\hat{C}$ that are no longer visible, indeed contains exactly the constraints that are satisfied under interpretation $\hat{M}$, this means $\{c \in t{\Downarrow_{\mathcal{C}}} |\ \hat{M} \models_{t{\downarrow_A}} c\}$. On the other hand $\hat{C} \cap t|_{\mathcal{C}}$ represents the guess of those constraints we still want to become true when we further traverse the tree towards the root node.

$\hat{M}, \hat{C}, \hat{R}$, and $\hat{\gamma}$ are used to ensure that the answer set is a model of our program. $\hat{L}_<$ is a strict linear order whose existence is demanded in the definition of answer sets. $\hat{\gamma}_<$ will be used to check condition (R3) of stable models, i.e., it will contain the cardinality on the left side of the equation in (R3).

The derivation of atoms $a \in \hat{M}$ is represented by $\hat{\Delta}$. The definition of answer sets requires for each $a \in \hat{M}$ the existence of some rule $r \in \mathcal{R}$ satisfying (R1)-(R3). The set of those rules will be represented by $\hat{\delta}_R$. Sets $\hat{\delta}_h$ and $\hat{\delta}_b$ contain, respectively, the head and body constraints of rules in $\hat{\delta}_R$. The set $\hat{\delta}_M$ contains those atoms for which we have already found a head constraint to derive it. $\hat{\sigma}$ is a utility function, which ensures that each (guessed) constraint in $\hat{\delta}_h$ is indeed the head of some rule in $\hat{\delta}_R$. Thereby $\hat{\sigma}(c) = 1$ marks that such a rule was found.

Note that without any loss of generality we may assume that the root node of a normalized tree decomposition has an empty bag. Indeed, this can always be achieved by introducing at most $tw(\Pi) + 1$ additional nodes above the root of a given tree decomposition. Then the following proposition shows the correspondence between answer sets and partial solutions for the root node of a given normalized tree decomposition.

Proposition 35. *Let t_{root} be the root node of T and let $\chi(t_{root}) = \emptyset$. Then $\mathcal{AS}(\Pi) \neq \emptyset$ if and only if there exists a partial solution $\hat{\vartheta} = (t_{root}, \hat{M}, \hat{C}, \hat{R}, \hat{L}_<, \hat{\gamma}, \hat{\gamma}_<, \hat{\Delta})$ for t_{root}.*

Proof. ($\Rightarrow$) Given an answer set $M \in \mathcal{AS}(\Pi)$, we construct a partial solution $\hat{\vartheta}$ for t_{root} with derivation witness $\hat{\Delta} = (\hat{\delta}_R, \hat{\delta}_M, \hat{\delta}_h, \hat{\delta}_b, \hat{\sigma})$ as follows. Let $\hat{M} := M$, let $\hat{C} := \{c \in \mathcal{C} : M \models c\}$ and let $\hat{R} := \mathcal{R}$. Let $L_< := [a_1, \ldots, a_{|M|}] \in [\![M]\!]$ be the linear order from Proposition 33 and $f : M \to \mathcal{R}$ be the function that assigns each atom $a \in M$ the rule $r \in \mathcal{R}$ that satisfies conditions (R1)–(R3) of Proposition 33 for a. Furthermore, let $\hat{\delta}_R := \{f(a) : a \in M\}$. In order to create $\hat{L}_<$, we modify $L_<$ as follows. For every $r \in \hat{\delta}_R$ let a_r be the smallest atom in $L_<$ such that $f(a_r) = r$. Atom a_r is then replaced in $L_<$ by the sequence $c_1, \ldots, c_j, r, c_{j+1}, a_r$, where $\{c_1, \ldots, c_j\} = B(r)$ and $c_{j+1} = H(r)$. Note that by construction $\{c_1, \ldots, c_{j+1}\} \subseteq \hat{C}$. The remaining clauses from $\hat{C}$ as well as the rules $\mathcal{R} \setminus R$ are arbitrarily appended at the end of $\hat{L}_<$. For every constraint $c \in \mathcal{C}$ we set $\hat{\gamma}(c) := \Gamma(c, M, A)$. For every constraint $c \in \hat{C}$ we set $\hat{\gamma}_<(c) := \Gamma_<(c, M, A, \hat{L}_<)$. Let $\hat{\delta}_M := M$, $\hat{\delta}_h := \{H(r) : r \in \hat{\delta}_R\}$, and $\hat{\delta}_b := \bigcup_{r \in \hat{\delta}_R} B(r)$. Finally, let $\hat{\sigma}(c) := 1$ for all $c \in \hat{\delta}_h$.

We now show that $\hat{\vartheta}$ is indeed a partial solution by checking conditions 1–10 of Definition 34. Conditions 1–4, 6–7, and 9–10 are satisfied by construction. In order to see that condition 5 is satisfied, let $c_a := H(f(a))$ for each $a \in M$. Then $c_a \in \hat{\delta}_h$, $a \in Cl(c_a)$, and $c_a < a$. Therefore, $\hat{\delta}_M = \{a \in \hat{M} \mid c \in \hat{\delta}_h, a \in Cl(c), a > c\}$, which satisfies condition 5. Condition 8 is satisfied because of (R3) of Proposition 33. Hence, $\hat{\vartheta}$ is a partial solution for t_{root}.

($\Leftarrow$) For the other direction, the requirement that $\chi(t_{root}) = \emptyset$ ensures that the guessing part of a given partial solution $\hat{\vartheta}$ is nonexistent. Therefore, $\hat{C} = \{c \in \mathcal{C} : \hat{M} \models c\}$ and $\hat{R} = \{r \in \mathcal{R} \mid \hat{C} \models r\} = \mathcal{R}$. This ensures that $\hat{M} \models \Pi$ and is therefore a model of Π.

Let the linear order $L_<$ be the restriction of $\hat{L}_<$ to the set $\hat{M}$. Let $a \in \hat{M}$ be an arbitrary atom. By condition 5 of Proposition 33 there exists a constraint $c \in \hat{\delta}_h$ with $a \in Cl(c)$ and $a > c$. Therefore, by condition 9 and 10 there exists a rule $r \in \hat{\delta}_R$ with $H(r) = c$ and $c > r$. We now show that rule r is the one fulfilling (R1)–(R3) of Proposition 33 for atom a. (R1) is satisfied by construction. By condition 6, $B(r) \subseteq \hat{C}$. Therefore, $\hat{M} \models B(r)$, satisfying (R2). Finally, (R3) is satisfied through condition 8. This shows that $\hat{M}$ is indeed an answer set of Π. □

An algorithm that computes all partial solutions at each node of the tree decomposition is highly inefficient, since the size and the number of such solutions can grow exponentially in the input size. Therefore we introduce *bag assignments*, which is a data structure similar to partial solutions, but instead of ranging over the whole subtree, their scope is restricted to a single bag of the tree decomposition.

Definition 36. A *bag assignment (for node $t \in T$)* is a tuple $\vartheta = (t, M, C, R, L_<, \gamma, \gamma_<, \Delta)$ with partial model $M \subseteq t|_A$, satisfied constraints $C \subseteq t|_{\mathcal{C}}$, satisfied rules $R \subseteq t|_{\mathcal{R}}$, linear order $L_< \in [\![M \cup C \cup t|_{\mathcal{R}}]\!]$, cardinality functions $\gamma : t|_{\mathcal{C}} \to \mathbb{N}$ and $\gamma_< : C \to \mathbb{N}$, and derivation witness $\Delta = (\delta_R, \delta_M, \delta_h, \delta_b, \sigma)$ with derivation rules $\delta_R \subseteq t|_{\mathcal{R}}$, derived atoms $\delta_M \subseteq M$, derivation head constraints $\delta_h \subseteq C$, derivation body constraints $\delta_b \subseteq C$, and check function $\sigma : \delta_h \to \{0, 1\}$.

But we are not interested in arbitrary bag assignments. Instead we consider only those that can be seen as the projection of a partial solution for node t to the bag of node t. Formally, these so-called bag models are defined as follows.

Definition 37. A bag assignment ϑ for node t with $\vartheta = (t, M, C, R, L_<, \gamma, \gamma_<, \Delta)$ and $\Delta = (\delta_R, \delta_M, \delta_h, \delta_b, \sigma)$ is called a *bag model (for node t)* if there exists a partial solution $\hat{\vartheta} = (t, \hat{M}, \hat{C}, \hat{R}, \hat{L}_<, \hat{\gamma}, \hat{\gamma}_<, \hat{\Delta})$ with $\hat{\Delta} = (\hat{\delta}_R, \hat{\delta}_M, \hat{\delta}_h, \hat{\delta}_b, \hat{\sigma})$ such that

- $\hat{M} \cap \chi(t) = M$,
- $\hat{C} \cap \chi(t) = C$,
- $\hat{R} \cap \chi(t) = R$,
- $\hat{L}_<$ and $L_<$ are consistent,
- $\hat{\gamma}(c) = \gamma(c)$ for all $c \in t|_{\mathcal{C}}$,
- $\hat{\gamma}_<(c) = \gamma_<(c)$ for all $c \in t|_{\mathcal{C}}$,
- $\hat{\delta}_R \cap \chi(t) = \delta_R$,
- $\hat{\delta}_M \cap \chi(t) = \delta_M$,
- $\hat{\delta}_h \cap \chi(t) = \delta_h$,
- $\hat{\delta}_b \cap \chi(t) = \delta_b$,
- $\hat{\sigma}(c) = \sigma(c)$ for all $c \in \delta_h$.

Indeed, it turns out that it is sufficient to maintain only bag models during tree traversal.

Proposition 38. *Let t_{root} be the root node of T, and let $\chi(t_{root}) = \emptyset$. Then $\mathcal{AS}(\Pi) \neq \emptyset$ if and only if $\vartheta = (t_{root}, \emptyset, \emptyset, \emptyset, [\,], \emptyset, \emptyset, \Delta)$ with $\Delta = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$ is a bag model for t_{root}.*

Proof. Since $\chi(t_{root}) = \emptyset$, every partial solution for t_{root} is an extension of ϑ according to the conditions of Definition 37. Therefore, this statement follows from Proposition 35. □

By the same argument as for the root node, we may assume that $\chi(t) = \emptyset$ for leaf nodes t. Now a dynamic programming algorithm can be achieved by creating the only possible bag model $\vartheta = (t, \emptyset, \emptyset, \emptyset, [\,], \emptyset, \emptyset, \Delta)$ with $\Delta = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$ for each leaf t, and then propagating these bag models along the paths to the root node. Thereby the bag models are altered according to algorithmic rules, which depend only on the bag of the current node.

In order to sketch the cornerstones of the dynamic programming algorithm more clearly, we distinguish between eight types of nodes in the tree decomposition: leaf (L), branch (B), atom introduction (AI), atom removal (AR), rule introduction (RI), rule removal (RR), constraint introduction (CI), and constraint removal (CR) node. The last six types will be often augmented with the element e (either an atom, a rule, or a constraint), which is removed or added compared to the bag of the child node.

Next we define a relation $\prec_{\mathcal{T}}$ between bag assignments, which will be used to propagate bag models in a bottom-up direction along the tree decomposition $\mathcal{T}$. Afterwards we demonstrate the intuition of these algorithmic rules with the help of a small example.

Definition 39. Let $\vartheta = (t, M, C, R, L_<, \gamma, \gamma_<, \Delta)$ and $\vartheta' = (t', M', C', R', L'_<, \gamma', \gamma'_<, \Delta')$ with $\Delta = (\delta_R, \delta_M, \delta_h, \delta_b, \sigma)$ and $\Delta' = (\delta'_R, \delta'_M, \delta'_h, \delta'_b, \sigma')$ be bag assignments for nodes $t, t' \in T$. We relate $\vartheta' \prec_{\mathcal{T}} \vartheta$ if t has a single child t' and the following properties are satisfied, depending on the node type of t:

(r-RR): $r \in R'$ and

$$\begin{aligned} \vartheta &= (t, M', C', R' \setminus \{r\}, L'_< - [r], \gamma', \gamma'_<, \Delta), \text{ with} \\ \Delta &= (\delta'_R \setminus \{r\}, \delta'_M, \delta'_h, \delta'_b, \sigma'). \end{aligned}$$

(r-RI):

$$\vartheta \in \{(t, M', C', R^*, L^*_<, \gamma', \gamma'_<, \Delta) \mid L^*_< \in (L'_< + [r])\}, \text{ with}$$

$$R^* = \begin{cases} R' \cup \{r\} & \text{if } C' \models_{t|_{\mathcal{C}}} r, \\ R' & \text{otherwise,} \end{cases}$$

and one of the following two groups of properties has to be satisfied:

- "r is used": $H(r) \in t|_{\mathcal{C}} \Rightarrow (H(r) \in \delta'_h \wedge H(r) > r)$, for all $b \in B(r) \cap t|_{\mathcal{C}}$: $b \in C' \wedge b < r$, and

$$\Delta = (\delta'_R \cup \{r\}, \delta'_M, \delta'_h, \delta'_b \cup (B(r) \cap t|_{\mathcal{C}}), \sigma^*), \text{ with}$$

$$\sigma^*(c) = \begin{cases} 1 & \text{if } c = H(r), \\ \sigma'(c) & \text{otherwise.} \end{cases}$$

- "r is not used": $\Delta = \Delta'$.

(a-AR): $a \in M' \Rightarrow a \in \delta'_M$ and

$$\begin{aligned} \vartheta &= (t, M' \setminus \{a\}, C', R', L'_< - [a], \gamma', \gamma'_<, \Delta), \text{ with} \\ \Delta &= (\delta'_R, \delta'_M \setminus \{a\}, \delta'_h, \delta'_b, \sigma'). \end{aligned}$$

(a-AI): One of the following two groups of properties has to be satisfied:

- "set a to false":

$$\vartheta = (t, M', C', R', L'_<, \gamma^*, \gamma^*_<, \Delta'), \text{ with}$$

$\gamma^*(c) = \gamma'(c) + \Gamma(c, M', t|_A) - \Gamma(c, M', t'|_A)$, and
$\gamma^*_<(c) = \gamma'_<(c) + \Gamma_<(c, M', t|_A, L'_<) - \Gamma_<(c, M', t'|_A, L'_<)$.

- "set a to true":

$$\begin{aligned}\vartheta &\in \{(t, M^* = M' \cup \{a\}, C', R', L^*_<, \gamma^*, \gamma^*_<, \Delta) \mid \\ &\quad L^*_< \in (L'_< + [a])\}, \text{ with} \\ \Delta &= (\delta'_R, \delta'_M \cup \delta^*_M, \delta'_h, \delta'_b, \sigma'), \text{ where}\end{aligned}$$

$$\delta^*_M = \begin{cases}\{a\} & \text{if } \exists c \in \delta'_h, a \in Cl(c), a > c, \\ \emptyset & \text{otherwise,}\end{cases}$$

$\gamma^*(c) = \gamma'(c) + \Gamma(c, M^*, t|_A) - \Gamma(c, M', t'|_A)$, and
$\gamma^*_<(c) = \gamma'_<(c) + \Gamma_<(c, M^*, t|_A, L^*_<) - \Gamma_<(c, M', t'|_A, L'_<)$.

(c-CR): $c \in C' \Leftrightarrow l(c) \leq \gamma'(c) \leq u(c)$, $c \in \delta'_h \Rightarrow \sigma'(c) = 1$, $c \in \delta'_b \Rightarrow \gamma'_<(c) \geq l(c)$, and

$$\begin{aligned}\vartheta &= (t, M', C' \setminus \{c\}, R', L'_< - [c], \gamma', \gamma'_<, \Delta), \text{ with} \\ \Delta &= (\delta'_R, \delta'_M, \delta'_h \setminus \{c\}, \delta'_b \setminus \{c\}, \sigma').\end{aligned}$$

(c-CI): One of the following two groups of properties has to be satisfied:

- "set c to false": $c \notin B(r) \wedge c \neq H(r)$ for all $r \in \delta'_R$, and

$$\vartheta = (t, M', C', R' \cup R^*, L'_<, \gamma' \cup \gamma^*, \gamma'_<, \Delta'), \text{ with}$$

  $R^* = \{r \in t|_\mathcal{R} \mid C' \models_{t|_\mathcal{C}} r\}$, and $\gamma^* = \{(c, \Gamma(c, M', t|_A)\}$.
- "set c to true": $(c \in B(r) \Rightarrow r > c) \wedge (c = H(r) \Rightarrow r < c)$ for all $r \in \delta'_R$, and

$$\begin{aligned}\vartheta &\in \{(t, M', C^* = C' \cup \{c\}, R' \cup R^*, L^*_<, \gamma^*, \gamma^*_<, \Delta) \mid \\ &\quad L^*_< \in (L'_< + [c])\}, \text{ with} \\ \Delta &= (\delta'_R, \delta'_M \cup \delta^*_M, \delta'_h \cup \delta^*_h, \delta'_b \cup \delta^*_b, \sigma^*), \text{ where}\end{aligned}$$

  $R^* = \{r \in t|_\mathcal{R} \mid C^* \models_{t|_\mathcal{C}} r\}$, $\gamma^* = \gamma' \cup \{(c, \Gamma(c, M', t|_A)\}$,
  $\gamma^*_< = \gamma'_< \cup \{(c, \Gamma_<(c, M', t|_A, L^*_<)\}$,

$$\begin{aligned}\delta^*_b &= \begin{cases}\{c\} & \text{if } \exists r \in \delta'_R : c \in B(r), \\ \emptyset & \text{otherwise,}\end{cases} \\ \delta^*_h &\in \begin{cases}\{\{c\}\} & \text{if } \exists r \in \delta'_R : c = H(r), \\ \{\emptyset, \{c\}\} & \text{otherwise,}\end{cases}\end{aligned}$$

  $\delta^*_M = \{a \in M' \mid a \in Cl(c), c \in \delta^*_h, a > c\}$, and
  $\sigma^*(c) = 1 \Leftrightarrow c \in \delta^*_h \wedge \exists r \in \delta_R : H(r) = c$.

For branch nodes, we extend (by slight abuse of notation) $\prec_\mathcal{T}$ to a ternary relation.

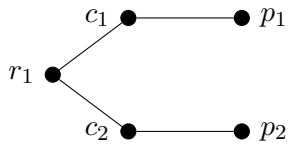

Figure 5.3: Incidence graph of Example 41.

Definition 40. Let $\vartheta = (t, M, C, R, L_<, \gamma, \gamma_<, \Delta)$, $\vartheta' = (t', M', C', R', L'_<, \gamma', \gamma'_<, \Delta')$, and $\vartheta'' = (t'', M'', C'', R'', L''_<, \gamma'', \gamma''_<, \Delta'')$ be three bag assignments for nodes $t, t', t'' \in T$ with $\Delta = (\delta_R, \delta_M, \delta_h, \delta_b, \sigma)$, $\Delta' = (\delta'_R, \delta'_M, \delta'_h, \delta'_b, \sigma')$, and $\Delta'' = (\delta''_R, \delta''_M, \delta''_h, \delta''_b, \sigma'')$. We relate $(\vartheta', \vartheta'') \prec_{\mathcal{T}} \vartheta$ if t has two children t' and t'', and the following conditions are satisfied.

- $M = M' = M''$,
- $C = C' = C''$,
- $R = R' \cup R''$,
- $L_< = L'_< = L''_<$,
- $\gamma(c) = \gamma'(c) + \gamma''(c) - \Gamma(c, M, t|_A)$ for all $c \in t|_C$,
- $\gamma_<(c) = \gamma'_<(c) + \gamma''_<(c) - \Gamma_<(c, M, t|_A, L_<)$ for all $c \in C$,
- $\delta_R = \delta'_R = \delta''_R$,
- $\delta_M = \delta'_M \cup \delta''_M$,
- $\delta_h = \delta'_h = \delta''_h$,
- $\delta_b = \delta'_b \cup \delta''_b$, and
- $\sigma(c) = \max\{\sigma'(c), \sigma''(c)\}$ for all $c \in \delta_h$.

What follows is a small example which demonstrates how this $\prec_{\mathcal{T}}$ relation is used to solve the consistency problem for PCCs. Thereby we start with the only possible bag model $\vartheta = (t, \emptyset, \emptyset, \emptyset, [\,], \emptyset, \emptyset, \Delta)$ and $\Delta = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$ for each leaf node. Now we traverse through the tree decomposition and calculate for each node all the bag assignments according to the relation $\prec_{\mathcal{T}}$. Finally, we check whether for the root node any such bag assignment could be generated.

Example 41. We are given a PCC $\Pi = (\{p_1, p_2\}, \{c_1, c_2\}, \{r_1\})$ with $c_1 = (\{(p_1, 1)\}, 1, 1)$, $c_2 = (\{(\neg p_2, 1)\}, 1, 1)$, and $r_1 = (c_1, \{c_2\})$.

Its incidence graph is depicted in Figure 5.3. A normalized tree decomposition of width 1 is shown in Figure 5.4. What follows is a list of all the bag assignments that can be computed according to the relation $\prec_{\mathcal{T}}$, starting from the trivial bag assignments of empty leaf nodes.

Node t_1: (L)

$$\vartheta_1 = (t_1, \emptyset, \emptyset, \emptyset, [\,], \emptyset, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$

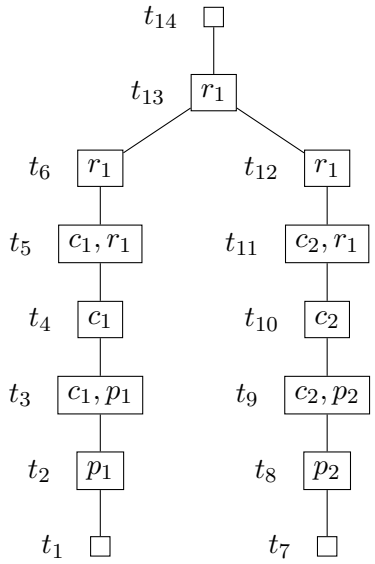

Figure 5.4: Normalized tree decomposition of Example 41.

Node t_2: (p_1-AI)

$$\vartheta_{2,1} = (t_2, \emptyset, \emptyset, \emptyset, [\,], \emptyset, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{2,2} = (t_2, \{p_1\}, \emptyset, \emptyset, [p_1], \emptyset, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$

Node t_3: (c_1-CI)

$$\vartheta_{3,1} = (t_3, \emptyset, \emptyset, \emptyset, [\,], \{(c_1, 0)\}, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{3,2} = (t_3, \emptyset, \{c_1\}, \emptyset, [c_1], \{(c_1, 0)\}, \{(c_1, 0)\}, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{3,3} = (t_3, \emptyset, \{c_1\}, \emptyset, [c_1], \{(c_1, 0)\}, \{(c_1, 0)\}, (\emptyset, \emptyset, \{c_1\}, \emptyset, \{(c_1, 0)\}))$$
$$\vartheta_{3,4} = (t_3, \{p_1\}, \emptyset, \emptyset, [p_1], \{(c_1, 1)\}, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{3,5} = (t_3, \{p_1\}, \{c_1\}, \emptyset, [c_1, p_1], \{(c_1, 1)\}, \{(c_1, 0)\}, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{3,6} = (t_3, \{p_1\}, \{c_1\}, \emptyset, [c_1, p_1], \{(c_1, 1)\}, \{(c_1, 0)\}, (\emptyset, \{p_1\}, \{c_1\}, \emptyset, \{(c_1, 0)\}))$$
$$\vartheta_{3,7} = (t_3, \{p_1\}, \{c_1\}, \emptyset, [p_1, c_1], \{(c_1, 1)\}, \{(c_1, 1)\}, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{3,8} = (t_3, \{p_1\}, \{c_1\}, \emptyset, [p_1, c_1], \{(c_1, 1)\}, \{(c_1, 1)\}, (\emptyset, \emptyset, \{c_1\}, \emptyset, \{(c_1, 0)\}))$$

Node t_4: (p_1-AR)

$$\vartheta_{4,1} = (t_4, \emptyset, \emptyset, \emptyset, [\,], \{(c_1, 0)\}, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{4,2} = (t_4, \emptyset, \{c_1\}, \emptyset, [c_1], \{(c_1, 0)\}, \{(c_1, 0)\}, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{4,3} = (t_4, \emptyset, \{c_1\}, \emptyset, [c_1], \{(c_1, 0)\}, \{(c_1, 0)\}, (\emptyset, \emptyset, \{c_1\}, \emptyset, \{(c_1, 0)\}))$$
$$\vartheta_{4,4} = (t_4, \emptyset, \{c_1\}, \emptyset, [c_1], \{(c_1, 1)\}, \{(c_1, 0)\}, (\emptyset, \emptyset, \{c_1\}, \emptyset, \{(c_1, 0)\}))$$

Node t_5: (r_1-RI)

$$\vartheta_{5,1} = (t_5, \emptyset, \emptyset, \emptyset, [r_1], \{(c_1, 0)\}, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{5,2} = (t_5, \emptyset, \{c_1\}, \{r_1\}, [r_1, c_1], \{(c_1, 0)\}, \{(c_1, 0)\}, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{5,3} = (t_5, \emptyset, \{c_1\}, \{r_1\}, [c_1, r_1], \{(c_1, 0)\}, \{(c_1, 0)\}, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{5,4} = (t_5, \emptyset, \{c_1\}, \{r_1\}, [r_1, c_1], \{(c_1, 0)\}, \{(c_1, 0)\}, (\{r_1\}, \emptyset, \{c_1\}, \emptyset, \{(c_1, 1)\}))$$
$$\vartheta_{5,5} = (t_5, \emptyset, \{c_1\}, \{r_1\}, [r_1, c_1], \{(c_1, 0)\}, \{(c_1, 0)\}, (\emptyset, \emptyset, \{c_1\}, \emptyset, \{(c_1, 0)\}))$$
$$\vartheta_{5,6} = (t_5, \emptyset, \{c_1\}, \{r_1\}, [c_1, r_1], \{(c_1, 0)\}, \{(c_1, 0)\}, (\emptyset, \emptyset, \{c_1\}, \emptyset, \{(c_1, 0)\}))$$
$$\vartheta_{5,7} = (t_5, \emptyset, \{c_1\}, \{r_1\}, [r_1, c_1], \{(c_1, 1)\}, \{(c_1, 0)\}, (\{r_1\}, \emptyset, \{c_1\}, \emptyset, \{(c_1, 1)\}))$$
$$\vartheta_{5,8} = (t_5, \emptyset, \{c_1\}, \{r_1\}, [r_1, c_1], \{(c_1, 1)\}, \{(c_1, 0)\}, (\emptyset, \emptyset, \{c_1\}, \emptyset, \{(c_1, 0)\}))$$
$$\vartheta_{5,9} = (t_5, \emptyset, \{c_1\}, \{r_1\}, [c_1, r_1], \{(c_1, 1)\}, \{(c_1, 0)\}, (\emptyset, \emptyset, \{c_1\}, \emptyset, \{(c_1, 0)\}))$$

Node t_6: (c_1-CR)

$$\vartheta_{6,1} = (t_6, \emptyset, \emptyset, \emptyset, [r_1], \emptyset, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{6,2} = (t_6, \emptyset, \emptyset, \{r_1\}, [r_1], \emptyset, \emptyset, (\{r_1\}, \emptyset, \emptyset, \emptyset, \emptyset))$$

Node t_7: (L)

$$\vartheta_7 = (t_7, \emptyset, \emptyset, \emptyset, [\,], \emptyset, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$

Node t_8: (p_2-AI)

$$\vartheta_{8,1} = (t_8, \emptyset, \emptyset, \emptyset, [\,], \emptyset, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{8,2} = (t_8, \{p_2\}, \emptyset, \emptyset, [p_2], \emptyset, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$

Node t_9: (c_2-CI)

$$\vartheta_{9,1} = (t_9, \emptyset, \emptyset, \emptyset, [\,], \{(c_2, 1)\}, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{9,2} = (t_9, \emptyset, \{c_2\}, \emptyset, [c_2], \{(c_2, 1)\}, \{(c_2, 1)\}, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{9,3} = (t_9, \emptyset, \{c_2\}, \emptyset, [c_2], \{(c_2, 1)\}, \{(c_2, 1)\}, (\emptyset, \emptyset, \{c_2\}, \emptyset, \{(c_2, 0)\}))$$
$$\vartheta_{9,4} = (t_9, \{p_2\}, \emptyset, \emptyset, [p_2], \{(c_2, 0)\}, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{9,5} = (t_9, \{p_2\}, \{c_2\}, \emptyset, [c_2, p_2], \{(c_2, 0)\}, \{(c_2, 0)\}, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{9,6} = (t_9, \{p_2\}, \{c_2\}, \emptyset, [c_2, p_2], \{(c_2, 0)\}, \{(c_2, 0)\}, (\emptyset, \emptyset, \{c_2\}, \emptyset, \{(c_2, 0)\}))$$
$$\vartheta_{9,7} = (t_9, \{p_2\}, \{c_2\}, \emptyset, [p_2, c_2], \{(c_2, 0)\}, \{(c_2, 0)\}, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{9,8} = (t_9, \{p_2\}, \{c_2\}, \emptyset, [p_2, c_2], \{(c_2, 0)\}, \{(c_2, 0)\}, (\emptyset, \emptyset, \{c_2\}, \emptyset, \{(c_2, 0)\}))$$

Node t_{10}: (p_2-AR)

$$\vartheta_{10,1} = (t_{10}, \emptyset, \emptyset, \emptyset, [\,], \{(c_2, 1)\}, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{10,2} = (t_{10}, \emptyset, \{c_2\}, \emptyset, [c_2], \{(c_2, 1)\}, \{(c_2, 1)\}, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{10,3} = (t_{10}, \emptyset, \{c_2\}, \emptyset, [c_2], \{(c_2, 1)\}, \{(c_2, 1)\}, (\emptyset, \emptyset, \{c_2\}, \emptyset, \{(c_2, 0)\}))$$

Node t_{11}: (r_1-RI)

$$\vartheta_{11,1} = (t_{11}, \emptyset, \emptyset, \{r_1\}, [r_1], \{(c_2, 1)\}, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{11,2} = (t_{11}, \emptyset, \{c_2\}, \emptyset, [r_1, c_2], \{(c_2, 1)\}, \{(c_2, 1)\}, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{11,3} = (t_{11}, \emptyset, \{c_2\}, \emptyset, [c_2, r_1], \{(c_2, 1)\}, \{(c_2, 1)\}, (\{r_1\}, \emptyset, \emptyset, \{c_2\}, \emptyset))$$
$$\vartheta_{11,4} = (t_{11}, \emptyset, \{c_2\}, \emptyset, [c_2, r_1], \{(c_2, 1)\}, \{(c_2, 1)\}, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{11,5} = (t_{11}, \emptyset, \{c_2\}, \emptyset, [r_1, c_2], \{(c_2, 1)\}, \{(c_2, 1)\}, (\emptyset, \emptyset, \{c_2\}, \emptyset, \{(c_2, 0)\}))$$
$$\vartheta_{11,6} = (t_{11}, \emptyset, \{c_2\}, \emptyset, [c_2, r_1], \{(c_2, 1)\}, \{(c_2, 1)\}, (\{r_1\}, \emptyset, \{c_2\}, \{c_2\}, \{(c_2, 0)\}))$$
$$\vartheta_{11,7} = (t_{11}, \emptyset, \{c_2\}, \emptyset, [c_2, r_1], \{(c_2, 1)\}, \{(c_2, 1)\}, (\emptyset, \emptyset, \{c_2\}, \emptyset, \{(c_2, 0)\}))$$

Node t_{12}: (c_2-CR)

$$\vartheta_{12,1} = (t_{12}, \emptyset, \emptyset, \emptyset, [r_1], \emptyset, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{12,2} = (t_{12}, \emptyset, \emptyset, \emptyset, [r_1], \emptyset, \emptyset, (\{r_1\}, \emptyset, \emptyset, \emptyset, \emptyset))$$

Node t_{13}: (B)

$$\vartheta_{13,1} = (t_{13}, \emptyset, \emptyset, \emptyset, [r_1], \emptyset, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$
$$\vartheta_{13,2} = (t_{13}, \emptyset, \emptyset, \{r_1\}, [r_1], \emptyset, \emptyset, (\{r_1\}, \emptyset, \emptyset, \emptyset, \emptyset))$$

Node t_{14}: (r_1-RR)

$$\vartheta_{14} = (t_{14}, \emptyset, \emptyset, \emptyset, [\,], \emptyset, \emptyset, (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset))$$

Since ϑ_{14} could be derived, the example is a yes-instance of the consistency problem. Indeed, it has exactly one answer set $\{p_1\}$. ⊣

Let us look exemplarily at (CR) nodes in more detail. Consider nodes t that remove a constraint c, i.e., $\chi(t) = \chi(t') \setminus \{c\}$, where t' is the child of t (see, for instance, the node with bag $\{p_3, c_3\}$ in the left branch of $\mathcal{T}_{Ex}$ in Figure 5.2, which is the c_4-removal node). Let $\vartheta' = (t', M', C', R', L'_<, \gamma', \gamma'_<, \Delta')$ with $\Delta' = (\delta'_R, \delta'_M, \delta'_h, \delta'_b, \sigma')$ be a bag model for t'. We then create a bag model for t as follows: First we have to check whether the conditions $c \in C' \Leftrightarrow l(c) \leq \gamma'(c) \leq u(c)$, $c \in \delta'_h \Rightarrow \sigma'(c) = 1$, and $c \in \delta'_b \Rightarrow \gamma'_<(c) \geq l(c)$ are satisfied. Note that those checks correspond to the conditions 1, 10, and 8 of Definition 34. They ensure that all guesses with respect to c are correct. In the case of an affirmative answer, we remove c from all sets of ϑ' in order to create the new bag model $\vartheta = (t, M', C' \setminus \{c\}, R', L'_< - [c], \gamma', \gamma'_<, \Delta)$ with $\Delta = (\delta'_R, \delta'_M, \delta'_h \setminus \{c\}, \delta'_b \setminus \{c\}, \sigma')$.

The following two theorems state that the above-defined rules indeed help in finding bag models.

Theorem 42 (Soundness)**.** *Given a bag model ϑ' (respectively bag models ϑ' and ϑ''). Then each bag assignment ϑ with $\vartheta' \prec_{\mathcal{T}} \vartheta$ (respectively $(\vartheta', \vartheta'') \prec_{\mathcal{T}} \vartheta$) is a bag model.*

Proof. Let ϑ' be a bag model for $t' \in T$ and let ϑ be a bag assignment for node $t \in T$ with $\vartheta' \prec_{\mathcal{T}} \vartheta$. Then t' is the single child of t, with t being of type (RR), (RI), (AR), (AI), (CR), or

(CI). Assume t is a (r-RR) node. According to Definition 39, we have $r \in R'$ with ϑ and ϑ' differing only in $R = R' \setminus \{r\}$, $L_< = L'_< - [r]$, and $\delta_R = \delta'_R \setminus \{r\}$. Since ϑ' is a bag model, there exists a partial solution $\hat{\vartheta}$ of t', satisfying all the conditions of Definition 37.

Claim: $\hat{\vartheta}$ is also a partial solution of t.
To verify this claim, we have to check the conditions of Definition 34. Since $t' \Downarrow_{\mathcal{C}} = t \Downarrow_{\mathcal{C}}$, $t'\downarrow_{\mathcal{C}} = t\downarrow_{\mathcal{C}}$, $t' \Downarrow_A = t \Downarrow_A$, $t'\downarrow_A = t\downarrow_A$, and $t'\downarrow_{\mathcal{R}} = t\downarrow_{\mathcal{R}}$, the only nontrivial is condition 2, where we have to check $t \Downarrow_{\mathcal{R}} \subseteq \hat{R}$. Since $r \in R'$ and $R' = \hat{R} \cap t'|_{\mathcal{R}}$, we have $r \in \hat{R}$. Hence, from $t' \Downarrow_{\mathcal{R}} \subseteq \hat{R}$ follows that $t \Downarrow_{\mathcal{R}} = t' \Downarrow_{\mathcal{R}} \cup \{r\} \subseteq \hat{R}$.

Furthermore, the projection of $\hat{\vartheta}$ to the bag $\chi(t)$ is exactly ϑ, since ϑ' and ϑ differ only by the fact that r is removed from every set in ϑ. Therefore, ϑ is a bag model. Analogously, the theorem can be checked for the other five node types above.

Now let ϑ' and ϑ'' be bag models for $t', t'' \in T$ and let ϑ be a bag assignment for node $t \in T$ with $(\vartheta', \vartheta'') \prec_{\mathcal{T}} \vartheta$. Then t has two children t' and t'' and all the properties of Definition 40 are satisfied. Since ϑ' and ϑ'' are bag models, there exist partial solutions $\hat{\vartheta}'$ of t' and $\hat{\vartheta}''$ of t''. Using these two partial solutions we construct $\hat{\vartheta} = (t, \hat{M}' \cup \hat{M}'', \hat{C}' \cup \hat{C}'', \hat{R}' \cup \hat{R}'', \hat{L}_<, \hat{\gamma}, \hat{\gamma}_<, \hat{\Delta})$ with $\hat{\Delta} = (\hat{\delta}'_R \cup \hat{\delta}''_R, \hat{\delta}'_M \cup \hat{\delta}''_M, \hat{\delta}'_h \cup \hat{\delta}''_h, \hat{\delta}'_b \cup \hat{\delta}''_b, \hat{\sigma})$. Thereby $\hat{L}_< \in (\hat{L}'_< + \hat{L}''_<)$,

$$\hat{\gamma}(c) = \begin{cases} \hat{\gamma}'(c) & c \in t' \Downarrow_{\mathcal{C}}, \\ \hat{\gamma}''(c) & c \in t'' \Downarrow_{\mathcal{C}}, \\ \hat{\gamma}'(c) + \hat{\gamma}''(c) - \Gamma(c, t|_{\hat{M}}, t|_A) & \text{otherwise}, \end{cases}$$

$$\hat{\gamma}_<(c) = \begin{cases} \hat{\gamma}'_<(c) & c \in t' \Downarrow_{\mathcal{C}}, \\ \hat{\gamma}''_<(c) & c \in t'' \Downarrow_{\mathcal{C}}, \\ \hat{\gamma}'_<(c) + \hat{\gamma}''_<(c) - \Gamma_<(c, t|_{\hat{M}}, t|_A, \hat{L}_<) & \text{otherwise}, \end{cases}$$

$$\hat{\sigma}(c) = \begin{cases} \hat{\sigma}'(c) & c \in \hat{\delta}'_h \setminus \hat{\delta}''_h, \\ \hat{\sigma}''(c) & c \in \hat{\delta}''_h \setminus \hat{\delta}'_h, \\ \max\{\hat{\sigma}'(c), \hat{\sigma}''(c)\} & \text{otherwise}. \end{cases}$$

One can now check the conditions of Definition 34 in order to verify that $\hat{\vartheta}$ is a partial solution for t. Furthermore, our construction ensures that the projection of $\hat{\vartheta}$ to the bag $\chi(t)$ is exactly ϑ, which is therefore a bag model. □

Theorem 43 (Completeness)**.** *Given a bag model ϑ for node $t \in T$. Then either t is a leaf node, or there exists a bag model ϑ' (respectively two bag models ϑ' and ϑ'') with $\vartheta' \prec_{\mathcal{T}} \vartheta$ (respectively $(\vartheta', \vartheta'') \prec_{\mathcal{T}} \vartheta$).*

Proof. Again, we have to distinguish between the node type of t. For instance, let $t \in T$ be an (r-RR) node with child t'. Let ϑ be a bag model for t. We have to show that there exists a bag model ϑ' for t' with $\vartheta' \prec_{\mathcal{T}} \vartheta$. Since ϑ is a bag model, there exists a partial solution $\hat{\vartheta}$ of t, satisfying all the conditions of Definition 37. From $r \in t \Downarrow_{\mathcal{R}}$ follows that $r \in \hat{R}$.

Now consider the projection of $\hat{\vartheta}$ onto the bag of t'. Then the result is a bag model ϑ' of t' satisfying the conditions of Definition 37 and having $r \in R'$. But then it is easy to check that

$\vartheta' \prec_{\mathcal{T}} \vartheta$, which closes the proof for (RR) nodes. Analogously the theorem can be checked for the other six node types. □

Theorem 42 and Theorem 43 show that starting from the trivial bag models for empty leafs, the dynamic programming algorithm creates all bag models for the root node. According to Proposition 38, those bag models are all we need to know. Thus, this dynamic programming algorithm solves the consistency problem.

Theorem 44. *The consistency problem for PCCs* Π *can be solved in time*

$$\mathcal{O}(2^{6w} w! k^{4w} \cdot \|\Pi\|),$$

with $w = tw(\Pi)$ *and* $k = cw(\Pi)$.

Proof. We first show that the number of different bag models at each node $t \in T$ is bounded. The number of possible sets M, C, R is bounded by 2^w, there are at most $w!$ different orderings $L_<$, the number of cardinality functions $\gamma, \gamma_<$ is bounded by k^{2w}, the number of possible sets δ_R, δ_h as well as δ_M, δ_b is bounded by 2^w each, and finally the number of check functions σ is bounded by 2^w. This leads to at most $2^{4w} w! k^{2w}$ many different bag models at node t. At each node the effort to compute a single bag model is constant

with the exception of branch nodes, where one has to compare possible pairs of bag models of each child node. Thereby only pairs are combined which have identical $M, C, R, L_<, \delta_R, \delta_h$. This means that for each bag model of the first child node there are at most $2^{2w} k^{2w}$ (the number of possible functions/sets $\gamma, \gamma_<, \delta_M, \delta_b, \sigma$) bag models at the second child to consider. The time per node is therefore bounded by $2^{6w} w! k^{4w}$ and since the number of nodes in our tree decomposition is bounded by $\mathcal{O}(\|\Pi\|)$, the total time of $\mathcal{O}(2^{6w} w! k^{4w} \cdot \|\Pi\|)$ follows. □

5.5 Extensions

In this section, we discuss some extensions of our dynamic programming approach and that of Theorem 44.

5.5.1 PWCs with Unary Weights

Our dynamic programming algorithm for the consistency problem of PCCs can be easily extended to PWCs with *unary representation* of both weights and constraint bounds (*PWCs with unary weights*, for short).

Theorem 45. *Given an arbitrary PWC* Π. *The consistency problem for PWCs with unary weights can be solved in time* $\mathcal{O}(2^{6w} w! k^{4w} \cdot \|\Pi\|)$ *with* $w = \max(3, tw(\Pi))$ *and* $k = cw(\Pi)$.

Proof. It suffices to show that every PWC Π with unary weights can be efficiently transformed into a PCC Π' such that Π is only linearly bigger than Π, the constraint-width remains the same, and the treewidth is $\max(3, tw(\Pi))$. The transformation from Π to Π' processes each literal ℓ with weight $j > 1$ in each constraint c of Π as follows: reduce the weight of ℓ to 1 and add $j - 1$ fresh atoms $\ell_2, \ldots, \ell_j$ (each of weight 1) to c. Moreover, we add, for $\alpha \in \{2, \ldots, j\}$, new

constraints $c_\alpha := (\{(\ell, 1), (\neg\ell_\alpha, 1)\}, 1, 1)$ and new rules $r_\alpha := (c_\alpha, \emptyset)$ to ensure that the fresh variables $\ell_2, \ldots, \ell_j$ have the same truth value as ℓ in every model of Π.

It is easy to check that Π' is only linearly bigger than Π (since j is given in unary representation) and that the constraint-width and treewidth are not increased (respectively changed from treewidth ≤ 2 to treewidth 3). □

5.5.2 Reasoning with PCCs and PWCs with Unary Weights

In nonmonotonic reasoning, two kinds of reasoning are usually considered, namely skeptical and credulous reasoning. Recall that an atom a is skeptically implied by a program Π if a is true (i.e., contained) in every stable model of Π. Likewise, an atom a is credulously implied by Π if a is true in some stable model of Π. Our algorithm for the consistency problem can be easily extended to an algorithm for skeptical or credulous reasoning with PCCs and PWCs with unary weights. The above upper bounds on the complexity thus carry over from the consistency problem to the reasoning problems. We only work out the PCC case below.

Theorem 46. *Both the skeptical and the credulous reasoning problem for PCCs Π can be solved in time $\mathcal{O}(2^{6w} w! k^{4w} \cdot \|\Pi\|)$ with $w = tw(\Pi)$ and $k = cw(\Pi)$.*

Proof. Suppose that we are given a PCC Π and an atom a. The dynamic programming algorithm for the consistency problem has to be extended in such a way that we additionally maintain two flags $cr(\vartheta)$ and $sk(\vartheta)$ for every bag assignment ϑ. These flags may take one of the values $\{\bot, \top\}$ with the intended meaning that $cr(\vartheta) = \top$ (respectively $sk(\vartheta) = \top$) if and only if there exists a partial solution $\hat{\vartheta} = (t, \hat{M}, \ldots)$ (respectively if and only if for all partial solutions $\hat{\vartheta} = (t, \hat{M}, \ldots)$), the atom a is true in $\hat{M}$. Otherwise this flag is set to $\bot$. Then a is credulously (respectively skeptically) implied by Π if and only if there exists a bag model (respectively if and only if for all bag models) ϑ of the root node t_{root} of T, we have $cr(\vartheta) = \top$ (respectively $sk(\vartheta) = \top$). Clearly, maintaining the two flags fits within the desired complexity bound. □

5.5.3 Bounded Treewidth and Bounded Constraint-Width

Recall that we have proved the fixed-parameter linearity of the consistency problem of PWCs when treewidth and constraint-width are taken as parameter (see Theorem 32). This fixed-parameter linearity result (as well as the analogous result for the skeptical and credulous reasoning problem which can be easily seen to be expressible in MSO logic) could also be obtained as a corollary of Theorem 45.

Indeed, consider a PWC Π whose treewidth w and constraint-width k are bounded by some fixed constant. By previous considerations, we may thus assume that all weights occurring in Π are bounded by a constant. Therefore, we can transform all weights and bounds into unary representation such that the size of the resulting PWC with unary weights differs from $\|\Pi\|$ only by a constant factor (namely 2^k). The upper bound on the complexity in Theorem 45 immediately yields the desired fixed-parameter linearity result since $f(w) \cdot \mathcal{O}(k^{2w})$ is bounded by a constant that is independent of the size of Π.

5.6 W[1]-Hardness

In this section we present the result that it is impossible under usual complexity theoretic assumptions, that one can improve the non-uniform polynomial-time result of Theorem 44 to a fixed-parameter tractability result without bounding the constraint-width as in Theorem 32.

We showed the proof of Theorem 47 in [92].

Theorem 47 ([92])**.** *The consistency problem for PCCs parameterized by treewidth is* W[1]*-hard.*

This result can be obtained by an fpt-reduction from the MINIMUM MAXIMUM OUTDEGREE problem. To state this problem we need to introduce some concepts. A (positive integral) *edge weighting* of a graph $G = (V, E)$ is a mapping w that assigns to each edge of G a positive integer. An *orientation* of G is a mapping $\Lambda : E \to V \times V$ with $\Lambda(\{u, v\}) \in \{(u, v), (v, u)\}$. The *weighted outdegree* of vertex $v \in V$ with respect to an edge weighting w and an orientation Λ is defined as

$$d^+_{G,w,\Lambda}(v) = \sum_{\{v,u\} \in E \text{ with } \Lambda(\{v,u\})=(v,u)} w(\{v, u\}).$$

An instance of MINIMUM MAXIMUM OUTDEGREE consists of a graph G, an edge weighting w of G, and a positive integer r; the question is whether there exists an orientation Λ of G such that $d^+_{G,w,\Lambda}(v) \leq r$ for each $v \in V$. The MINIMUM MAXIMUM OUTDEGREE problem with edge weights (and therefore also r) given in unary is W[1]-hard when parameterized by the treewidth of G [107].

5.7 Implementation and Evaluation

As we have seen in Chapter 4 as well as Section 5.4, tree decomposition based dynamic algorithms start at the leaf nodes and traverse the tree to the root. Thereby, at each node a set of partial solutions is generated by taking those solutions into account that have been computed for the child nodes. These partial solutions are then combined in the case of branch nodes or they are altered according to the information that is visible in the bag of the current node.

The most difficult part in constructing such an algorithm is to identify an appropriate data structure to represent the partial solutions at each node: on the one hand, this data structure must contain sufficient information so as to compute the representation of the partial solutions at each node from the corresponding representation at the child node(s). On the other hand, the size of the data structure must only depend on the size of the bag (and not on the size of the entire answer-set program). The rules according to which partial solutions have to be altered at each node are most of the time not difficult to derive once the appropriate data structure has been found.

In this section we study the problem of answer-set programming with disjunctive logic programs. Recall from Section 5.1 that this is the standard version of ASP without weight or cardinality constraints. We will now compare two two completely different realizations of the data structure for tree decomposition based dynamic algorithms for ASP. We call the resulting algorithms Dyn-ASP1 and Dyn-ASP2.

5.7.1 Dyn-ASP1

The first algorithm was proposed by Jakl, Pichler, and Woltran [57]. It was designed for propositional disjunctive programs Π which are not necessarily head-cycle free. Its data structure, called tree interpretation, follows very closely the characterization of answer sets presented in Section 5.1.

A tree interpretation for tree decomposition $\mathcal{T}$ is a tuple $(t, M, \mathcal{C})$, where t is a node of $\mathcal{T}$, $M \subseteq \chi(t)$ is called assignment, and $\mathcal{C} \subseteq 2^{\chi(t)}$ is called certificate. The idea is that M represents a partial solution limited to what is visible in the bag $\chi(t)$. That means it contains parts of a final answer set as well as all those rules which are already satisfied. The certificate $\mathcal{C}$ takes care of the minimality criteria for answer sets. It is a list of those partial solutions which are smaller than M together with the rules which are satisfied by them. This means when reaching the root node of $\mathcal{T}$, assignment M can only represent a real answer set if the associated certificate is empty or contains only entries which do not satisfy all rules.

It turns out that due to the properties of tree decompositions it is indeed enough to store only the information of the partial solution which is still visible in the current bag of the tree decomposition. Hence, for each node the number of different assignments M is bounded by a function that is single exponential in the treewidth. Together with the possible exponential size of the certificate this leads to an algorithm with a worst case running time linear in the input size and double exponential in the treewidth.

5.7.2 Dyn-ASP2

We proposed the second dynamic algorithm based on tree decompositions in [74]. In contrast to Dyn-ASP1 it is limited to head-cycle free programs. Its data structure is motivated by the following new characterization of answer sets for head-cycle free programs (HCFPs).

Theorem 48 ([74])**.** *Let $\Pi = (\mathcal{A}, \mathcal{R})$ be an HCFP. Then, $M \subseteq \mathcal{A}$ is an answer set of Π if and only if the following holds:*

- $M \in Mod(\Pi)$*, and*
- *there exists a set $\rho \subseteq \mathcal{R}$ such that*
  1. $M \subseteq \bigcup_{r \in \rho} H(r)$*;*
  2. *the derivation graph induced by M and ρ is acyclic; and*
  3. *for all $r \in \rho$: $B^+(r) \subseteq M$, $B^-(r) \cap M = \emptyset$, and $|H(r) \cap M| = 1$.*

Here the derivation graph induced by M and ρ is given by $V = M \cup \rho$ and E is the transitive closure of the edge set

$$E' = \{(b, r) : r \in \rho, b \in B^+(r) \cap M\} \cup \{(r, a) : r \in \rho, a \in H(r) \cap M\}.$$

Hence, the data structure used in Dyn-ASP2 is a tuple (G, S), where G is a derivation graph (extended by a special node due to technical reasons) and S is the set of satisfied rules used to test the first condition in Theorem 48. Again it is enough to limit G and S to the elements of the

current bag $\chi(t)$. Therefore the number of possible tuples (G, S) in each node is at most single exponential in the treewidth. This leads to an algorithm with a worst case running time linear in the input size and single exponential in the treewidth.

5.7.3 System Implementation

In order to evaluate tree decomposition based ASP solving, we implemented the algorithms mentioned above. This was done using the SHARP framework[2], a C++ interface that enables rapid development of algorithms which are based on tree or hypertree decompositions by providing (hyper-)tree decomposition routines and algorithm interfaces. It thus allows the designer to focus on the problem-specific part of the algorithm.

SHARP itself uses the htdecomp library[3], which implements several heuristics for (hyper)tree decompositions, see also Dermaku et al. [22]. The resulting ASP solver was presented in [76]. First experiments on programs of low treewidth resulted in competitive performance compared to the state-of-the-art ASP solver DLV [67]. We could outperform this solver in the task of counting the number of answer sets.

5.7.4 Evaluation of Tree Decompositions

Recall from above that the algorithm Dyn-ASP1 has a theoretical worst case running time that is double exponential in the treewidth. In contrast the worst case running time of Dyn-ASP2 is single exponential in the treewidth. Despite this theoretically better performance of Dyn-ASP2, experiments showed that Dyn-ASP2 is not always faster than Dyn-ASP1 [75]. Another observation was that the running time depends as well on the chosen heuristic for computing the tree decomposition. It turned out that not always the tree decomposition with the smallest width leads to the fastest running time. Indeed, the used heuristics resulted in tree decompositions that differed in various features or parameters.

In [75] we therefore identified a number of features of tree decompositions which influence the running time of dynamic algorithms. We did this by using machine learning techniques in order to determine which features correlate strongly with the running time of the algorithms. The features with the highest information gain were:

- Percentage of branch nodes in the normalized decomposition.
- Percentage of branch nodes before normalizing the decomposition.
- Percentage of leaf nodes before normalizing the decomposition.
- Average distance between two branch nodes.
- Relative size increase during the normalization.
- Average bag size of branch nodes.

[2] `http://www.dbai.tuwien.ac.at/proj/sharp/`
[3] `http://www.dbai.tuwien.ac.at/proj/hypertree/downloads.html`

- Relative size of the tree decomposition (number of tree nodes) compared to the size of the original graph (vertices plus edges).

Furthermore, we applied machine learning techniques to automatically select the best dynamic algorithm based on the features of a given input tree decomposition. Using different classification techniques, like k-nearest neighbor or random forest classifiers, we could correctly predict in up to 89.2% of the test instances whether Dyn-ASP1 or Dyn-ASP2 works faster on a given input.

Our second approach for selecting the faster algorithm was to use regression techniques, like k-NN (see [1]) or M5P (pruned regression tree, see [96, 113]). Thereby the main idea is to use machine learning algorithms to first predict the runtime of each dynamic algorithm in a particular instance, and then select the algorithm that has better predicted runtime. In our experiments the k-NN approach could select the faster algorithm in 88% of the test instances.

CHAPTER 6

Abduction

This chapter is based on joint work with Michael Fellows, Andreas Pfandler, and Frances Rosamond. It will appear in the proceedings of the Twenty-Sixth AAAI Conference [36].

In this chapter we study logic-based abduction, where knowledge is represented by a (set of) propositional formula(s). In the propositional abduction problem we are given a propositional theory T, a set of hypotheses H and a set of manifestations M. The task is to find a solution $S \subseteq H$ such that $S \cup T$ is consistent and logically entails M. Thus, we require that the situation represented in S is possible in the system described by T and that S explains the observations.

Abduction has recently been shown to be fixed-parameter tractable when parameterized by treewidth [47], but all other possible parameters remained unexplored. In this chapter we consider various fragments of propositional logic, namely Horn, definite Horn and Krom together with (combinations of) natural parameters.

Very related to the quest of searching for fixed-parameter tractable algorithms is the search for efficient preprocessing techniques. More precisely the goal is to obtain in polynomial time an equivalent instance (called *kernel*) whose size is bounded by a function of the parameter. While it is trivial to construct a kernel of exponential size for an arbitrary FPT problem, obtaining in polynomial time a kernel of size polynomial in the parameter remains a central algorithmic challenge that may or may not be achievable.

An overview of the results of this chapter can be found in Tables 6.1, 6.2, and 6.3.

6.1 Abduction Background

We start with a formal definition of the propositional abduction problem. First, we describe an abduction instance.

Definition 49. Let $\mathcal{C} \subseteq$ PROP be a class of propositional formulas. A *(propositional) abduction instance* for $\mathcal{C}$-theories consists of a tuple $\langle V, H, M, T \rangle$, where V is the set of variables, $H \subseteq V$ is the set of hypotheses, $M \subseteq V$ is the set of manifestations, and $T \in \mathcal{C}$ is the theory, a formula over V. It is required that $M \cap H = \emptyset$.

	ABD[PROP]	ABD[HORN]	ABD[HORN]$_{\leq/=}$
M	para-NP-h*	para-NP-h*	para-NP-h (Thm 52)
$H, \|M\| = 1$	para-NP-h (Prop 51)	npk (Thm 61)	npk (Thm 58)
k	–	–	W[P]-c (Cor 56)
$k, \|M\| = 1$	–	–	W[P]-c (Thm 55)
tw	npk (Thm 61)	npk (Thm 61)	npk (Cor 62/63)
(τ, H)	npk (Thm 61)	npk (Thm 61)	npk (Cor 62/63)
V	npk (Thm 61)	npk (Thm 61)	npk (Cor 62/63)

* cf. [103]

Table 6.1: Parameterized complexity results for PROP and HORN.

	ABD[DEFHORN]	ABD[DEFHORN]$_{\leq/=}$
M	P *	para-NP-h (Thm 52)
$H, \|M\| = 1$	P *	npk (Thm 58)
k	–	W[P]-c (Cor 56)
$k, \|M\| = 1$	–	W[P]-c (Thm 55)
tw	P *	npk (Cor 59)
(τ, H)	P *	npk (Cor 59)
V	P *	FPT (Prop 60)

* cf. [31]

Table 6.2: Parameterized complexity results for DEFHORN.

	ABD[KROM]	ABD[KROM]$_{\leq}$	ABD[KROM]$_{=}$
M	W[1]-c (Thm 73)	W[1]-c (Thm 72)	para-NP-h (Thm 54)
(H, M)	pk (Thm 76)	pk (Thm 76)	pk (Thm 76)
k	–	W[2]-c (Thm 68)	W[2]-c (Thm 68)
(k, M)	–	W[1]-c (Thm 69)	W[1]-c (Thm 70)
$k, \|M\| = 1$	–	P *	W[1]-c (Thm 70)
τ	npk (Thm 74)	npk (Thm 74)	npk (Thm 74)
V	pk (Thm 76)	pk (Thm 76)	pk (Thm 76)

* cf. [19]

Table 6.3: Parameterized complexity results for KROM.

Next we formalize the notion of a solution to an abduction instance.

Definition 50. Let $\mathcal{P} = \langle V, H, M, T \rangle$ be an abduction instance. $S \subseteq H$ is a *solution* (or explanation) to $\mathcal{P}$ if $T \cup S$ is consistent and $T \cup S \models M$. $Sol(\mathcal{P})$ denotes the set of all solutions to $\mathcal{P}$.

Note that $\models$ is the classical entailment relation from propositional logic. Let $\mathcal{C} \subseteq$ PROP be a class of propositional formulas. The *solvability problem for propositional abduction* ABD[$\mathcal{C}$] for $\mathcal{C}$-theories is the following problem:

ABD[$\mathcal{C}$]	
Instance:	An abduction instance $\mathcal{P}$.
Problem:	Decide $Sol(\mathcal{P}) \neq \emptyset$.

We introduce a version of the abduction problem where the size of the solutions is limited.

ABD[$\mathcal{C}$]$_{\leq/=}$	
Instance:	An abduction instance $\mathcal{P}$ and an integer k.
Problem:	Is there a set $S \in Sol(\mathcal{P})$ such that S has cardinality less than / equal to k.

In the sequel, we will consider parameterizations by the *vertex cover number* and the *treewidth* of abduction instances. Since those structural parameters are defined for graphs, we represent abduction instances by their *primal graphs*. Recall that for an instance $\mathcal{P} = \langle V, H, M, T \rangle$, such a graph has vertex set V and there is an edge between two vertices v and w if they occur together in a clause $c \in T$, i.e., either v or $\neg v$ as well as either w or $\neg w$ occurs in c.

We introduce now some further notation that we use in this section. For $m \in \mathbb{N}$, we use $[m]$ to denote the set $\{1, \ldots, m\}$. $\mathcal{O}^*(\cdot)$ is defined in the same way as $\mathcal{O}(\cdot)$ but ignores polynomial factors.

6.2 Classical Complexity

Early work on the computational complexity of propositional abduction was done by Selman and Levesque [103]. Among others they showed that ABD[HORN] is NP-complete. A systematic complexity analysis was done by Eiter and Gottlob [31]. Their results include that ABD[PROP] is Σ_2^{P}-complete and that ABD[DEFHORN] is in P. Note that the hardness results for PROP and HORN hold even for $|M| = 1$ since in those classes one can add a new clause to the theory where all existing manifestations imply a single new manifestation.

The problem ABD[KROM] was shown to be NP-complete by Nordh and Zanuttini [86] while Creignou and Zanuttini [19] showed that it is in P when restricted to $|M| = 1$. In the latter result, they use the fact that ABD[KROM] restricted to a single manifestation has a solution if and only if it has a solution of size ≤ 1. Therefore, ABD[KROM]$_{\leq}$ with $|M| = 1$ is in P by the same argument.

The following proposition is a consequence from a remark in [31], which states that deciding $S \in Sol(\mathcal{P})$ for an instance $\mathcal{P}$ is DP-complete. This also shows that for PROP parameters H and M are not sufficient.

Proposition 51. ABD[PROP] *and* ABD[PROP]$_{\leq/=}$ *are* DP*-complete for* $|H| = 0$*. Those problems remain* DP*-hard even if* $|M| = 1$*.*

In order to motivate a parameterized complexity analysis, the remainder of this section is dedicated to showing that the problems ABD[HORN]$_{\leq/=}$, ABD[DEFHORN]$_{\leq/=}$, as well as ABD[KROM]$_{\leq/=}$ are intractable in the classical setting. According to Definition 50, our reductions must ensure both consistency and entailment.

Theorem 52. ABD[HORN]$_{\leq/=}$ *and* ABD[DEFHORN]$_{\leq/=}$ *are* NP*-complete, even if* $|M| = 1$.

Proof. Membership is trivial. We show hardness by reduction from VERTEX COVER. Recall that this problem is defined as follows. Given a graph $G = (N, E)$ and integer k. Does G have a vertex cover of size $\leq k$ respectively of size $= k$? We construct an instance $(\langle V, H, M, T\rangle, k)$ of ABD[DEFHORN]$_{\leq/=}$ as follows. We introduce a new variable for each vertex in N, for each edge in E and for the manifestation m. By slight abuse of notation this means $V := N \cup E \cup \{m\}$, where m is a new variable, $H := N$, $M := \{m\}$, and

$$T := \big(m \vee \bigvee_{e \in E} \neg e\big) \wedge \bigwedge_{\substack{e \in E \\ e = \{x,y\}}} \big((x \to e) \wedge (y \to e)\big). \tag{6.1}$$

Note that $T \cup S$ is satisfiable for every $S \subseteq H$. The first clause of T ensures that m is entailed if and only if each $e \in E$ is entailed. This in turn is the case if and only if S contains an endpoint of each edge and therefore is a vertex cover of (N, E). □

A slight variation of the above proof allows to show that the problem asking for solutions less or equal to a certain size is also hard for KROM.

Corollary 53. ABD[KROM]$_{\leq}$ *is* NP*-complete.*

Proof. This can be shown similarly to the proof of Theorem 52. Thereby the first clause of Equation 6.1 is removed, all edges are used as manifestations $M := E$, and we set $V := N \cup E$. □

If we ask instead for solutions having exactly certain size then abduction for KROM formulas is hard even if there is only a single manifestation.

Theorem 54. ABD[KROM]$_{=}$ *is* NP*-complete even if* $|M| = 1$.

Proof sketch. Membership is trivial. We show hardness by reduction from INDEPENDENT SET. Let an instance of INDEPENDENT SET be given by a graph $G = (N, E)$ and integer k. We construct an instance $\langle V, H, M, T\rangle$ of ABD[KROM]$_{=}$ as follows. Let $V := N \cup \{m\}$, where m is a new symbol, $H := N$, $M := \{m\}$, and

$$T := m \wedge \bigwedge_{\{x,y\} \in E} (\neg x \vee \neg y).$$

Note that m is trivially entailed. The construction ensures that T is satisfiable by a solution $S \subseteq H$ if and only if S is an independent set of (N, E) of size k. □

6.3 Parameterized Complexity

In this section we study the parameterized complexity of abduction. The first part is mainly dedicated to the HORN and DEFHORN fragments, whereas the second part deals with the KROM fragment. Unless otherwise specified, V, H, M, and T refer to the components of an abduction instance (see Definition 50). Additionally, k denotes the bound on the solution size. When parameterizing by the cardinality of some set, we omit the vertical bars, for example "parameterized by M" means parameterized by $|M|$. Two parameters together are denoted by a tuple, e.g. (k, M) instead of $k + |M|$.

6.3.1 Horn and Definite Horn Theories

We start by showing that the HORN fragments are intractable when parameterized by the solution size k.

Theorem 55. ABD[HORN]$_{\leq/=}$ *and* ABD[DEFHORN]$_{\leq/=}$ *parameterized by* (k, M) *are* W[P]*-complete even if* $|M| = 1$.

Proof. For the W[P]-membership, note that the problem can be solved by nondeterministically guessing k times a (not necessarily distinct in case of $\leq$) hypothesis, each of which can be described by $\log n$ bits, and then deterministically checking consistency and entailment. The checking part can be done in polynomial time for HORN as well as DEFHORN theories.

We show hardness by reduction from WEIGHTED MONOTONE CIRCUIT SAT, where an instance is given by a monotone circuit C and an integer k. Recall that this problem is W[P]-complete, when parameterized by k, even when every AND-gate and every OR-gate is binary. We construct an instance $(\langle V, H, M, T\rangle, k)$ for the abduction problem. First, we introduce a new variable for each gate of C and call the resulting set V. By slight abuse of notation, in the following we will identify the variables in V with the gates they represent. Let H be the set of input gates and let $M := \{m\}$, where m represents the output gate.

Theory T is constructed as follows. For each AND-gate a with input i_1 and i_2, we add $(i_1 \wedge i_2 \rightarrow a)$ to T. For each OR-gate o with input i_1 and i_2, we add $(i_1 \rightarrow o) \wedge (i_2 \rightarrow o)$ to T. By construction, for each set $S \subseteq H$, $T \cup S$ is consistent. Furthermore, $T \cup S \models M$ if and only if activating only the input gates in S satisfies C. □

Note that removing a parameter can not lower the parameterized complexity of a problem. Therefore the problems remain W[P]-hard when parameterized by k alone. Since the membership result above only uses parameter k, we immediately get the following corollary.

Corollary 56. ABD[HORN]$_{\leq/=}$ *as well as* ABD[DEFHORN]$_{\leq/=}$ *parameterized by* k *are* W[P]*-complete.*

On the other hand, the parameterization by the number of hypotheses is trivially FPT.

Proposition 57. ABD[HORN] *and* ABD[HORN]$_{\leq/=}$ *parameterized by* H *are* FPT*, solvable in time* $\mathcal{O}^*(2^{|H|})$.

Proof. Let $\langle V, H, M, T\rangle$ be an abduction instance with theory $T \in$ HORN. There exist $2^{|H|}$ many subsets of H. Given $S \subseteq H$, checking if $S \in Sol(\mathcal{P})$ can be done in polynomial time for HORN-theories [31]. Therefore, the time bound follows. □

Recall from Section 2.1 that every fixed-parameter tractable problem admits a kernelization algorithm. But the resulting kernel might be exponential in the parameter values. Despite ABD[HORN] and ABD[HORN]$_{\leq/=}$ being trivially FPT, it turns out that they do not admit a polynomial kernel, even when adding the solution size as a parameter. This follows from the more general result below.

Theorem 58. ABD[DEFHORN]$_{\leq}$ *and* ABD[DEFHORN]$_{=}$ *parameterized by* H *do not admit a polynomial kernel unless the Polynomial Hierarchy collapses, even if* $|M| = 1$.

Proof. We show the result for ABD[DEFHORN]$_{\leq}$ by a PPT reduction from SMALL UNIVERSE HITTING SET, where an instance is given by a family of sets $\mathcal{F} = \{F_1, \ldots, F_l\}$ over an universe $U = \bigcup_{i=1}^{l} F_i$ with $|U| = d$ and an integer k. The question is to find a set $U' \subseteq U$ of cardinality $\leq k$ such that each set in the family has a non-empty intersection with U'. This problem, parameterized by k and d does not admit a polynomial kernel unless the Polynomial Hierarchy collapses [23].

We construct an ABD[DEFHORN]$_{\leq}$ instance $(\langle V, H, M, T\rangle, k)$ as follows. Let X be a set of new variables $\{x_1, \ldots, x_l\}$ representing elements of $\mathcal{F}$. Let $H := U$, $M := \{m\}$, where m is a new variable, $V := H \cup X \cup M$, and

$$T := (x_1 \wedge \cdots \wedge x_l \to m) \wedge \bigwedge_{i \in [l]} \bigwedge_{e \in F_i} (e \to x_i).$$

Manifestation m is entailed if and only if all variables in X are entailed. Variable $x_i \in X$ is entailed if and only if at least one of the elements in the set F_i is selected. Therefore, a solution $S \subseteq H$ corresponds to a hitting set of the same size.

We can reduce ABD[DEFHORN]$_{\leq}$ to ABD[DEFHORN]$_{=}$ because of the monotonicity of DEFHORN formulas. To be more precise, if there is a solution $S \subseteq H$ of an ABD[DEFHORN]$_{\leq}$ instance, then also all $S' \supset S$ are solutions as well. □

A similar reduction can be used to show that even the aggregate parameterization with both τ and H does not yield a polynomial kernel. Since $tw(G) \leq \tau(G)$ for every graph G, we immediately obtain the same result for parameter tw.

Corollary 59. ABD[DEFHORN]$_{\leq}$ *and* ABD[DEFHORN]$_{=}$ *parameterized by* (τ, H) *do not admit a polynomial kernel unless the Polynomial Hierarchy collapses.*

Proof. This can be shown by using the same reduction as in the proof of Theorem 58, but without the restriction to a single manifestation. That means, the conjunct $(x_1 \wedge \cdots \wedge x_l \to m)$ is removed from T and $M := X$. Observe that $U = H$ is a vertex cover of the abduction instance. □

For parameter V, even ABD[PROP] is trivially FPT.

Proposition 60. ABD[PROP] *parameterized by V is* FPT*, solvable in time $\mathcal{O}^*(2^{2|V|})$.*

Proof. There are at most $2^{|H|} \leq 2^{|V|}$ possible solution candidates. For each of them we need to test consistency and entailment, which can be done in time $\mathcal{O}(2^{|H|}(n + 2^{|V|}n))$. □

Again we show that this parameterization is not sufficient for a polynomial kernel.

Theorem 61. ABD[HORN] *parameterized by V does not admit a polynomial kernel unless the Polynomial Hierarchy collapses.*

Proof. We show that the problem parameterized by (V, H) is compositional. Parameter H does not change the problem since $H \subseteq V$, but allows us to assume in the composition that all instances have the same number of hypotheses. Let $\mathcal{P}_1, \ldots, \mathcal{P}_t$ be a given sequence of instances of ABD[HORN] where $\mathcal{P}_i = \langle V_i, H_i, M_i, T_i \rangle$, $1 \leq i \leq t$, with $|V_i| = d$ and $|H_i| = e$. We assume without loss of generality that $V_i = V_j$ and $H_i = H_j$ for all $1 \leq i < j \leq t$ since otherwise we could rename the variables. We distinguish two cases.

Case 1: $t > 2^{2d}$. Let $n := \max_{i=1}^{t} \|\mathcal{P}_i\|$. Whether $\mathcal{P}_i$ has a solution can be decided in time $\mathcal{O}(2^{2d}n)$ by the FPT algorithm from Proposition 60. We can check whether at least one of $\mathcal{P}_1, \ldots, \mathcal{P}_t$ has a solution in time $\mathcal{O}(t2^{2d}n) \leq \mathcal{O}(t^2 n)$ which is polynomial in $\sum_{i=1}^{t} \|\mathcal{P}_i\|$. If some $\mathcal{P}_i$ has a solution, we output $\mathcal{P}_i$; otherwise we output $\mathcal{P}_1$, which has no solution. Hence, we have a composition algorithm in Case 1.

Case 2: $t \leq 2^{2d}$. We construct a new instance $\mathcal{P} := \langle V, H, M, T \rangle$ of ABD[HORN] as follows. Let $s := \lceil \log_2 t \rceil$. Let $V := V_1 \cup X \cup X' \cup Y \cup \{m\}$, where $X := \{x_1, \ldots, x_s\}$, $X' := \{x'_1, \ldots, x'_s\}$, $Y := \{y_1, \ldots, y_s\}$, and m. Thereby $X \cup X' \cup Y \cup \{m\}$ are $3s + 1$ new variables. Let $H := H_1 \cup X \cup X'$ and let $M := Y \cup \{m\}$. For each theory T_i we create a new theory

$$T'_i := T_i \cup \{\{\neg m' \mid m' \in M_i\} \cup \{m\}\}.$$

Let $C_1, \ldots, C_{2^s}$ be a sequence of all 2^s possible clauses $\{l_1, \ldots, l_s\}$ where l_j is either $\neg x_j$ or $\neg x'_j$, $1 \leq j \leq s$. For each theory T'_i we create a new theory $T''_i := \{C \cup C_i \mid C \in T_i\}$. Finally, let

$$T := \bigcup_{i=1}^{t} T''_i \cup \bigcup_{j=1}^{s} \left\{\{\neg x_j, y_j\}, \{\neg x'_j, y_j\}, \{\neg x_j, \neg x'_j\}\right\}.$$

The idea is the following. Since the y_j's are manifestations, the clauses $\{\neg x_j, y_j\}$, $\{\neg x'_j, y_j\}$, and $\{\neg x_j, \neg x'_j\}$, $1 \leq j \leq s$, ensure the equivalence $\neg x_j \equiv x'_j$ which is not directly expressible in HORN. Because of this equivalence, a solution S of $\mathcal{P}$ has to contain exactly one of x_j or x'_j for each $1 \leq j \leq s$. Therefore, there is exactly one subclause (in the construction they were merged with other clauses) C_l, $1 \leq l \leq 2^s$, which is not satisfied by S. Hence, all theories T_i with $i \neq l$ are trivially satisfied and cannot entail the manifestation m. Thus, $\mathcal{P}$ has a solution if and only if $\mathcal{P}_l$ has one. Since $|V|$ and $|H|$ is polynomial in d respectively e, we have also a composition algorithm in Case 2, and thus ABD[HORN] parameterized by V and H is compositional.

It follows now from Theorem 5 that this problem does not admit a polynomial kernel unless the Polynomial Hierarchy collapses. □

Since ABD[HORN] has a solution if and only if there is a solution for ABD[HORN]$_{\leq}$ of size $\leq |H|$, we immediately get the following corollary.

Corollary 62. ABD[HORN]$_{\leq}$ *parameterized by V does not admit a polynomial kernel unless the Polynomial Hierarchy collapses.*

To show the same result for ABD[HORN]$_{=}$, slightly more effort is needed.

Corollary 63. ABD[HORN]$_{=}$ *parameterized by V does not admit a polynomial kernel unless the Polynomial Hierarchy collapses.*

Proof. We present a PPT reduction from ABD[HORN] parameterized by V. Let an instance of ABD[HORN] be given by $\mathcal{P} = \langle V, H, M, T\rangle$ with $H = \{h_1, \ldots, h_e\}$. We construct a new instance $\mathcal{P}' := (\langle V \cup H' \cup M', H \cup H', M \cup M', T'\rangle, k)$ for ABD[HORN]$_{=}$ as follows. Let $H' := \{h'_1, \ldots, h'_e\}$ and $M' := \{m_1, \ldots, m_e\}$ be new variables. Let $k := e$ and let

$$T' := T \cup \bigcup_{i=1}^{e} \{\{\neg h_i, \neg h'_i\}, \{\neg h_i, m_i\}, \{\neg h'_i, m_i\}\}.$$

Then $\mathcal{P}$ has a solution if and only if $\mathcal{P}'$ has a solution of size k. The reason for this is that the clauses $\{\neg h_i, \neg h'_i\}, \{\neg h_i, m_i\}, \{\neg h'_i, m_i\}$ enforce that a solution S contains exactly one of the two hypotheses h_i, h'_i for each $1 \leq i \leq e$. Selecting h'_i in $\mathcal{P}'$ is equivalent to not selecting h_i in $\mathcal{P}$, since the variables h'_i occur nowhere in T. □

6.3.2 Krom Theories

Next we study the KROM fragment. Thereby the following preprocessing function will be very useful. Recall the definition of the resolution operator $Res(\cdot)$ from Section 2.3.

Definition 64. Given an abduction instance for KROM theories $\langle V, H, M, T\rangle$. We define the function $TrimRes(T, H, M) := \{C \in Res(T) \mid C \subseteq X\}$, with $X = H \cup M \cup \{\neg x \mid x \in (H \cup M)\}$.

In other words, the function $TrimRes(T, H, M)$ first computes the closure under resolution $Res(T)$ and then keeps only those clauses which solely consist of hypotheses and manifestations.

We will show that while the complexity is lower than in the HORN fragment when parameterized by k, the problem remains intractable. In order to simplify the proof of this result we first show some lemmas which we need later on.

Lemma 65. *Let T be a satisfiable* KROM *theory and let S be a set of propositional variables. Then $T \wedge S$ is unsatisfiable if and only if there exist $x, y \in S$ such that $T \models \neg x \vee \neg y$.*

Lemma 66. *Let $\langle V, H, M, T\rangle$ be an abduction instance for* KROM *theories and let $S \subseteq H$. Then $T \wedge S$ is satisfiable if and only if $TrimRes(T, H, M) \wedge S$ is satisfiable.*

Proof. Applying resolution does not change the set of models of theory T, i.e., $Mod(T) = Mod(Res(T))$. Removing clauses can only increase the set of models. Hence, $Mod(T) \subseteq Mod(TrimRes(T))$. Therefore, $T \wedge S$ being satisfiable implies that $TrimRes(T) \wedge S$ is satisfiable. For the other direction assume that $T \wedge S$ is unsatisfiable. By Lemma 65 we know that there exist $h_1, h_2 \in S$ with $T \models \neg h_1 \vee \neg h_2$. Hence, $\{\neg h_1, \neg h_2\} \in Res(T)$ and further $\{\neg h_1, \neg h_2\} \in TrimRes(T)$. But then, by Lemma 65, $TrimRes(T) \wedge S$ is unsatisfiable. □

Lemma 67. *Let $\langle V, H, M, T\rangle$ be an abduction instance for* KROM *theories, $S \subseteq H$, $m \in M$, and $T \wedge S$ be satisfiable. Then $T \wedge S \models m$ implies that either $\{m\} \in TrimRes(T, H, M)$ or there exists some $h \in S$ with $\{\neg h, m\} \in TrimRes(T, H, M)$.*

Proof. From $T \wedge S \models m$ follows that $\{m\} \in Res(T \wedge S)$. In case $\{m\} \in Res(T)$ we are done since then $\{m\} \in TrimRes(T)$. For the other case, i.e. $\{m\} \notin Res(T)$, note that $Res(T \wedge S) = Res(Res(T) \wedge S)$. Assuming that $Res(T)$ has already been computed, calculating $Res(Res(T) \wedge S)$ results only in resolution steps involving a singleton and a binary clause. Hence there must exist a chain of resolution steps of the form $\langle\{l_1\}, \{\neg l_1, l_2\} \Rightarrow \{l_2\}\rangle, \langle\{l_2\}, \{\neg l_2, l_3\} \Rightarrow \{l_3\}\rangle, \ldots, \langle\{l_{n-1}\}, \{\neg l_{n-1}, l_n\} \Rightarrow \{l_n\}\rangle$ with $l_1 \in S$, $l_n = m$, and all $\{l_i, l_j\} \in Res(T)$. But then the clause $\{\neg l_1, m\}$ can be obtained by resolution as well, i.e., $\{\neg l_1, m\} \in Res(T)$ and in consequence $\{\neg l_1, m\} \in TrimRes(T)$. □

These results now allow us to finally show that finding small solutions for KROM abduction is still hard when parameterized by the solution size.

Theorem 68. ABD[KROM]$_{\leq}$ *and* ABD[KROM]$_{=}$ *parameterized by k are* W[2]*-complete.*

Proof. We show membership by reducing an abduction instance $(\langle V, H, M, T\rangle, k)$ to an instance $(\mathcal{A}, \varphi)$ of MC[$\Sigma_{2,1}$]. First we check whether the empty set is already a solution. In that case we return a tautology. In the other case we first ensure that T is satisfiable and compute $TrimRes(T, H, M)$ as defined in Definition 64. We construct structure

$$\mathcal{A} := \langle A, \mathsf{hyp}, \mathsf{mani}, \mathsf{fact}, \mathsf{cl}, \mathsf{pos}, \mathsf{neg}\rangle$$

as follows. Domain A contains an element for each hypothesis in H, each manifestation in M and two distinct elements denoted by *positive* and *negative*. Let the sets hyp (respectively mani) represent the hypotheses (respectively manifestations). We use the following notation. Let l be a literal, then $pol(l)$ denotes the element *positive* (respectively *negative*) if l is a positive (respectively negative) literal. Relation fact contains the pairs $\{(pol(l), l) \mid \{l\} \in TrimRes(T, H, M)\}$. Relation cl contains the tuples $\{(pol(l_1), l_1, pol(l_2), l_2) \mid \{l_1, l_2\} \in TrimRes(T, H, M), l_1 \neq l_2\}$. Finally, we have that pos := {*positive*} and that neg := {*negative*}. We define

$$\psi := \mathsf{pos}(p) \wedge \mathsf{neg}(n) \wedge \bigwedge_{i\in[k]} \mathsf{hyp}(h_i) \wedge \bigwedge_{i\in[k]} \neg\mathsf{fact}(n,h_i) \wedge \bigwedge_{i,j\in[k]} \neg\mathsf{cl}(n,h_i,n,h_j),$$

$$\chi[x] := \mathsf{fact}(p,x) \vee \bigvee_{j\in[k]} \mathsf{cl}(n,h_j,p,x),$$

$$\varphi := \exists h_1 \cdots \exists h_k\, \exists p\, \exists n\, \forall m\, \psi \wedge (\mathsf{mani}(m) \to \chi[m]).$$

Since T is satisfiable, we know by Lemma 65 that $T \wedge S$ is unsatisfiable if and only if there exist (not necessarily distinct) $x, y \in S$ such that $T \wedge x \wedge y$ is unsatisfiable. Remember that we used $\mathit{TrimRes}(T,H,M)$ to construct φ at the beginning of the reduction. It follows from Lemma 66 that $T \wedge S$ is satisfiable if and only if for all $h_1, h_2 \in S$, $\{\neg h_1\} \notin \mathit{TrimRes}(T,H,M)$ and $\{\neg h_1, \neg h_2\} \notin \mathit{TrimRes}(T,H,M)$. This is encoded in φ by requiring $\neg\mathsf{fact}(n,h_i)$ and $\neg\mathsf{cl}(n,h_i,n,h_j)$ for all $i,j \in [k]$. Having ensured consistency it remains to check entailment. From Lemma 67 we know that in this setting it is sufficient to check whether each manifestation m is either contained as a fact in $\mathit{TrimRes}(T,H,M)$ or there is a single hypothesis $h \in S$ such that $\{\neg h, m\} \in \mathit{TrimRes}(T,H,M)$. In φ this is ensured by subformula χ. Therefore, ABD[KROM]$_\leq$ is in W[2].

The same reduction can be used to show membership for ABD[KROM]$_=$ if we additionally add the conjuncts $\bigwedge_{1\leq i<j\leq k}(h_i \neq h_j)$ to the formula φ.

We show hardness by reduction from RED-BLUE DOMINATING SET. Let an instance of RED-BLUE DOMINATING SET be given by a bipartite graph $G = (N_{\text{red}} \cup N_{\text{blue}}, E)$ and an integer k. Recall that this problem is W[2]-complete when parameterized by k. We construct an instance $(\langle V,H,M,T\rangle, k)$ of ABD[KROM]$_\leq$ as follows. Let $V := N_{\text{red}} \cup N_{\text{blue}}$, $H := N_{\text{red}}$, $M := N_{\text{blue}}$, and

$$T := \bigwedge_{n\in N_{\text{red}}, b\in N[n]} (n \to b),$$

where $N[n]$ contains n and all its adjacent vertices. A set $S \subseteq H$ of size $\leq k$ is a solution for ABD[KROM]$_\leq$ if and only if it is a solution for the dominating set problem. Due to the monotonicity of dominating sets, i.e., each superset of a dominating set is a dominating set as well, there is a solution of size k if and only if there is a solution of size $\leq k$ (assuming that $|N_{\text{red}}|$ is big enough). Thus, hardness also holds for ABD[KROM]$_=$. □

The next two theorems show that in KROM, adding M as a parameter reduces the complexity by one level in the W-hierarchy.

Theorem 69. ABD[KROM]$_\leq$ *parameterized by* (k,M) *is* W[1]*-complete.*

Proof. We show W[1]-membership by reducing an instance $(\langle V,H,M,T\rangle, k)$ to an MC[Σ_1] instance $(\mathcal{A},\varphi)$. The reduction is similar to the one used in the proof of Theorem 68. Let

$\mathcal{A} := \langle A, \mathsf{hyp}, \mathsf{mani}, \mathsf{fact}, \mathsf{cl}, \mathsf{pos}, \mathsf{neg}\rangle$, ψ, and χ be defined as before, let $|M| = d$, and

$$\varphi := \exists h_1 \cdots \exists h_k \; \exists p \; \exists n \; \exists m_1 \cdots \exists m_d \; \psi \wedge \bigwedge_{1 \leq i < j \leq d} (m_i \neq m_j) \wedge \bigwedge_{i \in [d]} (\mathsf{mani}(m_i) \wedge \chi[m_i]).$$

The correctness can be shown similarly as in Theorem 68.

Next, we show hardness by reduction from MULTICOLORED INDEPENDENT SET, where an instance is given by a graph $G = (N, E)$, a size bound k and a k-coloring of the vertices $\mathsf{c} : N \to \{c_1, \ldots, c_k\}$. The task is to find a subset $N' \subseteq N$, $|N'| = k$, such that for all $x, y \subseteq N'$: $\{x, y\} \notin E$ and $\mathsf{c}(x) \neq \mathsf{c}(y)$. This problem is W[1]-complete when parameterized by k [35]. We construct an abduction instance $(\langle V, H, M, T\rangle, k)$ as follows. Let $V := N \cup \{c_1, \ldots, c_k\}$, $H := N$, $M := \{c_1, \ldots, c_k\}$, and

$$T := \bigwedge_{x,y \in E} (\neg x \vee \neg y) \wedge \bigwedge_{n \in N} (n \to c(n)).$$

The k different manifestations (colors) imply that a solution contains at least k hypotheses. The independent set property is ensured by the consistency check. □

For ABD[KROM]$_=$ this holds even if there exists only a single manifestation.

Theorem 70. ABD[KROM]$_=$ *is* W[1]*-complete, when parameterized by* (k, M)*, even when* $|M| = 1$.

Proof. Membership can be shown analogously to the proof of Theorem 69 by adding

$$\bigwedge_{1 \leq i < j \leq k} (h_i \neq h_j)$$

to formula φ.

Hardness is shown by reduction from INDEPENDENT SET. We reduce a graph $G = (N, E)$ and integer k to an abduction instance $(\langle V, H, M, T\rangle, k)$. Let $V := N \cup \{m\}$, $H := N$, $M := \{m\}$, and

$$T := m \wedge \bigwedge_{\{x,y\} \in E} (\neg x \vee \neg y).$$

By construction, entailment is always fulfilled. □

Recall that by Theorem 54, ABD[KROM]$_=$ parameterized by M is para-NP-hard. We show now that interestingly ABD[KROM]$_\leq$ parameterized by M is W[1]-complete. Towards this goal we first prove the following lemma.

Lemma 71. *If an instance of* ABD[KROM] *has a solution, then it has a solution* S *such that* $|S| \leq |M|$.

Proof. It is known (see [19]) that an ABD[KROM] instance with $|M| = 1$ has a solution if and only if there is a solution of size ≤ 1. Thus, a single manifestation is either entailed by the theory itself or by putting a single hypothesis into the solution. Now consider an ABD[KROM] instance $\langle V, H, M, T\rangle$ with $|M| > 1$. Assume towards a contradiction that there is a solution, but no solution of size $\leq |M|$. Then, we can construct $|M|$ instances $\mathcal{P}_i = \langle V, H, M_i, T\rangle$, where M_i is the set containing only the i-th manifestation. At least one of these instances must have a solution of size > 1. This however contradicts the previously mentioned fact for ABD[KROM] instances with $|M| = 1$. □

This now allows us to show that $\text{ABD[KROM]}_{\leq}$ is significantly easier than $\text{ABD[KROM]}_{=}$ when parameterized by M.

Theorem 72. $\text{ABD[KROM]}_{\leq}$ *is* W[1]*-complete, when parameterized by* M.

Proof. Hardness follows immediately from Theorem 69.

For the membership consider the reduction to $\text{MC}[\Sigma_1]$ from the proof of Theorem 69. By Lemma 71 we know that ABD[KROM] has a solution if and only if there is a solution of size $\leq |M|$. Therefore, it is sufficient to consider only solutions of size $b := \min(k, |M|)$. Thus, we can replace in formula φ from the proof of Theorem 69 any occurrence of k by b. The length of this formula can be bounded in terms of M. □

The following theorem generalizes the classical results of KROM-abduction. It is NP-complete in general and in P when $|M| = 1$. In fact, the P-membership for every fixed number of manifestations follows from the W[1]-completeness.

Theorem 73. ABD[KROM] *is* W[1]*-complete, when parameterized by* M.

Proof. Membership follows from Theorem 72.

We show hardness by reduction from INDEPENDENT SET. Let (G, k) with graph $G = (N, E)$ and vertices $N = \{v_1, \ldots, v_l\}$ be an instance of INDEPENDENT SET. The construction is inspired by [64]. We construct an instance $(\langle V, H, M, T\rangle, k)$ for the abduction problem. Let $H := \{h_i^j \mid i \in [l], j \in [k]\}$, $M := \{m_i \mid i \in [k]\}$, and $V := N \cup H \cup M$. Next we create

$$T := T_{\text{IS}} \wedge \bigwedge_{i \in [4]} T_i,$$

where

$$
\begin{aligned}
T_{\text{IS}} &:= \bigwedge_{\{x,y\}\in E} (\neg x \vee \neg y),\\
T_1 &:= \bigwedge_{i\in[l]} \bigwedge_{j\in[k]} (h_i^j \to m_j),\\
T_2 &:= \bigwedge_{i\in[l]} \bigwedge_{\substack{j,j'\in[k],\\ j\neq j'}} (h_i^j \to \neg h_i^{j'}),\\
T_3 &:= \bigwedge_{\substack{i,i'\in[l],\\ i\neq i'}} \bigwedge_{j\in[k]} (h_i^j \to \neg h_{i'}^j),\\
T_4 &:= \bigwedge_{i\in[l]} \bigwedge_{j\in[k]} \big(h_i^j \to v_i\big).
\end{aligned}
$$

The intended meaning of the subformulas is the following. Formula T_{IS} encodes the independent set property. What is needed in addition, is a mechanism that ensures that k many vertices are picked. To this end, we introduce k hypotheses $h_i^1 \dots h_i^k$ for each vertex v_i. Selecting hypothesis h_i^j corresponds to selecting vertex i as the j-th pick in the independent set. Subformula T_1 ensures that the k manifestations are only entailed if for each $j \in [k]$ (the j-th pick) at least one hypothesis of $h_1^j, \dots, h_l^j$ is selected. Subformula T_2 ensures that for each vertex at most one hypothesis is picked. In contrast, subformula T_3 ensures that we select at most one hypothesis as the same pick. Finally, subformula T_4 forces the variable corresponding to a vertex to true if one of the corresponding hypotheses was chosen.

To sum up, any solution will contain exactly k hypotheses each of which forces one of the vertex variables to true. Since the corresponding vertices form an independent set, a solution of the abduction instance corresponds to an independent set of size k. □

The fixed-parameter tractability of KROM-abduction parameterized by vertex cover number follows from the FPT result for parameter treewidth [47] and the fact that $tw(G) \leq \tau(G)$ for every graph G. We show that this parameterization does not lead to a polynomial kernel.

Theorem 74. ABD[KROM] *and* ABD[KROM]$_{\leq/=}$ *parameterized by* τ *do not admit a polynomial kernel unless the Polynomial Hierarchy collapses.*

Proof. We show this by PPT-reduction from SAT of $\varphi \in$ CNF parameterized by the number of variables $\mathit{var}(\varphi)$. This problem does not admit a polynomial kernel unless the Polynomial Hierarchy collapses [16]. Given φ we create an abduction instance $\langle V, H, M, T\rangle$ as follows. Let $V := X \cup X' \cup M$, where $X := \mathit{var}(\varphi)$, $X' := \{x' \mid x \in X\}$, and M contains a manifestation for each clause in φ. Let $H := X \cup X'$, and

$$
\begin{aligned}
T := \bigwedge_{x\in X} \big((x \vee x') \wedge (\neg x \vee \neg x')\big) \wedge\\
\bigwedge_{c\in\varphi} \left(\bigwedge_{x\in c} (x \to c) \wedge \bigwedge_{\neg x\in c} (x' \to c) \right).
\end{aligned}
$$

For instances of ABD[KROM]$_{\leq/=}$, we additionally set $k := |X|$. Observe that the primal graph of T can be covered by the set $X \cup X'$. Thus, τ can be bounded by $2 \cdot |\mathit{var}(\varphi)|$. □

Next we will show that ABD[KROM] and and ABD[KROM]$_{\leq/=}$ have a polynomial kernel when parameterized by (H, M). Thereby $\mathit{TrimRes}$ from Definition 64 is used as a kernelization function.

Lemma 75. *Let $\langle V, H, M, T\rangle$ be an abduction instance for* KROM *theories, let $S \subseteq H$, let $m \in M$, and let $T \wedge S$ be satisfiable. Then $T \wedge S \wedge \neg m$ is unsatisfiable if and only if $\mathit{TrimRes}(T, H, M) \wedge S \wedge \neg m$ is unsatisfiable.*

Proof. Similar to the argument in the proof of Lemma 66, $\mathit{Mod}(T \wedge S) \subseteq \mathit{Mod}(\mathit{TrimRes}(T) \wedge S)$. Therefore, $\mathit{TrimRes}(T) \wedge S \wedge \neg m$ being unsatisfiable implies that $T \wedge S \wedge \neg m$ is unsatisfiable. For the other direction assume that $T \wedge S \wedge \neg m$ is unsatisfiable. By Lemma 67 this means that either $\{m\} \in \mathit{TrimRes}(T)$ or there exists some $h \in S$ with $\{\neg h, m\} \in \mathit{TrimRes}(T)$. In either case $\mathit{TrimRes}(T) \wedge S \wedge \neg m$ is unsatisfiable. □

Theorem 76. ABD[KROM] *and* ABD[KROM]$_{\leq/=}$ *have a polynomial kernel when parameterized by (H, M).*

Proof. Given an instance $\langle V, H, M, T\rangle$ with $T \in$ KROM. We can test T for unsatisfiability in polynomial time and output a trivial no-instance in case the answer is yes. Otherwise we compute in polynomial time $\mathit{TrimRes}(T, H, M)$ which has size $O((|H| + |M|)^2)$. Indeed $\langle H \cup M, H, M, \mathit{TrimRes}(T, H, M)\rangle$ gives a kernel for our instance. Given a set $S \subseteq H$, by Lemma 66 testing the satisfiability of $T \wedge S$ is equivalent to testing $\mathit{TrimRes}(T, H, M) \wedge S$. Testing whether $T \wedge S \models M$ is equivalent to testing if $T \wedge S \models m$ for all $m \in M$. By Lemma 75 each of those tests can be done on $\mathit{TrimRes}(T, H, M)$. □

It follows immediately that ABD[KROM] and ABD[KROM]$_{\leq/=}$ have a polynomial kernel when parameterized by V. Note that these problems are already FPT when parameterized by H alone.

CHAPTER 7

Conclusion

7.1 Summary

Many computational tasks in the area of nonmonotonic reasoning are intractable. To overcome this obstacle, traditionally research was focused on studying restricted syntactical fragments such as Horn or Krom formulas. Often, such a restriction indeed leads to a lower computational complexity of these problems. But this does not necessarily yield tractability. For example propositional abduction is still NP-complete when restricted to Horn formulas.

Another way of dealing with NP-hard problems comes from the area of parameterized complexity theory. Thereby one studies the complexity not only in terms of the input size but also with respect to one or more (structural) parameters. The quest is to find an appropriate set of parameters such that the problem becomes fixed-parameter tractable. This usually means that the problem can be efficiently solved as long as the parameter values remain sufficiently small.

While the field of parameterized complexity theory is growing rapidly, most of the work so far has been focused on problems from the area of graph theory. Although parameterized complexity in the areas of AI and reasoning is not new, it is still less important there as compared to graph theory. This means of course that there are a lot of problems in nonmonotonic reasoning which have not yet been studied from a parameterized point of view. Additionally, for problems that have been considered already, there is still a lot of improvement possible.

In this thesis we presented research in both these directions. We started a parameterized analysis of belief revision problems and problems from propositional abduction. Furthermore, we gained new results for problems in the area of answer set programming.

7.1.1 Belief Revision

In Chapter 4, we have identified new tractable classes of revision problems with respect to three of the most fundamental approaches [20, 102, 115]. Moreover, we provided novel dynamic programming algorithms for Dalal's revision operator [20] (i.e. for the problem of deciding

$\alpha \circ_D \beta \models \gamma$, and enumerating the models of $\alpha \circ_D \beta$) which run in linear time (resp. with linear delay) if the treewidth of the revision scenario is bounded.

To the best of our knowledge, neither Courcelle's Theorem [17] (or one of its extensions such as [4]) nor dynamic programming approaches (along the lines of tree decompositions) have been applied to belief revision problems, so far. Although, for other AI and reasoning formalisms such approaches already proved to be successful (see, e.g., [47, 57]).

7.1.2 Answer Set Programming

In Chapter 5, we have shown how the notion of bounded treewidth can be used to identify tractable fragments of answer-set programming with weight constraints. However, by proving hardness results, we have also shown that a straightforward application of treewidth is not sufficient to achieve the desired tractability.

The upper bounds on the time complexity of our dynamic programming algorithms were obtained by very coarse estimates (see Theorems 44, 45, 46). In particular, we assumed straightforward methods for storing and manipulating bag assignments. Using sophisticated methods and data structures in implementing the functionality of the different node types of our algorithm should eventually result in a further improvement of the (theoretical) upper bounds on the time complexity provided in this paper.

7.1.3 Abduction

We have drawn a detailed picture of the parameterized complexity of abduction as depicted in Tables 6.1 and 6.3. We gained a number of results for abduction in general as well as for restrictions to Horn, definite Horn, and Krom theories. We did not only study the usual decision problem that asks whether an instance admits a solution. Instead we also introduced new decision problems that explicitly ask for small solution. The obtained results include fixed-parameter tractability, completeness results for the classes W[1], W[2], W[P], para-NP-hardness results, as well as polynomial kernels and results that such kernels are impossible. Although there are many cases where ABD[HORN] is FPT, in non of the considered cases it admits a polynomial kernel. For ABD[KROM] we were able to show the existence of a polynomial kernel when parameterized by the number of hypotheses plus the number manifestations.

7.2 Discussion

In this work, we have presented several fixed-parameter tractable algorithms for problems parameterized by treewidth without addressing the problem of actually computing a tree decomposition of appropriate width. As has been mentioned earlier, Bodlaender [7] showed that deciding whether a graph has treewidth $\leq w$, and if this is the case then computing a tree decomposition of width w is fixed-parameter linear for parameter w. Unfortunately, this linear time algorithm is only of theoretical interest and the practical usefulness is limited [62]. However, considerable progress has been recently made in developing heuristic-based tree decomposition algorithms, which can handle graphs with moderate size of several hundreds of vertices [62, 12, 112, 13, 58].

Note that for our algorithms it is not necessary to have tree decompositions of optimal width. Of course the runtime of these algorithms, which depends exponentially on the width of the tree decomposition, will be worse. But we expect that in practice this is compensated by a faster computation of the tree decomposition.

7.3 Comparison with Related Work

Recently, a meta-theorem for MSO problems on graphs with cardinality and weight constraints was shown [109]. This meta-theorem allows one to handle cardinality constraints with respect to sets that occur as free variables in the corresponding MSO formula. For ASP programs Π with cardinality constraints (PCC) or weight constraints (PWC) with weights in unary and bounded treewidth (see Chapter 5), it is possible to derive from this meta-theorem a polynomial time algorithm for checking whether Π has a model. However, in order to check whether Π has a *stable* model, one needs to handle cardinality constraints with respect to sets that occur as quantified variables in the MSO formula, which is not possible with the above-mentioned meta-theorem.

We have already mentioned the dynamic programming algorithm for ASP by Jakl, Pichler, and Woltran [57]. This algorithm works for programs without cardinality or weight constraints, but possibly with disjunction in the head of the rules. The data structure manipulated at each node for this ASP algorithm is conceptually much simpler than the one used here: Potential models of the given program are represented by the so-called tree-models. A tree-model consists of a subset of the atoms in a bag (the ones that are true in the models thus represented) and a subset of rules in a bag (the ones that are validated by the models thus represented).

However, to handle the minimality condition on stable models, it is not sufficient to propagate potential models along the bottom-up traversal of the tree decomposition. In addition, it is required, for each potential model M, to keep track of all those models of the reduct with respect to M that would prevent M from being minimal. Those models are represented by a set of tree-models accompanying each tree-model. Hence, despite the simplicity of the data structure, the time complexity of the algorithm by Jakl, Pichler, and Woltran [57] is *double exponential* in the treewidth, since it has to handle *sets of subsets of the bag* at each node. Therefore, rather than extending that algorithm by mechanisms to handle weight or cardinality constraints, we have presented here an algorithm based on a completely different data structure – in particular, keeping track of orderings of the atoms. We have thus managed to obtain an algorithm whose time complexity is only *single exponential* in the treewidth.

7.4 Future Work and Open Issues

A direction of future research is to apply our methods to approaches for iterated belief revision. This however calls for the additional requirement that the outcome of a single revision step has to be of bounded treewidth as well. An interesting research question of its own is how to ensure such a property.

Furthermore, It is an open problem whether the revision operator due to Forbus [43] is fixed-parameter tractable with respect to treewidth. Since the operator requires to compare cardinali-

ties of different sets, it seems to be not possible to express this problem in one of the extensions of MSO for which fixed-parameter tractability would follow.

Another direction of future work is to extend the parameterized complexity analysis and the development of efficient algorithms to further problems where weights or cardinalities play a role. Note that weights are a common feature in the area of knowledge representation and reasoning, for instance, to express costs or probabilities.

Recall the solvability problem for propositional abduction restricted to definite Horn theories ABD[DEFHORN] (Chapter 6). This problem is FPT (Proposition 60). However, we could not yet settle the question whether this problem admits a polynomial kernel.

In general, future work includes the analysis of the parameterized complexity of further problems in the area of nonmonotonic reasoning. For the reasoning formalisms we studied so far it is of interest to search for further parameters yielding fixed-parameter tractability or to study other syntactical restrictions as well.

Bibliography

[1] David W. Aha, Dennis F. Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50(2):510–530, 1985.

[3] Michael Alekhnovich and Alexander A. Razborov. Satisfiability, branch-width and Tseitin tautologies. *Computational Complexity*, 20(4):649–678, 2011.

[4] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

[5] Christer Bäckström, Yue Chen, Peter Jonsson, Sebastian Ordyniak, and Stefan Szeider. The complexity of planning revisited – a parameterized analysis. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2012, Toronto, Ontario, Canada, July 22-26, 2012*. AAAI Press, 2012. To appear.

[6] Rachel Ben-Eliyahu and Rina Dechter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, 12(1-2):53–87, 1994.

[7] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[8] Hans L. Bodlaender. Treewidth: Characterizations, applications, and computations. In Fedor V. Fomin, editor, *Proceedings of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2006, Bergen, Norway, June 22-24, 2006*, volume 4271 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

[9] Hans L. Bodlaender. Treewidth: Structure and algorithms. In Giuseppe Prencipe and Shmuel Zaks, editors, *Proceedings of the 14th International Colloquium on Structural Information and Communication Complexity, SIROCCO 2007, Castiglioncello, Italy, June 5-8, 2007*, volume 4474 of *Lecture Notes in Computer Science*, pages 11–25. Springer, 2007.

[10] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009.

[11] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Preprocessing for treewidth: A combinatorial analysis through kernelization. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *Proceedings of the 38th International Colloquium on Automata, Languages and Programming, ICALP 2011, Part I, Zurich, Switzerland, July 4-8, 2011*, volume 6755 of *Lecture Notes in Computer Science*, pages 437–448. Springer, 2011.

[12] Hans L. Bodlaender and Arie M. C. A. Koster. Safe separators for treewidth. *Discrete Mathematics*, 306(3):337–350, 2006.

[13] Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008.

[14] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theoretical Computer Science*, 412(35):4570–4578, 2011.

[15] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[16] Yijia Chen, Jörg Flum, and Moritz Müller. Lower bounds for kernelizations and other preprocessing procedures. *Theory of Computing Systems*, 48(4):803–839, 2011.

[17] Bruno Courcelle. Graph rewriting: An algebraic and logic approach. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 193–242. Elsevier Science Publishers, 1990.

[18] Nadia Creignou, Johannes Schmidt, and Michael Thomas. Complexity of propositional abduction for restricted sets of Boolean functions. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczyński, editors, *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*. AAAI Press, 2010.

[19] Nadia Creignou and Bruno Zanuttini. A complete classification of the complexity of propositional abduction. *SIAM Journal on Computing*, 36(1):207–229, 2006.

[20] Mukesh Dalal. Investigations into a theory of knowledge base revision. In Howard E. Shrobe, Tom M. Mitchell, and Reid G. Smith, editors, *Proceedings of the 7th National Conference on Artificial Intelligence, AAAI 1988, St. Paul, MN, August 21-26, 1988*, pages 475–479. AAAI Press / The MIT Press, 1988.

[21] Adnan Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, 11(1-2):11–34, 2001.

[22] Artan Dermaku, Tobias Ganzow, Georg Gottlob, Benjamin J. McMahan, Nysret Musliu, and Marko Samer. Heuristic methods for hypertree decomposition. In Alexander F. Gelbukh and Eduardo F. Morales, editors, *Proceedings of the 7th Mexican International Conference on Artificial Intelligence, MICAI 2008: Advances in Artificial Intelligence, Atizapán de Zaragoza, Mexico, October 27-31, 2008*, volume 5317 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 2008.

[23] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Incompressibility through colors and IDs. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Proceedings of the 36th International Colloquium on Automata, Languages and Programming, ICALP 2009, Part I, Rhodes, Greece, July 5-12, 2009*, volume 5555 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2009.

[24] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999.

[25] Rodney G. Downey, Michael R. Fellows, and Kenneth W. Regan. Descriptive complexity and the W hierarchy. In *Proof Complexity and Feasible Arithmetic*, volume 39 of *AMS-DIMACS Volume Series*, pages 119–134. AMS, 1998.

[26] Rodney G. Downey, Michael R. Fellows, and Ulrike Stege. Parameterized complexity: A framework for systematically confronting computational intractability. In Ronald L. Graham, Jan Kratochvíl, Jaroslav Nesetril, and Fred S. Roberts, editors, *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, volume 49 of *AMS-DIMACS Proceedings Serie*, pages 49–99. American Mathematical Society, 1999.

[27] Paul E. Dunne. Computational properties of argument systems satisfying graph-theoretic constraints. *Artificial Intelligence*, 171(10-15):701–729, 2007.

[28] Wolfgang Dvorák, Sebastian Ordyniak, and Stefan Szeider. Augmenting tractable fragments of abstract argumentation. *Artificial Intelligence*, 186:157–173, 2012.

[29] Wolfgang Dvorák, Reinhard Pichler, and Stefan Woltran. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artificial Intelligence*, 186:1–37, 2012.

[30] Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57(2-3):227–270, 1992.

[31] Thomas Eiter and Georg Gottlob. The complexity of logic-based abduction. *Journal of the ACM*, 42(1):3–42, 1995.

[32] Thomas Eiter and Georg Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence*, 15(3-4):289–323, 1995.

[33] Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. *Artificial Intelligence*, 172(12-13):1495–1539, 2008.

[34] Esra Erdem. Applications of answer set programming in phylogenetic systematics. In Marcello Balduccini and Tran Cao Son, editors, *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning - Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday*, volume 6565 of *Lecture Notes in Computer Science*, pages 415–431. Springer, 2011.

[35] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009.

[36] Michael R. Fellows, Andreas Pfandler, Frances A. Rosamond, and Stefan Rümmele. The parameterized complexity of abduction. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2012, Toronto, Ontario, Canada, July 22-26, 2012*. AAAI Press, 2012. To appear.

[37] Henning Fernau. Roman domination: a parameterized perspective. *International Journal of Computer Mathematics*, 85(1):25–38, 2008.

[38] Johannes Klaus Fichte and Stefan Szeider. Backdoors to tractable answer-set programming. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 863–868. IJCAI/AAAI, 2011.

[39] Eldar Fischer, Johann A. Makowsky, and Elena V. Ravve. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discrete Applied Mathematics*, 156(4):511–529, 2008.

[40] Jörg Flum and Martin Grohe. Describing parameterized complexity classes. *Information and Computation*, 187(2):291–319, 2003.

[41] Jörg Flum and Martin Grohe. Model-checking problems as a basis for parameterized intractability. *Logical Methods in Computer Science*, 1(1), 2005.

[42] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[43] Kenneth D. Forbus. Introducing actions into qualitative simulation. In N. S. Sridharan, editor, *Proceedings of the 11th International Joint Conference on Artificial Intelligence, IJCAI 1989, Detroit, MI, USA, August 1989*, pages 1273–1278. Morgan Kaufmann, 1989.

[44] Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. clasp: A conflict-driven answer set solver. In Chitta Baral, Gerhard Brewka, and John S. Schlipf, editors, *Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2007, Tempe, AZ, USA, May 15-17, 2007*, volume 4483 of *Lecture Notes in Computer Science*, pages 260–265. Springer, 2007.

[45] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.

[46] Enrico Giunchiglia, Yuliya Lierler, and Marco Maratea. Answer set programming based on propositional satisfiability. *Journal of Automated Reasoning*, 36(4):345–377, 2006.

[47] Georg Gottlob, Reinhard Pichler, and Fang Wei. Bounded treewidth as a key to tractability of knowledge representation and reasoning. *Artificial Intelligence*, 174(1):105–132, 2010.

[48] Georg Gottlob, Francesco Scarcello, and Martha Sideri. Fixed-parameter complexity in AI and nonmonotonic reasoning. *Artificial Intelligence*, 138(1-2):55–86, 2002.

[49] Georg Gottlob and Stefan Szeider. Fixed-parameter algorithms for artificial intelligence, constraint satisfaction and database problems. *The Computer Journal*, 51(3):303–325, 2008.

[50] Gianluigi Greco and Francesco Scarcello. Structural tractability of enumerating CSP solutions. In David Cohen, editor, *Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010*, volume 6308 of *Lecture Notes in Computer Science*, pages 236–251. Springer, 2010.

[51] Gianluigi Greco and Francesco Scarcello. Structural tractability of constraint optimization. In Jimmy Ho-Man Lee, editor, *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming, CP 2011, Perugia, Italy, September 12-16, 2011*, volume 6876 of *Lecture Notes in Computer Science*, pages 340–355. Springer, 2011.

[52] Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1), 2007.

[53] Martin Grohe and Dániel Marx. Constraint solving via fractional edge covers. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 289–298. ACM Press, 2006.

[54] Youssef Hamadi, Saïd Jabbour, and Lakhdar Sais. ManySAT: a parallel SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 6(4):245–262, 2009.

[55] Jerry R. Hobbs, Mark E. Stickel, Douglas E. Appelt, and Paul A. Martin. Interpretation as abduction. *Artificial Intelligence*, 63(1-2):69–142, 1993.

[56] Sang il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006.

[57] Michael Jakl, Reinhard Pichler, and Stefan Woltran. Answer-set programming with bounded treewidth. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009, Pasadena, California, USA, July 11-17, 2009*, pages 816–822, 2009.

[58] Kalev Kask, Andrew Gelfand, Lars Otten, and Rina Dechter. Pushing the power of stochastic greedy ordering schemes for inference in graphical models. In Wolfram Burgard and Dan Roth, editors, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*, pages 54–60. AAAI Press, 2011.

[59] Hirofumi Katsuno and Alberto O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52(3):263–294, 1992.

[60] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

[61] Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61(2):302–332, 2000.

[62] Arie M. C. A. Koster, Hans L. Bodlaender, and Stan P. M. van Hoesel. Treewidth: Computational experiments. *Electronic Notes in Discrete Mathematics*, 8:54–57, 2001.

[63] Martin Lackner and Andreas Pfandler. Fixed-parameter algorithms for closed world reasoning. In *Proceedings of the 20th European Conference on Artificial Intelligence, ECAI 2012, Montpellier, France, August 27-31, 2012*. IOS Press, 2012. To appear.

[64] Martin Lackner and Andreas Pfandler. Fixed-parameter algorithms for finding minimal models. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning, KR 2012, Rome, Italy, June 10-14, 2012*, pages 85–95. AAAI Press, 2012.

[65] Alexander Leitsch. *The resolution calculus*. Texts in theoretical computer science. Springer, 1997.

[66] Nicola Leone, Gianluigi Greco, Giovambattista Ianni, Vincenzino Lio, Giorgio Terracina, Thomas Eiter, Wolfgang Faber, Michael Fink, Georg Gottlob, Riccardo Rosati, Domenico Lembo, Maurizio Lenzerini, Marco Ruzzi, Edyta Kalka, Bartosz Nowicki, and Witold Staniszkis. The INFOMIX system for advanced integration of incomplete and inconsistent data. In Fatma Özcan, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, pages 915–917. ACM, 2005.

[67] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.

[68] Paolo Liberatore and Marco Schaerf. Belief revision and update: Complexity of model checking. *Journal of Computer and System Sciences*, 62(1):43–72, 2001.

[69] Vladimir Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138(1-2):39–54, 2002.

[70] Guohua Liu and Jia-Huai You. Level mapping induced loop formulas for weight constraint and aggregate logic programs. *Fundamenta Informaticae*, 106(1):25–43, 2011.

[71] Zbigniew Lonc and Miroslaw Truszczyński. Fixed-parameter complexity of semantics for logic programs. *ACM Transactions on Computational Logic*, 4(1):91–119, 2003.

[72] V. Wiktor Marek and Miroslaw Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38(3):588–619, 1991.

[73] Victor W. Marek and Miroslaw Truszczyński. Stable models and an alternative logic programming paradigm. In Krzysztof R. Apt, Victor W. Marek, Miroslaw Truszczyński, and David S. Warren, editors, *The Logic Programming Paradigm: A 25-Year Perspective*, pages 375–398. Springer, 1999.

[74] Michael Morak, Nysret Musliu, Reinhard Pichler, Stefan Rümmele, and Stefan Woltran. A new tree-decomposition based algorithm for answer set programming. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2011, Boca Raton, FL, USA, November 7-9, 2011*, pages 916–918. IEEE, 2011.

[75] Michael Morak, Nysret Musliu, Reinhard Pichler, Stefan Rümmele, and Stefan Woltran. Evaluating tree-decomposition based algorithms for answer set programming. In *Proceedings of the 6th International Conference on Learning and Intelligent Optimization, LION 6, Paris, France, January 16-20, 2012*, Lecture Notes in Computer Science. Springer, 2012. To appear.

[76] Michael Morak, Reinhard Pichler, Stefan Rümmele, and Stefan Woltran. A dynamic-programming based ASP-solver. In Tomi Janhunen and Ilkka Niemelä, editors, *Proceedings of the 12th European Conference on Logics in Artificial Intelligence, JELIA 2010, Helsinki, Finland, September 13-15, 2010*, volume 6341 of *Lecture Notes in Computer Science*, pages 369–372. Springer, 2010.

[77] Charles G. Morgan. Hypothesis generation by machine. *Artificial Intelligence*, 2(2):179–187, 1971.

[78] Hwee Tou Ng and Raymond J. Mooney. Abductive plan recognition and diagnosis: A comprehensive empirical evaluation. In Bernhard Nebel, Charles Rich, and William R. Swartout, editors, *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning, KR 92, Cambridge, MA, October 25-29, 1992*, pages 499–508. Morgan Kaufmann, 1992.

[79] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, 2006.

[80] Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):241–273, 1999.

[81] Ilkka Niemelä and Patrik Simons. Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In Jürgen Dix, Ulrich Furbach, and Anil Nerode, editors, *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR'97, Dagstuhl Castle, Germany, July 28-31, 1997*, volume 1265 of *Lecture Notes in Computer Science*, pages 421–430. Springer, 1997.

[82] Ilkka Niemelä, Patrik Simons, and Timo Soininen. Stable model semantics of weight constraint rules. In Michael Gelfond, Nicola Leone, and Gerald Pfeifer, editors, *Proceedings of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 1999, El Paso, Texas, USA, December 2-4, 1999*, volume 1730 of *Lecture Notes in Computer Science*, pages 317–331. Springer, 1999.

[83] Naomi Nishimura, Prabhakar Ragde, and Stefan Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In *Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing, SAT 2004, 10-13 May 2004, Vancouver, BC, Canada*, 2004.

[84] Naomi Nishimura, Prabhakar Ragde, and Stefan Szeider. Solving #SAT using vertex covers. *Acta Informatica*, 44(7-8):509–523, 2007.

[85] Monica Nogueira, Marcello Balduccini, Michael Gelfond, Richard Watson, and Matthew Barry. An A-Prolog decision support system for the space shuttle. In I. V. Ramakrishnan, editor, *Proceedings of the Third International Symposium on Practical Aspects of Declarative Languages, PADL 2001, Las Vegas, Nevada, March 11-12, 2001*, volume 1990 of *Lecture Notes in Computer Science*, pages 169–183. Springer, 2001.

[86] Gustav Nordh and Bruno Zanuttini. What makes propositional abduction tractable. *Artificial Intelligence*, 172(10):1245–1284, 2008.

[87] Sebastian Ordyniak and Stefan Szeider. Algorithms and complexity results for exact Bayesian structure learning. In Peter Grünwald and Peter Spirtes, editors, *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, UAI 2010, Catalina Island, CA, USA, July 8-11, 2010*, pages 401–408. AUAI Press, 2010.

[88] Christos H. Papadimitriou and Mihalis Yannakakis. On the complexity of database queries. *Journal of Computer and System Sciences*, 58(3):407–427, 1999.

[89] Charles Sanders Peirce. Abduction and induction. In Justus Buchler, editor, *Philosophical Writings of Peirce*, pages 150–156. Dover, 1955.

[90] Pavlos Peppas. Belief revision. In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, *Handbook of Knowledge Representation*, pages 317–360. Elsevier, 2007.

[91] Reinhard Pichler, Stefan Rümmele, Stefan Szeider, and Stefan Woltran. Tractable answer-set programming with weight constraints: Bounded treewidth is not enough. In Fangzhen

Lin, Ulrike Sattler, and Miroslaw Truszczyński, editors, *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*, pages 508–517. AAAI Press, 2010.

[92] Reinhard Pichler, Stefan Rümmele, Stefan Szeider, and Stefan Woltran. Tractable answer-set programming with weight constraints: Bounded treewidth is not enough. *Theory and Practice of Logic Programming*, 2012. To appear.

[93] Reinhard Pichler, Stefan Rümmele, and Stefan Woltran. Belief revision with bounded treewidth. In Esra Erdem, Fangzhen Lin, and Torsten Schaub, editors, *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2009, Potsdam, Germany, September 14-18, 2009*, volume 5753 of *Lecture Notes in Computer Science*, pages 250–263. Springer, 2009.

[94] Reinhard Pichler and Stefan Woltran. The complexity of handling minimal solutions in logic-based abduction. In Helder Coelho, Rudi Studer, and Michael Wooldridge, editors, *Proceedings of the 19th European Conference on Artificial Intelligence, ECAI 2010, Lisbon, Portugal, August 16-20, 2010*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 895–900. IOS Press, 2010.

[95] Harry E. Pople. On the mechanization of abductive logic. In Nils J. Nilsson, editor, *Proceedings of the 3rd International Joint Conference on Artificial Intelligence, IJCAI 1973, Standford, CA, August 1973*, pages 147–152. William Kaufmann, 1973.

[96] J. Ross Quinlan. Learning with continuous classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, AI'92, Singapore*, pages 343–348, 1992.

[97] Neil Robertson and Paul D. Seymour. Graph minors. I. Excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39–61, 1983.

[98] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

[99] Marko Samer and Stefan Szeider. Backdoor sets of quantified Boolean formulas. *Journal of Automated Reasoning*, 42(1):77–97, 2009.

[100] Marko Samer and Stefan Szeider. Algorithms for propositional model counting. *Journal of Discrete Algorithms*, 8(1):50–64, 2010.

[101] Marko Samer and Stefan Szeider. Constraint satisfaction with bounded treewidth revisited. *Journal of Computer and System Sciences*, 76(2):103–114, 2010.

[102] Ken Satoh. Nonmonotonic reasoning by minimal belief revision. In *Proceedings of the International Conference on Fifth Generation Computer Systems, FGCS 1988, Tokyo, Japan, November 28-December 2, 1988*, pages 455–462, 1988.

[103] Bart Selman and Hector J. Levesque. Abductive and default reasoning: A computational core. In Howard E. Shrobe, Thomas G. Dietterich, and William R. Swartout, editors, *Proceedings of the 8th National Conference on Artificial Intelligence, AAAI 1990, Boston, Massachusetts, July 29 - August 3, 1990*, pages 343–348. AAAI Press / The MIT Press, 1990.

[104] Timo Soininen and Ilkka Niemelä. Developing a declarative rule language for applications in product configuration. In Gopal Gupta, editor, *Proceedings of the First International Workshop on Practical Aspects of Declarative Languages, PADL '99, San Antonio, Texas, USA, January 18-19, 1999*, volume 1551 of *Lecture Notes in Computer Science*, pages 305–319. Springer, 1999.

[105] Stefan Szeider. On fixed-parameter tractable parameterizations of SAT. In Enrico Giunchiglia and Armando Tacchella, editors, *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing, SAT 2003, Santa Margherita Ligure, Italy, May 5-8, 2003*, volume 2919 of *Lecture Notes in Computer Science*, pages 188–202. Springer, 2003.

[106] Stefan Szeider. Backdoor sets for DLL subsolvers. *Journal of Automated Reasoning*, 35(1-3):73–88, 2005.

[107] Stefan Szeider. Not so easy problems for tree decomposable graphs. In *Advances in discrete mathematics and applications: Mysore, 2008*, volume 13 of *Ramanujan Mathematical Society Lectures Notes Series*, pages 179–190. Ramanujan Mathematical Society, Mysore, 2010.

[108] Stefan Szeider. Limits of preprocessing. In Wolfram Burgard and Dan Roth, editors, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press, 2011.

[109] Stefan Szeider. Monadic second order logic on graphs with local cardinality constraints. *ACM Transactions on Computational Logic*, 12(2):12, 2011.

[110] Miroslaw Truszczyński. Computing large and small stable models. *Theory and Practice of Logic Programming*, 2(1):1–23, 2002.

[111] Miroslaw Truszczyński. Trichotomy and dichotomy results on the complexity of reasoning with disjunctive logic programs. *Theory and Practice of Logic Programming*, 11(6):881–904, 2011.

[112] Frank van den Eijkhof, Hans L. Bodlaender, and Arie M. C. A. Koster. Safe reduction rules for weighted treewidth. *Algorithmica*, 47(2):139–158, 2007.

[113] Yong Wang and Ian H. Witten. Induction of model trees for predicting continuous classes. In *Proceedings of the Poster papers of the 9th European Conference on Machine Learning, Prague, Czech Republic*, 1997.

[114] Mary-Anne Williams. Applications of belief revision. In Burkhard Freitag, Hendrik Decker, Michael Kifer, and Andrei Voronkov, editors, *Transactions and Change in Logic Databases, International Seminar on Logic Databases and the Meaning of Change, Schloss Dagstuhl, Germany, September 23-27, 1996 and ILPS '97 Post-Conference Workshop on (Trans)Actions and Change in Logic Programming and Deductive Databases, (DYNAMICS'97) Port Jefferson, NY, USA, October 17, 1997*, volume 1472 of *Lecture Notes in Computer Science*, pages 287–316. Springer, 1998.

[115] Marianne Winslett. Reasoning about action using a possible models approach. In Howard E. Shrobe, Tom M. Mitchell, and Reid G. Smith, editors, *Proceedings of the 7th National Conference on Artificial Intelligence, AAAI 1988, St. Paul, MN, August 21-26, 1988*, pages 89–93. AAAI Press / The MIT Press, 1988.

[116] Masaki Yamamoto. An improved $O(1.234^m)$-time deterministic algorithm for SAT. In Xiaotie Deng and Ding-Zhu Du, editors, *Proceedings of the 16th International Symposium on Algorithms and Computation, ISAAC 2005, Sanya, Hainan, China, December 19-21, 2005*, volume 3827 of *Lecture Notes in Computer Science*, pages 644–653. Springer, 2005.