Tractable Answer-Set Programming with Weight Constraints: Bounded Treewidth is not Enough

Reinhard Pichler[*] and **Stefan Rümmele**[*] and **Stefan Szeider**[†] and **Stefan Woltran**[*]
Institute of Information Systems, Vienna University of Technology, A-1040 Vienna, Austria.
email: {pichler, ruemmele, woltran}@dbai.tuwien.ac.at, stefan@szeider.net

Abstract

Cardinality constraints or, more generally, weight constraints are well recognized as an important extension of answer-set programming. Clearly, all common algorithmic tasks related to programs with cardinality or weight constraints (PWCs) – like checking the consistency of a program – are intractable. Many intractable problems in the area of knowledge representation and reasoning have been shown to become tractable if the treewidth of the programs or formulas under consideration is bounded by some constant. The goal of this paper is to apply the notion of treewidth to PWCs and to identify tractable fragments. It will turn out that the straightforward application of treewidth to PWCs does not suffice to obtain tractability. However, by imposing further restrictions, tractability can be achieved.

Introduction

Answer-set programming has evolved as a paradigm that allows for very elegant solutions to many combinatorial problems (Marek and Truszczyński 1999). The basic idea is to describe a problem by a logic program in such a way that the stable models correspond to the solutions of the considered problem. By extending logic programs with cardinality or, more generally, weight constraints, an even larger class of problems is accessible to this method (Niemelä, Simons, and Soininen 1999). 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.

In this paper, we restrict ourselves to *normal* logic programs with cardinality constraints (PCCs, for short) or weight constraints (PWCs, for short). 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. An interesting approach to dealing with intractable problems comes from parameterized complexity theory and is based on the following observation: Many hard problems become tractable if some parameter that represents a structural aspect of the problem instance is small. One important parameter is treewidth, which measures the "tree-likeness" of a graph or, more generally, of a structure. In the area of knowledge representation and reasoning (KR & R), many tractability results for instances of bounded treewidth have been recently proven (Gottlob, Pichler, and Wei 2006). The goal of this work is to obtain tractability results via bounded treewidth also for PCCs and PWCs. It will turn out that the straightforward application of treewidth to PWCs does not suffice to obtain tractability. However, by imposing further restrictions, tractability can be achieved.

Main results of the paper

- We show that the consistency problem of PWCs remains NP-complete even if the treewidth of the considered programs is bounded by a constant (actually, even if this constant is 1). Hence, we have to search for further restrictions on the PWCs to ensure tractability.

- We thus consider the largest integer occurring in (lower and upper) bounds of the constraints in the PWC, and call this parameter constraint-width. If also the constraint-width is bounded by an arbitrary but fixed constant, then the consistency problem of PWCs becomes *linear time tractable* (the bound on the running time entails a constant factor that is exponential in constraint-width and treewidth).

- For PCCs (i.e., PWCs where all weights are equal to 1) we obtain *non-uniform polynomial time* tractability by designing a new dynamic programming algorithm, i.e.: Let w denote the treewidth of a PCC and let n denote the size of the PCC. Then our algorithm works in time $O(f(w) \cdot n^{2w})$ 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. We shall also discuss further extensions of this dynamic programming algorithm for PCCs, e.g.: 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 operates in time $O(f(w) \cdot n^{O(1)})$ would be preferable, i.e., the parameter w does not occur in the exponent of the program size n. A problem with such an algorithm

[*]Supported by the Austrian Science Fund (FWF), project P20704-N18.
[†]Supported by the European Research Council (ERC), project 239962.

is called *fixed-parameter tractable*. Alas, we show that under common complexity theoretical assumptions no such algorithm exists. Technically, we prove that the consistency problem of PCCs parameterized by treewidth is hard for the parameterized complexity class W[1], i.e., 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.

Structure of the paper After recalling the necessary background, we prove the NP-completeness of the consistency problem of PWCs in case of binary representation of the weights. Subsequently, we show the linear fixed-parameter tractability of the problem if we consider the treewidth plus the size of the bounds as parameter. Afterwards, the non-uniform polynomial-time upper bound for the consistency problem of PCCs is established by presenting a dynamic programming algorithm. By giving a W[1]-hardness proof in case of unary representation in the next section, we show that it is unlikely that this result can be significantly improved.

Background

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*, defined as follows: 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 $(a, j) \in Cl(c)$ respectively $(\neg a, j) \in Cl(c)$ for an arbitrary j. A rule $r \in \mathcal{R}$ is a pair (h, b) where $h \in \mathcal{C}$ (the head) and $b \subseteq \mathcal{C}$ (the body). We write $\text{H}(r) := h$ and $\text{B}(r) := b$. 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

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

I is a model of c (denoted by $I \models c$) iff $l(c) \leq W(c, I) \leq u(c)$. For a set $C \subseteq \mathcal{C}$, $I \models C \Leftrightarrow I \models c$ for all $c \in C$. Moreover, C is a model of a rule $r \in \mathcal{R}$ (denoted by $C \models r$) iff $\text{H}(r) \in C$ or $\text{B}(r) \nsubseteq C$. For a set $R \subseteq \mathcal{R}$, $C \models R \Leftrightarrow C \models r$ for all $r \in R$. I is a model of Π (denoted by $I \models \Pi$) iff $\{c \in \mathcal{C} : I \models c\} \models \mathcal{R}$. If the lower bound of a constraint $c \in \mathcal{C}$ is missing, we assume $l(c) = 0$. If the upper bound is missing, $I \models c$ iff $l(c) \leq W(c, I)$. A *program with cardinality constraints* (PCC) can be seen as a special case of a PWC, where each literal has weight 1.

Stable model semantics Given a PWC $\Pi = (A, \mathcal{C}, \mathcal{R})$ and an interpretation $I \subseteq A$. Following (Niemelä, Simons, and Soininen 1999), the reduct c^I of a constraint $c \in \mathcal{C}$ w.r.t. 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 w.r.t. 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 $h \in I \cap Cl(\text{H}(r))$ and $b = \{c^I : c \in \text{B}(r)\}$. Interpretation I is called a *stable model* (or *answer set*) of Π iff I is a model of Π and there exists no $J \subset I$ s.t. 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 asks, whether $\mathcal{AS}(\Pi) \neq \emptyset$.

Tree decompositions and treewidth A *tree decomposition* of a graph $\mathcal{G} = (V, E)$ is a pair $\mathcal{T} = (T, \chi)$, where T is a tree and χ maps each node n of T (we use $n \in T$ as a shorthand below) to a *bag* $\chi(n) \subseteq V$ s.t.

(1) for each $v \in V$, there is an $n \in T$ s.t. $v \in \chi(n)$;
(2) for each $(v, w) \in E$, there is an $n \in T$ s.t. $v, w \in \chi(n)$;
(3) for each $n_1, n_2, n_3 \in T$ s.t. n_2 lies on the path from n_1 to n_3, $\chi(n_1) \cap \chi(n_3) \subseteq \chi(n_2)$ holds.

A tree decomposition (T, χ) is called *normalized* (or *nice*) (Kloks 1994), if T is a rooted tree and the following conditions hold: (1) each $n \in T$ has ≤ 2 children; (2) for each $n \in T$ with two children n_1, n_2, $\chi(n) = \chi(n_1) = \chi(n_2)$; and (3) for each $n \in T$ with one child n', $\chi(n)$ and $\chi(n')$ differ in exactly one element.

The *width* of a tree decomposition is defined as the cardinality of its largest bag $\chi(n)$ minus one. It is known that every tree decomposition can be normalized in linear time without increasing the width (Kloks 1994). The *treewidth* of a graph $\mathcal{G}$, denoted as $tw(\mathcal{G})$, is the minimum width over all tree decompositions of $\mathcal{G}$. For arbitrary but fixed $w \geq 1$, it is feasible in linear time to decide whether a graph has treewidth $\leq w$ and, if so, to compute a tree decomposition of width w, see (Bodlaender 1996).

Treewidth and constraint-width of PWCs To build tree decompositions for programs, we use *incidence graphs*. For 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 $c \in \mathcal{C}$ iff $a \in Cl(c)$ or $\neg a \in Cl(c)$, and there is an edge between $c \in \mathcal{C}$ and $r \in \mathcal{R}$ iff $c \in \{\text{H}(r)\} \cup \text{B}(r)$. 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 1. *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\$ *s.t. the total cost is*

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

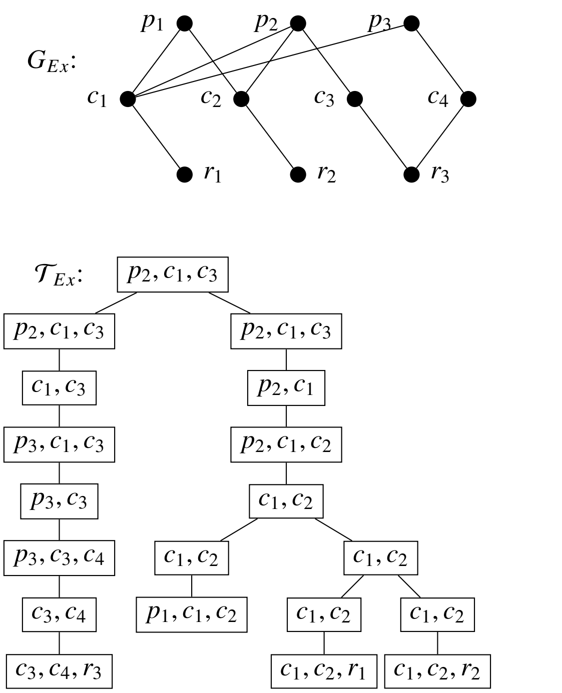

Figure 1: Incidence graph G_{Ex} and tree decomposition $\mathcal{T}_{Ex}$ of Example 1.

$\leq$ 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*

$$c_1 = (\{(p_1, 4), (p_2, 2), (p_3, 1)\}, 0, 5) \qquad r_1 = (c_1, \emptyset)$$
$$c_2 = (\{(p_1, 1), (p_2, 1)\}, 1, 2) \qquad r_2 = (c_2, \emptyset)$$
$$c_3 = (\{(p_2, 1)\}, 1, 1) \qquad r_3 = (c_3, \{c_4\})$$
$$c_4 = (\{(p_3, 1)\}, 1, 1)$$

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 Figure 1.*

NP-Completeness

Theorem 2. *The consistency problem for PWCs is NP-complete already for programs of treewidth 1.*

Proof. Clearly the problem is in NP. To show NP-hardness we reduce from the well-known NP-complete problem Partition. An instance of Partition is a collection of positive integers $X = \{x_1, \ldots, x_n\}$ (encoded in binary); the question is whether there exists a set $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} x_i = \sum_{i \notin I} x_i$. Given an instance $X = \{x_1, \ldots, x_n\}$, we construct a PWC $\Pi = (A, C, \mathcal{R})$ as follows. Let $S = \sum_{i=1}^{n} x_i$; we may assume that S is even since otherwise X is a no-instance and can immediately be rejected. We put $A = \{a_1, \ldots, a_n\}$, $C = \{c\}$ where $c = (\{(a_1, x_1), \ldots, (a_n, x_n)\}, S/2, S/2)$, and $\mathcal{R} = \{(c, \emptyset)\}$.

Claim 1: Π *has treewidth 1*. By construction the incidence graph of Π is a tree, hence of treewidth 1.

Claim 2: X *is a yes-instance of* Partition *iff* Π *has a model*. This claim follows easily from the definitions.

Claim 3. All models of Π *are stable*. Let M be a model of Π. Since each atom appears positively in a constraint at the head of a rule, and since all the rules have an empty body, it follows that the reduct Π^M is the conjunction of all the elements of M. Hence M is stable since no proper subset of M can satisfy Π^M. We conclude that X is a yes-instance of Partition iff Π is consistent.

It is evident that Π can be constructed from X in polynomial time. Hence, by Claims 1-3 we have a polynomial-time reduction from Partition to the consistency problem of PWCs of treewidth 1, and the theorem follows. $\square$

Note that Partition is "weakly NP-hard" since its NP-hardness depends on the binary encoding of the given integers. Accordingly, our reduction provides only weak NP-hardness for the consistency of PWCs of bounded treewidth. In fact, we shall prove later 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.

Linear-Time Tractability

Theorem 3. *The consistency problem for PWCs can be solved in linear time for instances whose treewidth and constraint-width are bounded by constants.*

To prove this result we shall take a logic approach and use Courcelle's Theorem (Courcelle 1987), see also (Downey and Fellows 1999; Flum and Grohe 2006). To this aim we consider *Monadic Second Order* (MSO) logic on labeled graphs in terms of their incidence structure whose universe contains vertices and edges. We assume an infinite supply of individual variables $x, x_1, x_2, \ldots$ and set variables $X, X_1, X_2, \ldots$ The atomic formulas are $E(x)$ ("x is an edge"), $V(x)$ ("x is a vertex"), $I(x, y)$ ("vertex x is incident with edge y"), $x = y$ (equality), and $X(y)$ ("element y belongs to set X"). Further we assume that a vertex or edge x can be labeled with an element a of some fixed finite set, denoted by the atomic formula $P_a(x)$. MSO formulas are built up from atomic formulas using the usual Boolean connectives ($\neg, \wedge, \vee$), quantification over individual variables ($\forall x$, $\exists x$), and quantification over set variables ($\forall X$, $\exists X$).

We write $G \models \varphi$ to indicate that an MSO formula φ is true for the labeled graph G. Courcelle's Theorem states that $G \models \varphi$ can be checked in linear time for labeled graphs whose treewidth is bounded by a constant.

Let k be a constant and consider a PWC $\Pi = (A, C, \mathcal{R})$ of constraint-width k. We encode all the information of Π by adding edge and vertex labels to the incidence graph of Π. We use the edge labels $+, -$ to indicate polarity of literals and the labels h, b to distinguish between head and body of rules. That is, an edge ac for $a \in A$ and $c \in C$ has label $+$ if $a \in Cl(c)$, and label $-$ if $\neg a \in Cl(c)$; an edge cr for $c \in C$ and $r \in \mathcal{R}$ has label h if $c = \mathrm{H}(r)$ and label b if $c \in \mathrm{B}(r)$. We use edge labels $1, \ldots, k+1$ to encode weights of literals (literals of weight 0 can be omitted, weights exceeding $k+1$ can be replaced by $k+1$). That is, an edge ac for $a \in A$ and $c \in C$ has label j if the constraint c contains the weight literal (a, j) or $(\neg a, j)$. We use vertex labels low$[i]$ for $i \in \{0 \ldots, k\}$ and up$[j]$ for $j \in \{0 \ldots, k, \infty\}$ to encode the bounds of constraints (we use low[0] and up[∞] in case the lower or upper bound is missing, respectively). Finally we use vertex labels $A, C, \mathcal{R}$ to indicate whether a vertex represents an atom, a clause or a rule, respectively.

Let G denote the incidence graph of the PWC Π with added labels as described above. In the following we will explain how to construct an MSO formula φ such that $G \models \varphi$ iff Π has a stable model. For convenience we will slightly abuse notation and use meta-language terms as shorthands for their obvious definitions in the MSO language; for example we will write $X \subseteq Y$ instead of $\forall x(X(x) \to Y(x))$, and $a \in A$ instead of $V(a) \wedge P_A(a)$.

Let X, Y be set variables and c an individual variable. For each integer $s \in \{0, \ldots, k+1\}$ we define an MSO formula $\mathrm{Sum}_s(X, Y, c)$ that is true for G iff X and Y are interpreted as sets of atoms, c is interpreted as a constraint, and we have

$$s = \sum_{\substack{(a,j) \in Cl(c) \\ a \in X}} j + \sum_{\substack{(\neg a,j) \in Cl(c) \\ a \notin Y}} j.$$

We use the fact that it is always sufficient to choose at most $k+1$ literals from c (say r positive and r' negative literals) to witness that the above equality holds.

$$
\begin{aligned}
&\mathrm{Sum}_s(X, Y, c) \equiv \\
&\quad X, Y \subseteq A \wedge c \in C && (1)\\
&\quad \wedge \bigvee_{1 \le r+r' \le k,\ 1 \le n_1, \ldots, n_{r+r'} \le k+1,\ s = n_1 + \cdots + n_{r+r'}} \exists e_1, \ldots, e_{r+r'} && (2)\\
&\quad\ \ [\, \bigwedge_{i=1}^{r+r'} (P_{n_i}(e_i) \wedge I(c, e_i) \wedge \exists a \in A, I(a, e_i)) && (3)\\
&\quad\ \ \wedge \bigwedge_{1 \le i < i' \le r+r'} e_i \neq e_{i'} && (4)\\
&\quad\ \ \wedge \forall e \in E\, (\neg I(c, e) \vee \forall a \in A, \neg I(a, e) \vee \bigvee_{i=1}^{r+r'} e = e_i) && (5)\\
&\quad\ \ \wedge \bigwedge_{i=1}^{r} (P_+(e_i) \wedge \exists a \in X, I(a, e_i)) && (6)\\
&\quad\ \ \wedge \bigwedge_{i=r+1}^{r'} (P_-(e_i) \wedge \neg\exists a \in Y, I(a, e_i))\ ] && (7)
\end{aligned}
$$

Some further explanation: Each of the $r + r'$ literals is represented by an edge e_i of weight n_i. The disjunction in line (2) runs over all possible combinations of weights $n_1, \ldots, n_{r+r'}$ that give the sum s. Line (3) makes sure that each edge e_i has weight n_i and runs between constraint c and some atom. Lines (4) and (5) make sure that the edges are mutually different and that no other edge runs between constraint c and an atom. Lines (6) and (7) make sure that $e_1, \ldots, e_r$ represent positive literals over atoms that belong to X, and $e_{r+1}, \ldots, e_{r+r'}$ represent negative literals over atoms that do not belong to Y.

The following formula is true if X satisfies c.

$\mathrm{Sat}(X, c) \equiv \mathrm{SatL}(X, X, c) \wedge \mathrm{SatU}(X, X, c)$ where

$\mathrm{SatL}(X, Y, c) \equiv$
$$P_{\mathrm{low}[0]} \vee \bigvee_{i \in \{1,\ldots,k\}} (P_{\mathrm{low}[i]}(c) \wedge \bigvee_{i \le s \le k+1} \mathrm{Sum}_s(X, Y, c)),$$

$\mathrm{SatU}(X, Y, c) \equiv$
$$P_{\mathrm{up}[\infty]} \vee \bigvee_{j \in \{0,\ldots,k\}} (P_{\mathrm{up}[j]}(c) \wedge \bigvee_{0 \le s \le j} \mathrm{Sum}_s(X, Y, c)).$$

The next formula is true if Y is a model of Π.

$\mathrm{Mod}(Y) \equiv \forall r \in \mathcal{R}\ \exists c \in C\ [(H(c, r) \wedge \mathrm{Sat}(Y, c)) \vee (B(c, r) \wedge \neg\mathrm{Sat}(Y, c))]$ where

$H(c, r) \equiv \exists e \in E\ (I(c, e) \wedge I(r, e) \wedge P_h(e))$,

$B(c, r) \equiv \exists e \in E\ (I(c, e) \wedge I(r, e) \wedge P_b(e))$.

Finally, the formula $\mathrm{SMod}(Y)$ is true iff Y is a stable model of Π. We make use of the formula $\mathrm{Red}(X, Y)$ that states that X satisfies the reduct Π^Y.

$\mathrm{SMod}(Y) \equiv \mathrm{Mod}(Y) \wedge \forall X \subseteq Y\ (X = Y \vee \neg\mathrm{Red}(X, Y))$

$\mathrm{Red}(X, Y) \equiv \forall r \in \mathcal{R}\ \forall a \in A\ [a \in X \vee a \notin Y \vee \neg\mathrm{InH}(a, r)$
$\vee \exists c\ \ (B(c, r) \wedge (\neg\mathrm{SatU}(Y, Y, c) \vee \neg\mathrm{SatL}(X, Y, c)))]$ where $\mathrm{InH}(a, r) \equiv \exists c \in C\ \exists e, e' \in E\ \ (I(c, e) \wedge P_h(e) \wedge I(a, e) \wedge P_+(e'))$, that is, a is an atom that occurs as a positive literal in the constraint at the head of rule r.

We summarize the correctness of the construction in the following lemma.

Lemma 4. *Let $\varphi = \exists Y\ \ \mathrm{SMod}(Y)$. Then Π has a stable model if and only if $G \models \varphi$.*

Theorem 3 now follows directly by Courcelle's Theorem.

Dynamic Programming Approach

Recently, Jakl, Pichler, and Woltran (2009) presented a dynamic programming algorithm for answer-set programming that 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 that algorithm of Jakl et al. by methods to handle the cardinality constraints. In principle, this should be feasible. However, computationally, this approach has a certain drawback, namely: the aforementioned algorithm is tractable for bounded treewidth, but it is *double exponential* w.r.t. the treewidth (basically this is due to the handling of disjunctions). Our goal here is to present an algorithm that is only *single exponential* w.r.t. 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.

For this purpose, we need an alternative characterization of stable models. Slightly rephrasing a result by Liu (2009), we can characterize answer sets of PCCs as follows: Given a PCC $\Pi = (A, C, \mathcal{R})$, $M \subseteq A$ is an answer set (stable model) of Π iff 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) $a \in Cl(\mathrm{H}(r))$,
  (R2) $M \models \mathrm{B}(r)$,
  (R3) for each $c \in \mathrm{B}(r)$, $l(c) \le$
  $|\{b \in Cl(c) : b < a\} \cup \{\neg b \in Cl(c) : b \in A \setminus M\}|$.

Since the handling of linear orders is crucial for utilizing the above characterization, we will fix some notation first. We denote by $[a_1, a_2, \ldots, a_n]$ a (strict) linear order $a_1 < a_2 < \ldots < a_n$ on a set $A = \{a_1, \ldots, a_n\}$. Moreover, $[\![A]\!]$ denotes the set of all possible linear orders over A. Two linear orders $[a_1, \ldots, a_n]$ and $[b_1, \ldots, b_m]$ are called *inconsistent*, if there are a_i, a_j, b_k, b_l s.t. $a_i < a_j$, $b_k < b_l$, $a_i = b_l$ and $a_j = b_k$. Otherwise, we call them *consistent*. Given two consistent linear orders $[a_1, \ldots, a_n] \in [\![A]\!]$ and $[b_1, \ldots, b_m] \in [\![B]\!]$, we denote by $[a_1, \ldots, a_n] + [b_1, \ldots, b_m] = S$ the set of their possible *combinations*. S contains those linear orders

$[c_1, \ldots, c_p] \in [\![A \cup B]\!]$ s.t. for every pair $a_i < a_j$ (respectively $b_i < b_j$), there exists $c_k < c_l$ with $c_k = a_i$ and $c_l = a_j$ (respectively $c_k = b_i$ and $c_l = b_j$). Note that in general, there exist more than one possible combinations. Furthermore, we denote by $[a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n] - [a_i]$ the linear order $[a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n]$.

Throughout the whole section, let $\mathcal{T} = (T, \chi)$ be a normalized tree decomposition of a PCC $\Pi = (A, C, \mathcal{R})$. We present a dynamic programming algorithm, traversing in a bottom-up direction through $\mathcal{T}$ 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 become handy. Given a node $n \in T$, we denote by T_n the subtree of T rooted at n. For a set $S \subseteq A \cup C \cup \mathcal{R}$, $n|_S$ is a shorthand for $\chi(n) \cap S$. Moreover, $n{\downarrow_S} := \bigcup_{m \in T_n} m|_S$ and $n{\Downarrow_S} := n{\downarrow_S} \setminus n|_S$. Since the scope of our solution is always limited to a subtree of the whole tree decomposition, the notion of a model has to be refined wrt. $U = n{\downarrow_A}$. To this end, the cardinality of a constraint $c \in C$ with respect to an interpretation $I \subseteq U$ is given by

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

Then I is a model of c under U (denoted by $I \models_U c$) iff $l(c) \leq \Gamma(c, I, U) \leq u(c)$. For rule $r \in \mathcal{R}$ and sets $C \subseteq C$, $R \subseteq \mathcal{R}$, the notions of $C \models r$, $I \models C$ and $I \models R$ are extended to $C \models_U r$, $I \models_U C$ and $I \models_U R$ in the straightforward way. Note that $\models_U$ and $\models$ coincide for $U = A$.

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

Definition 5. *A* partial solution *(for node $n \in T$) is a tuple $\hat{\vartheta} = (n, \hat{M}, \hat{C}, \hat{R}, \hat{L}_<, \hat{\gamma}, \hat{\gamma}_<, \hat{\Delta})$, with interpretation $\hat{M} \subseteq n{\downarrow_A}$, satisfied constraints $\hat{C} \subseteq n{\downarrow_C}$, satisfied rules $\hat{R} \subseteq n{\downarrow_\mathcal{R}}$, linear order $\hat{L}_< \in [\![\hat{M} \cup \hat{C} \cup n{\downarrow_\mathcal{R}}]\!]$, cardinality functions $\hat{\gamma} : n{\downarrow_C} \to \mathbb{N}$ 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 $\hat{\delta}_R \subseteq n{\downarrow_\mathcal{R}}$, 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 $\hat{\sigma} : \hat{\delta}_h \to \{0, 1\}$ s.t. the following conditions are jointly satisfied:*

1. $\hat{C} \cap n{\Downarrow_C} = \{c \in n{\Downarrow_C} \,:\, \hat{M} \models_{n\downarrow_A} c\}$
2. $\hat{R} = \{r \in n{\downarrow_\mathcal{R}} \,:\, \hat{C} \models_{n\downarrow_C} r\}$ *and* $n{\Downarrow_\mathcal{R}} \subseteq \hat{R}$
3. $\hat{\gamma}(c) = \Gamma(c, \hat{M}, n{\downarrow_A})$ *for all* $c \in n{\downarrow_C}$
4. $\hat{\gamma}_<(c) = |\{b \in Cl(c) \,:\, b < c\}| + \left|\{\neg b \in Cl(c) \,:\, b \in n{\downarrow_A} \setminus \hat{M}\}\right|$ *for all* $c \in n{\downarrow_C}$
5. $\hat{\delta}_M = \{a \in \hat{M} \,:\, c \in \hat{\delta}_h, a \in c, a > c\}$ *and* $\hat{M} \cap n{\Downarrow_A} \subseteq \hat{\delta}_M$
6. $\hat{\delta}_b = \bigcup_{r \in \hat{\delta}_R} \text{B}(r)$ *and* $\hat{\delta}_b \subseteq \hat{C}$
7. $c \in \text{B}(r) \Rightarrow r > c$ *for all* $c \in \hat{\delta}_b$ *and* $r \in \hat{\delta}_R$
8. $l(c) \leq \hat{\gamma}_<(c)$ *for all* $c \in \hat{\delta}_b \cap n{\Downarrow_C}$
9. $\hat{\sigma}(c) = 1 \Leftrightarrow \exists r \in \hat{\delta}_R$ *with* $\text{H}(r) = c$ *and* $c > r$
10. $\hat{\sigma}(c) = 1$ *for all* $c \in \hat{\delta}_h \cap n{\Downarrow_C}$

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, $\hat{C} \cap n{\Downarrow_C}$, the constraints in $\hat{C}$ that are no longer visible, indeed contains exactly the constraints that are satisfied under interpretation $\hat{M}$, i.e. $\{c \in n{\Downarrow_C} \,:\, \hat{M} \models_{n\downarrow_A} c\}$, while $\hat{C} \cap n|_C$ 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 the 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 the head, and respectively, body constraints of the rules in $\hat{\delta}_R$. The set $\hat{\delta}_M$ contains those atoms, for which we 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, w.l.o.g., 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 6. *Let n_{root} be the root node of T, and let $\chi(n_{root}) = \emptyset$. Then $\mathcal{AS}(\Pi) \neq \emptyset$ if and only if there exists a partial solution $\hat{\vartheta} = (n_{root}, \hat{M}, \hat{C}, \hat{R}, \hat{L}_<, \hat{\gamma}, \hat{\gamma}_<, \hat{\Delta})$ for n_{root}.*

Proof sketch. Given an answer set $M \in \mathcal{AS}(\Pi)$, it is easy to construct a partial solution $(n_{root}, M, \hat{C}, \hat{R}, \hat{L}_<, \hat{\gamma}, \hat{\gamma}_<, \hat{\Delta})$ fulfilling all the conditions of Definition 5. For the other direction, the requirement that $\chi(n_{root}) = \emptyset$ ensures, that the guessing part of a given partial solution $\hat{\vartheta}$ is nonexistent. One can check that the conditions of Definition 5 suffice to guarantee that $\hat{M} \in \mathcal{AS}(\Pi)$. The most difficult part is the linear order $\hat{L}_<$ ranging not only over the atoms in $\hat{M}$ but also over $\hat{C}$ and all rules in $\mathcal{R}$. The idea is, that whenever an atom b is contained positively in some constraint c_1, which is in the body of some rule r, which in turn has head c_2 containing atom a, then the ordering $[b, c_1, r, c_2, a]$ ensures that $b < a$ and, hence, b is contained in the set on the rhs of the inequality in condition (R3) of stable models. □

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 are similar data structures as partial solutions, but instead of ranging over the whole subtree, their scope is restricted to the current bag. But we are not interested in arbitrary bag assignments, instead we consider those, which can be seen as the projection of a partial solution to the current bag. Formally this is stated as follows:

Definition 7. *A* bag assignment (for node $n \in T$) *is a tuple* $\vartheta = (n, M, C, R, L_<, \gamma, \gamma_<, \Delta)$, *with partial model* $M \subseteq n|_A$, *satisfied constraints* $C \subseteq n|_C$, *satisfied rules* $R \subseteq n|_\mathcal{R}$, *linear order* $L_< \in [\![M \cup C \cup n|_\mathcal{R}]\!]$, *cardinality functions* $\gamma : n|_C \to \mathbb{N}$ *and* $\gamma_< : C \to \mathbb{N}$, *and derivation witness* $\Delta = (\delta_R, \delta_M, \delta_h, \delta_b, \sigma)$ *with derivation rules* $\delta_R \subseteq n|_\mathcal{R}$, *derived atoms* $\delta_M \subseteq M$, *derivation head constraints* $\delta_h \subseteq C$, *derivation body constraints* $\delta_b \subseteq C$, *and check function* $\sigma : \delta_h \to \{0, 1\}$.

Definition 8. *Bag assignment* $\vartheta = (n, M, C, R, L_<, \gamma, \gamma_<, \Delta)$ *for node* $n \in T$, *with* $\Delta = (\delta_R, \delta_M, \delta_h, \delta_b, \sigma)$ *is called* bag model (for node $n \in T$) *if and only if there exists a partial solution* $\hat{\vartheta} = (n, \hat{M}, \hat{C}, \hat{R}, \hat{L}_<, \hat{\gamma}, \hat{\gamma}_<, \hat{\Delta})$, *with* $\hat{\Delta} = (\hat{\delta}_R, \hat{\delta}_M, \hat{\delta}_h, \hat{\delta}_b, \hat{\sigma})$ *s.t.*

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

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

Proposition 9. *Let* n_{root} *be the root node of* T, *and let* $\chi(n_{root}) = \emptyset$. *Then* $\mathcal{AS}(\Pi) \neq \emptyset$ *if and only if* $\vartheta = (n_{root}, \emptyset, \emptyset, \emptyset, [], \emptyset, \emptyset, \Delta)$ *with* $\Delta = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$ *is a bag model for* n_{root}.

Proof. Since $\chi(n_{root}) = \emptyset$, every partial solution for n_{root} is an extension of ϑ according to the conditions of Definition 8. Therefore this statement follows from Proposition 6. □

By the same argument as for the root node, we may assume that $\chi(n) = \emptyset$ for leaf nodes n. Now a dynamic programming algorithm can be achieved, by creating the only possible bag model $\vartheta = (n, \emptyset, \emptyset, \emptyset, [], \emptyset, \emptyset, \Delta)$ with $\Delta = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$ for each leaf n, and then propagating these bag models along the paths to the root node. Thereby the bag models are altered according to rules, which depend only on the bag of the current node. In order to sketch the cornerstones of the DP 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 $\mathcal{T}$. In order to facilitate the discussion below, we define the following sum for constraint $c \in C$, interpretation $I \subseteq U$ over a set of atoms $U \subseteq A$ and linear order $L_< \in [\![I \cup \{c\}]\!]$:

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

Definition 10. *Let* $\vartheta = (n, M, C, R, L_<, \gamma, \gamma_<, \Delta)$ *and* $\vartheta' = (n', M', C', R', L'_<, \gamma', \gamma'_<, \Delta')$ *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* $n, n' \in T$. *We relate* $\vartheta' \prec_\mathcal{T} \vartheta$ *iff* n *has a single child* n' *and the following properties are satisfied, depending on the node type of* n:

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

$$\begin{aligned}\vartheta &= (n, M', C', R' \setminus \{r\}, L'_< - [r], \gamma', \gamma'_<, \Delta), \text{ with} \\ \Delta &= (\delta'_R \setminus \{r\}, \delta'_M, \delta'_h, \delta'_b, \sigma').\end{aligned}$$

(r-RI):

$$\vartheta \in \{(n, M', C', R^*, L^*_<, \gamma', \gamma'_<, \Delta) : L^*_< \in (L'_< + [r])\}, \text{ with}$$

$$R^* = \begin{cases} R' \cup \{r\} & \text{if } C' \models_{n|_C} r \\ R' & \text{otherwise} \end{cases}$$

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

- *"r is used":* $r \in R^*$, $\mathrm{H}(r) \in n|_C \Rightarrow (\mathrm{H}(r) \in \delta'_h \wedge \mathrm{H}(r) > r)$, *for all* $b \in \mathrm{B}(r) \cap n|_C : b \in C' \wedge b < r$, *and*

  $$\Delta = (\delta'_R \cup \{r\}, \delta'_M, \delta'_h, \delta'_b \cup (\mathrm{B}(r) \cap n|_C), \sigma^*), \text{ with}$$

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

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

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

$$\begin{aligned}\vartheta &= (n, M' \setminus \{a\}, C', R', L'_< - [a], \gamma', \gamma'_<, \Delta), \text{ with} \\ \Delta &= (\delta'_R, \delta'_M \setminus \{a\}, \delta'_h, \delta'_b, \sigma').\end{aligned}$$

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

- *"set a to false":*

  $$\vartheta = (n, M', C', R', L'_<, \gamma^*, \gamma^*_<, \Delta'), \text{ with}$$

  $\gamma^*(c) = \gamma'(c) + \Gamma(c, M', n|_A) - \Gamma(c, M', n'|_A)$, *and*
  $\gamma^*_<(c) = \gamma'_<(c) + \Gamma_<(c, M', n|_A, L'_<) - \Gamma_<(c, M', n'|_A, L'_<)$.

- *"set a to true":*

  $$\begin{aligned}\vartheta &\in \{(n, M^* = M' \cup \{a\}, C', R', L^*_<, \gamma^*, \gamma^*_<, \Delta) : \\ &\qquad L^*_< \in (L'_< + [a])\}, \text{ with} \\ \Delta &= (\delta'_R, \delta'_M \cup \delta^*_M, \delta'_h, \delta'_b, \sigma'), \text{ where}\end{aligned}$$

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

  $\gamma^*(c) = \gamma'(c) + \Gamma(c, M^*, n|_A) - \Gamma(c, M', n'|_A)$, *and*
  $\gamma^*_<(c) = \gamma'_<(c) + \Gamma_<(c, M^*, n|_A, L^*_<) - \Gamma_<(c, M', n'|_A, L'_<)$.

(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*

$$\begin{aligned}\vartheta &= (n, M', C' \setminus \{c\}, R', L'_< - [c], \gamma', \gamma'_<, \Delta), \text{ with} \\ \Delta &= (\delta'_R, \delta'_M, \delta'_h \setminus \{c\}, \delta'_b \setminus \{c\}, \sigma').\end{aligned}$$

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

- *"set c to false": $c \notin \text{B}(r)$ for all $r \in \delta'_R$, and*

  $\vartheta = (n, M', C', R' \cup R^*, L'_<, \gamma' \cup \gamma^*, \gamma'_< \cup \gamma^*_<, \Delta')$, *with*

  $R^* = \{r \in n|_{\mathcal{R}} : C' \models_{n|_C} r\}$, $\gamma^* = \{(c, \Gamma(c, M', n|_A)\}$, $\gamma^*_< = \{(c, \Gamma_<(c, M', n|_A, L'_<)\}$.
- *"set c to true": $(c \in \text{B}(r) \Rightarrow r > c) \wedge (c = \text{H}(r) \Rightarrow r < c)$ for all $r \in \delta'_R$, and*

  $$\vartheta \in \{(n, M', C^* = C' \cup \{c\}, R' \cup R^*, L^*_<, \gamma^*, \gamma^*_<, \Delta) : L^*_< \in (L'_< + [c])\},\ \textit{with}$$

  $$\Delta = (\delta'_R, \delta'_M \cup \delta^*_M, \delta'_h \cup \delta^*_h, \delta'_b \cup \delta^*_b, \sigma^*),\ \textit{where}$$

  $R^* = \{r \in n|_{\mathcal{R}} : C^* \models_{n|_C} r\}$, $\gamma^* = \gamma' \cup \{(c, \Gamma(c, M', n|_A)\}$, $\gamma^*_< = \gamma'_< \cup \{(c, \Gamma_<(c, M', n|_A, L'_<)\}$,

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

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

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

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

Definition 11. *Given any three bag assignments $\vartheta = (n, M, C, R, L_<, \gamma, \gamma_<, \Delta)$, $\vartheta' = (n', M', C', R', L'_<, \gamma', \gamma'_<, \Delta')$, and $\vartheta'' = (n'', M'', C'', R'', L''_<, \gamma'', \gamma''_<, \Delta'')$, with derivation witnesses $\Delta = (\delta_R, \delta_M, \delta_h, \delta_b, \sigma)$, $\Delta' = (\delta'_R, \delta'_M, \delta'_h, \delta'_b, \sigma')$, and $\Delta'' = (\delta''_R, \delta''_M, \delta''_h, \delta''_b, \sigma'')$. We relate $(\vartheta', \vartheta'') \prec_{\mathcal{T}} \vartheta$ iff*

- *n has two children n' and n''*
- $M = M' = M''$ $C = C' = C''$
- $R = R' \cup R''$ $L_< = L'_< = L''_<$
- $\gamma(c) = \gamma'(c) + \gamma''(c) - \Gamma(c, M, n|_A)$ *for all* $c \in n|_C$
- $\gamma_<(c) = \gamma'_<(c) + \gamma''_<(c) - \Gamma_<(c, M, n|_A, L_<)$ *for all* $c \in n|_C$
- $\delta_R = \delta'_R = \delta''_R$ $\delta_M = \delta'_M \cup \delta''_M$
- $\delta_h = \delta'_h = \delta''_h$ $\delta_b = \delta'_b \cup \delta''_b$
- $\sigma(c) = \max\{\sigma'(c), \sigma''(c)\}$ *for all* $c \in \delta_h$

Let us look exemplarily at (CR) nodes in more detail. Consider nodes n which remove a constraint c, i.e. $\chi(n) = \chi(n') \setminus \{c\}$, where n' is the child of n (see, for instance, the node with bag $\{p_3, c_3\}$ in the left branch of $\mathcal{T}_{Ex}$ in Figure 1, which is a c_4-removal node). Let $\vartheta' = (n', M', C', R', L'_<, \gamma', \gamma'_<, \Delta')$ with $\Delta' = (\delta'_R, \delta'_M, \delta'_h, \delta'_b, \sigma')$ be a bag model for n'. We then create a bag model for n 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 5. They ensure that all guesses with respect to c turned out be correct. In the case of an affirmative answer, we remove c from all sets of ϑ' in order to create the new bag model $\vartheta = (n, M', C' \setminus \{c\}, R', L'_< - [c], \gamma', \gamma'_<, \Delta)$ with $\Delta = (\delta'_R, \delta'_M, \delta'_h \setminus \{c\}, \delta'_b \setminus \{c\}, \sigma')$.

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

Theorem 12 (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 $n' \in T$ and let ϑ be a bag assignment for node $n \in T$ with $\vartheta' \prec_{\mathcal{T}} \vartheta$. Then n' is the single child of n, with n being of type (RR),(RI),(AR),(AI),(CR), or (CI). Assume n is a (r-RR) node. According to Definition 10, 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 n', satisfying all the conditions of Definition 8.

Claim: $\hat{\vartheta}$ is also a partial solution of n.
To verify this claim, we have to check the conditions of Definition 5. Since $n'{\Downarrow}_C = n{\Downarrow}_C$, $n'{\downarrow}_C = n{\downarrow}_C$, $n'{\Downarrow}_A = n{\Downarrow}_A$, $n'{\downarrow}_A = n{\downarrow}_A$, and $n'{\downarrow}_{\mathcal{R}} = n{\downarrow}_{\mathcal{R}}$, the only non-trivial condition is number 2 where we have to check $n{\Downarrow}_{\mathcal{R}} \subseteq \hat{R}$. Since $r \in R'$ and $R' = \hat{R} \cap n'|_{\mathcal{R}}$, we have $r \in \hat{R}$. Hence, from $n'{\Downarrow}_{\mathcal{R}} \subseteq \hat{R}$ follows that $n{\Downarrow}_{\mathcal{R}} = n'{\Downarrow}_{\mathcal{R}} \cup \{r\} \subseteq \hat{R}$.

Furthermore, the projection of $\hat{\vartheta}$ to the bag $\chi(n)$ 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 $n', n'' \in T$ and let ϑ be a bag assignment for node $n \in T$ with $(\vartheta', \vartheta'') \prec_{\mathcal{T}} \vartheta$. Then n has two children n' and n'' and all the properties of Definition 11 are satisfied. Since ϑ' and ϑ'' are bag models, there exist partial solutions $\hat{\vartheta}'$ of n' and $\hat{\vartheta}''$ of n''. Using these two partial solutions we construct $\hat{\vartheta} = (n, \hat{M}' \cup \hat{M}'', \hat{C}' \cup \hat{C}'', \hat{R}' \cup \hat{R}'', \hat{L}_<, \hat{\gamma}, \hat{\gamma}_<, \hat{\Delta})$ 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 $\hat{L}_< \in (\hat{L}'_< + \hat{L}''_<)$,

$$\hat{\gamma}(c) = \begin{cases} \hat{\gamma}'(c) & c \in n'{\Downarrow}_C, \\ \hat{\gamma}''(c) & c \in n''{\Downarrow}_C, \\ \hat{\gamma}'(c) + \hat{\gamma}''(c) - \Gamma(c, n|_{\hat{M}}, n|_A) & \text{otherwise}, \end{cases}$$

$$\hat{\gamma}_<(c) = \begin{cases} \hat{\gamma}'_<(c) & c \in n'{\Downarrow}_C, \\ \hat{\gamma}''_<(c) & c \in n''{\Downarrow}_C, \\ \hat{\gamma}'_<(c) + \hat{\gamma}''_<(c) - \Gamma_<(c, n|_{\hat{M}}, n|_A, \hat{L}_<) & \text{otherwise}, \end{cases}$$

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

One can now check the conditions of Definition 5 in order to verify that $\hat{\vartheta}$ is a partial solution for n. Furthermore, our construction ensures that the projection of $\hat{\vartheta}$ to the bag $\chi(n)$ is exactly ϑ, which is therefore a bag model. □

Theorem 13 (Completeness)**.** *Given a bag model ϑ for node $n \in T$. Then either n 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 n. Let $n \in T$ be a (r-RR) node with child n', let ϑ be a bag model for n. We have to show that there exists a bag model ϑ' for n' with $\vartheta' \prec_{\mathcal{T}} \vartheta$. Since ϑ is a bag model, there exists a partial solution $\hat{\vartheta}$ of n, satisfying all the conditions of Definition 8. From $r \in n{\Downarrow}_{\mathcal{R}}$ follows, that $r \in \hat{R}$. Now consider the projection of $\hat{\vartheta}$ onto the bag of n'. Then the result is a bag model ϑ' of n' satisfying the conditions of Definition 8 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 12 and Theorem 13 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 9, those bag models are all we need to know. Thus, this dynamic programming algorithm solves the consistency problem.

Theorem 14. *The consistency problem for PCCs* Π *can be solved in time* $O(2^{2w(\log w+4)}k^{4w} \cdot ||\Pi||)$*, with* $w = tw(\Pi)$*,* $k = cw(\Pi)$ *and* $||\Pi||$ *denoting the size of* Π*.*

Proof sketch. The number of different bag models at each node $n \in T$ is bounded by $O(2^{4w}w!k^{2w})$. The number of nodes in our tree decomposition is bounded by $||\Pi||$ and at each node the effort to compute a single bag model is constant with the exception of branch nodes, where one has to compare all possible pairs of bag models of each child node. The given time bound follows from these observations. □

Extensions

In this section, we discuss some extensions of our dynamic programming approach and of Theorem 14.

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

Theorem 15. *The consistency problem for PWCs* Π *with unary weights can be solved in time* $O(2^{2w(\log w+4)}k^{4w} \cdot ||\Pi||)$ *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' s.t. 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 (resp. changed from treewidth ≤ 2 to treewidth 3). □

Reasoning with PCCs and PWCs with unary weights. In the context of non-monotonic reasoning, two kinds of reasoning are usually considered, namely skeptical and credulous reasoning. Recall that an atom a is skeptically (resp. credulously) implied by a program Π if a is true (i.e. contained) in every (resp. 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 16. *Both the skeptical and the credulous reasoning problem for PCCs* Π *can be solved in time* $O(2^{2w(\log w+4)}k^{4w} \cdot ||\Pi||)$ *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 $cr(\vartheta) = \top$ (resp. $sk(\vartheta) = \top$) iff there exists a partial solution $\hat{\vartheta} = (n, \hat{M}, \ldots)$, (resp. iff for all partial solutions $\hat{\vartheta} = (n, \hat{M}, \ldots)$) the atom a is true in $\hat{M}$. Otherwise this flag is set to $\bot$. Then a is credulously (resp. skeptically) implied by Π iff there exists a bag model (resp. iff for all bag models) ϑ of the root node n_{root} of T, we have $cr(\vartheta) = \top$ (resp. $sk(\vartheta) = \top$). Clearly, maintaining the two flags fits within the desired complexity bound. □

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 3). 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 15. 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 s.t. 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 15 immediately yields the desired fixed-parameter linearity result since $f(w) \cdot O(k^{2w})$ is bounded by a constant that is independent of the size of Π.

W[1]-Hardness

In this section we will show that it is unlikely that one can improve the non-uniform polynomial-time result of Theorem 14 to a fixed-parameter tractability result (without bounding the constraint-width as in Theorem 3). We will develop our hardness result within the framework of *parameterized complexity*. Therefore we first outline some of the main concepts of the subject, for an in-depth treatment we refer to other sources (Downey and Fellows 1999; Flum and Grohe 2006; Niedermeier 2006).

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. A parameterized problem is *fixed-parameter tractable* if an instance (x, k) of size n can be solved in time $O(f(k)n^c)$ where f is a computable function and c is a constant independent of k. If $c = 1$ then we speak of linear-time fixed-parameter tractability. FPT denotes the class of all fixed-parameter tractable decision problems. Parameterized complexity theory offers a completeness theory similar to the theory of NP-completeness. An *fpt-reduction* from a parameterized decision problem P to a parameterized decision problem Q is a transformation that maps an instance (x, k) of P of size n to an instance $(x', g(k))$ of Q in time $O(f(k)n^c)$ (f, g are arbitrary computable functions, c is a constant) such that (x, k) is a yes-instance of P if and only if $(x', g(k))$ is a yes-instance of Q. A parameterized complexity class is the class of parameterized decision problems fpt-reducible to a certain parameterized decision problem Q. Of particular interest is the class W[1] which is considered as the parameterized analog to NP. For example, the Clique problem (given a graph G and an integer k, decide if G contains a complete subgraph on k vertices), parameterized by k, is a well-known W[1]-complete problem. It is believed that FPT $\neq$ W[1], and there is strong theoretical evidence that supports this belief, for example, FPT = W[1] would imply that the Exponential Time Hypothesis fails, see (Flum and Grohe 2006).

In the proof of Theorem 17 below we will devise an fpt-reduction from the Minimum Maximum Outdegree problem (or MMO, for short). To state this problem we need to introduce some concepts. A (positive integral) *edge weighting* of a graph $H = (V, E)$ is a mapping w that assigns to each edge of H a positive integer. An *orientation* of H is a mapping $\Lambda : E \to V \times V$ with $\Lambda(uv) \in \{(u, v), (v, u)\}$. The *weighted outdegree* of a vertex $v \in V$ with respect to an edge weighting w and an orientation Λ is defined as

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

An instance of MMO consists of a graph H, an edge weighting w of H, and a positive integer r; the question is whether there exists an orientation Λ of H such that $d^+_{H,w,\Lambda}(v) \leq r$ for each $v \in V$. Szeider (2008b) has shown that the MMO problem with edge weights (and therefore also r) given in unary is W[1]-hard when parameterized by the treewidth of H.

Theorem 17. *The consistency problem for PCCs is* W[1]-*hard when parameterized by treewidth.*

Proof. Let (H, w, r) be an instance of MMO of treewidth t, $H = (V, E)$. We may assume that no edge is of weight larger than r since otherwise we can reject the instance. Let $\prec$ be an arbitrary linear ordering of V. We form a PWC $\Pi = (A, C, \mathcal{R})$ with unary weights as follows: The set A contains an atom $a_{uv} = a_{vu}$ for each edge $uv \in E$; C contains a constraint $c_v = (S_v, 0, r)$ for each vertex $v \in V$ where $S_v = \{ (a_{uv}, w(vu)) : uv \in E,\ v \prec u \} \cup \{ (\neg a_{uv}, w(vu)) : uv \in E,\ u \prec v \}$; $\mathcal{R}$ contains a rule $r_v = (c_v, \emptyset)$ for each $v \in V$.

Claim 1. $tw(\Pi) \leq \max(2, t)$. Let (T, χ) be a tree decomposition of H of width t. We extend (T, χ) to a tree decomposition of Π as follows. For each edge $uv \in E$ we pick a node n_{uv} of T with $u, v \in \chi(n_{uv})$ and for each vertex $v \in V$ we pick a node n_v of T with $v \in \chi(n_v)$ (such nodes exist by the definition of a tree decomposition). We attach to n_{uv} a new neighbor n'_{uv} (of degree 1) and put $\chi(n'_{uv}) = \{u, v, a_{uv}\}$, and we attach to n_v a new neighbor n'_v (of degree 1) and put $\chi(n'_v) = \{v, r_v\}$. It is easy to verify that we obtain this way a tree decomposition of Π of width $\max(t, 2)$, hence the claim follows. We note in passing that in fact we have $tw(\Pi) \leq tw(H)$ since H is a graph minor of the incidence graph of Π.

Claim 2. H has an orientation Λ with $\max_{v \in V} d^+_{H,w,\Lambda}(v) \leq r$ *iff Π has a model*. We associate with an orientation Λ the subset $A_\Lambda = \{ a_{uv} \in A_\Lambda : u \prec v \text{ and } \Lambda(uv) = (u, v) \}$. This gives a natural one-to-one correspondence between orientations of H and subsets of A. We observe that for each $v \in V$, the sum of weights of the literals in constraint c_v satisfied by A_Λ is exactly the weighted outdegree of v with respect to Λ. Hence A_Λ is a model of Π iff $d^+_{H,w,\Lambda}(v) \leq r$ for all $v \in V$.

Claim 3. All models of Π are stable. This claim follows by exactly the same argument as in the proof of Theorem 2.

Π can certainly be obtained from (H, w, r) in polynomial time. We can even encode the weights of literals in unary since we assumed that that the edge weighting w is given in unary. Hence, by Claims 1-3 we have an fpt-reduction from MMO to the consistency problem for PWCs with unary weights. Using the construction as described in the proof of Theorem 15, we can transform Π in polynomial time into a decision-equivalent PCC Π' by increasing the treewidth at most by a small constant. In total we have an fpt-reduction from MMO to the consistency problem for PCCs (both problems parameterized by treewidth). The theorem now follows by the W[1]-hardness of MMO for parameter treewidth. □

Discussion

In this work, we have proved several results for PWCs and PCCs of bounded treewidth without addressing the problem of actually computing a tree decomposition of appropriate width. As has been mentioned earlier, Bodlaender (1996) showed that deciding if a graph has treewidth $\leq w$ and, if this is the case, computing a tree decomposition of width w is fixed-parameter linear w.r.t. the treewidth. Unfortunately, this linear time algorithm is only of theoretical interest and the practical usefulness is limited (Koster, Bodlaender, and van Hoesel 2001). 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 (Koster, Bodlaender, and van Hoesel 2001; Bodlaender and Koster 2006; van den Eijkhof, Bodlaender, and Koster 2007; Bodlaender and Koster 2008). Moreover, in some cases, a tree decomposition of low width may be obtained from a given problem in a "natural way".

In (Szeider 2008a), a meta-theorem for MSO problems on graphs with cardinality and weight constraints was shown. This meta-theorem allows to handle cardinality constraints with respect to sets that occur as free variables in the corresponding MSO formula. It provides a polynomial time algorithm for checking whether a PCC (or a PWC with weights in unary) of bounded treewidth has a model. However, in order to check whether a PCC 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 answer-set programming (ASP) presented by Jakl, Pichler, and Woltran (2009). 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 so-called tree-models. A tree-model consists of a subset of the atoms in a bag (the ones which are true in the models thus represented) and a subset of the rules in a bag (the ones which 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 w.r.t. M which 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 from (Jakl, Pichler, and Woltran 2009) is *double exponential* in the treewidth, since it has to handle *sets of subsets of the bag* at each node. Hence, 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 *single exponential* in the treewidth.

Conclusion

In this paper 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.

For future work, we plan 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 KR & R, e.g., to express costs or probabilities. Of particular interest would be extensions of Courcelle's Theorem such as presented by Szeider (2008a). We thus aim at a meta-theorem that yields an alternative proof of our non-uniform polynomial-time tractability result for PCCs and that can be used as a tool in proving further tractability results of this kind in the KR & R domain.

The upper bounds on the time complexity of our dynamic programming algorithms were obtained by very coarse estimates (see Theorems 14, 15, 16). In particular, we assumed straightforward methods for storing and manipulating bag assignments. For an actual implementation of our algorithm, more sophisticated methods and data structures have to be developed. This should also lead to a further improvement of the upper bounds on the time complexity.

References

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

Bodlaender, H. L., and Koster, A. M. C. A. 2008. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.* 51(3):255–269.

Bodlaender, H. L. 1996. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25(6):1305–1317.

Courcelle, B. 1987. Recognizability and second-order definability for sets of finite graphs. Technical Report I-8634, Université de Bordeaux.

Downey, R. G., and Fellows, M. R. 1999. *Parameterized Complexity*. Springer Verlag.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*. Springer Verlag.

Gottlob, G.; Pichler, R.; and Wei, F. 2006. Bounded treewidth as a key to tractability of knowledge representation and reasoning. In *Proc. AAAI'06*, 250–256.

Jakl, M.; Pichler, R.; and Woltran, S. 2009. Answer-set programming with bounded treewidth. In *Proc. IJCAI'09*, 816–822.

Kloks, T. 1994. *Treewidth, Computations and Approximations*. Springer Verlag.

Koster, A. M. C. A.; Bodlaender, H. L.; and van Hoesel, S. P. M. 2001. Treewidth: Computational experiments. *Electronic Notes in Discrete Mathematics* 8:54–57.

Liu, G. 2009. Level mapping induced loop formulas for weight constraint and aggregate programs. In *Proc. LPNMR'09*, volume 5753 of *LNCS*, 444–449. Springer.

Marek, V. W., and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: A 25-Year Perspective*, 375–398. Springer.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press.

Niemelä, I.; Simons, P.; and Soininen, T. 1999. Stable model semantics of weight constraint rules. In *Proc. LPNMR'99*, volume 1730 of *LNCS*, 317–331. Springer.

Szeider, S. 2008a. Monadic second order logic on graphs with local cardinality constraints. In *Proc. MFCS'08*, volume 5162 of *LNCS*, 601–612. Springer.

Szeider, S. 2008b. Not so easy problems for tree decomposable graphs (invited talk). In *Proc. ICDM'08, 2nd International Conference on Discrete Mathematics*, 161–171.

van den Eijkhof, F.; Bodlaender, H. L.; and Koster, A. M. C. A. 2007. Safe reduction rules for weighted treewidth. *Algorithmica* 47(2):139–158.