Bounded Treewidth as a Key to Tractability of Knowledge Representation and Reasoning

Georg Gottlob and **Reinhard Pichler** and **Fang Wei**
Database and Artificial Intelligence Group
Technische Universität Wien

Abstract

Several forms of reasoning in AI – like abduction, closed world reasoning, circumscription, and disjunctive logic programming – are well known to be intractable. In fact, many of the relevant problems are on the second or third level of the polynomial hierarchy. In this paper, we show how the powerful notion of treewidth can be fruitfully applied to this area. In particular, we show that all these problems become tractable (actually, even solvable in linear time), if the treewidth of the involved formulae (or of the disjunctive logic programs, resp.) is bounded by some constant. Experiments with a prototype implementation prove the feasibility of this new approach, in principle, and also give us hints for necessary improvements.

In many areas of computer science, bounded treewidth has been shown to be a realistic and practically relevant restriction. We thus argue that bounded treewidth is a key factor in the development of efficient algorithms also in knowledge representation and reasoning – despite the high worst case complexity of the problems of interest.

Introduction

In the nineteen-nineties, several forms of reasoning in AI - like abduction, closed world reasoning, circumscription, and disjunctive logic programming - were shown to be highly intractable, In fact, many relevant problems in this area are on the second or even third level of the polynomial hierarchy, see (Eiter & Gottlob 1993), (Eiter & Gottlob 1995a), (Eiter & Gottlob 1995b).

In recent years, an interesting approach to dealing with intractability has evolved, namely parameterized complexity, see (Downey & Fellows 1999). It has turned out that many hard problems become tractable if some problem parameter is fixed or bounded by a fixed constant. Such problems are called *fixed-parameter tractable* (FPT, for short). In the arena of graph problems, an important parameter thus investigated is the so-called *treewidth* of a graph G – which is a measure of the "tree-likeness" of G. If the treewidth of the graphs under consideration is bounded by a fixed constant, then many otherwise intractable problems become tractable, e.g. 3-colorability, Hamiltonicity, etc. It is generally believed that many practically relevant problems actually do have low treewidth, see e.g. the discussion of applications in (Bodlaender 1993). Moreover, in (Downey & Fellows 1999) it is noticed that many tractability results based on other interesting parameters of graphs (like bandwidth, cutwidth, radius of planar graphs, etc.) are in fact subsumed by the more general concept of treewidth. Treewidth has also been fruitfully applied to some areas of AI, notably to constraint satisfaction, see (Arnborg 1985).

A deep result and mathematical tool for deriving new FPT-results is Courcelle's famous theorem, see (Courcelle 1990), which states that graph properties expressible by Monadic Second Order (MSO, for short) sentences are tractable (actually, even decidable in linear time) if the treewidth of the graphs is bounded by a fixed constant.

In this paper, we revisit several intractable problems in AI. Our goal is to harness the powerful machinery of Courcelle's Theorem in the area of knowledge representation and reasoning (KR & R, for short). Building upon the work of (Szeider 2004) and (Gottlob, Scarcello, & Sideri 2002), we first have to introduce the notion of treewidth to reasoning problems. Then we show that virtually all relevant decision problems in the area of abduction, closed world reasoning, circumscription, and disjunctive logic programming become tractable if the treewidth of the involved formulae (or of the disjunctive logic programs, resp.) is bounded by some constant. The central idea for deriving these FPT-results is to encode the decision problems in terms of MSO sentences.

As usual, the benefit of FPT-results is twofold: First, they give a better understanding of the computational nature and of the real source of complexity of the problems under consideration. Second, we believe that the FPT-results shown here open the grounds for the development of smart parameterized algorithms for these problems. In this spirit, we propose a new approach to building a general solver for the KR & R problems studied here. Moreover, we have constructed a prototype implementation. Experiments with this prototype clearly show the feasibility of our new approach in principle. Moreover, they also show directions for necessary improvements of this first implementation.

The rest of the paper is organized as follows. After recalling some basic definitions and results, we prove the fixed-parameter tractability of many relevant decision problems

arising in disjunctive logic programming, closed world reasoning, circumscription, and abduction, respectively. Finally, we report on a prototype implementation and draw some conclusions for future work.

Preliminaries

A *tree decomposition* $\mathcal{T}$ of a graph $G = (V, E)$ is a pair (T, λ), where T is a tree and λ is a labeling function with $\lambda(N) \subseteq$V for every node $N \in T$, s.t. the following conditions hold:

1. $\forall v \in V$, there exists a node N in T, s.t. $v \in \lambda(N)$.
2. $\forall e \in E$, there exists a node N in T, s.t. $e \subseteq \lambda(N)$.
3. $\forall v \in V$, the set of nodes $\{N \mid v \in \lambda(N)\}$ induces a connected subtree of T.

The *width* of a tree decomposition (T, λ) is defined as $\max(\{|\lambda(N)| - 1 : N \text{ node in } T\})$. The *treewidth* $tw(G)$ is the minimum width over all tree decompositions of G.

The notions of tree decomposition and treewidth can be naturally generalized to arbitrary (finite) relational structures: The set U of values in the active domain corresponds to the vertex set V and condition 2 above has to be replaced in the sense that for all tuples $(a_1, \ldots, a_n)$ in the database, there exists a node N in $\mathcal{T}$, s.t. $\{a_1, \ldots, a_n\} \subseteq \lambda(N)$.

The *Monadic Second Order* (MSO, for short) formulae on graphs considered here are made up of the logical connectives $\vee$, $\wedge$, and $\neg$, variables (for vertices and vertex sets), the quantifiers $\exists$ and $\forall$ and the binary relations $x \in Y$, $e(x, y)$, and equality. It is common practice to denote vertex variables by lower-case letters and vertex set variables by upper-case letters. Moreover, it is convenient to use symbols like $\subseteq$, $\subset$, $\cap$, $\cup$, and $\rightarrow$ with the obvious meaning as abbreviations. In case of MSO-formulae over arbitrary relational structures, all relation symbols from the database (i.e., the "extensional DB-predicates") may also be used. The importance of MSO formulae in the context of parameterized complexity comes from the following result:

Theorem 1 (Courcelle 1990) *Let φ be a fixed MSO-sentence and let k be a fixed constant. Deciding whether φ holds for an input graph G (more generally, for an input structure A) can be done in linear time if the treewidth of the graphs (resp. of the structures) under consideration is bounded by k.*

In general, we simply say that graphs (resp. structures) "have bounded treewidth" without explicitly mentioning k.

Note that the fixed-parameter linearity according to Theorem 1 only applies to the *data complexity*, i.e. the formula φ is fixed. There is no such FPT-result, if we consider the *combined complexity* instead (i.e. also φ is part of the input). We shall come back to this point in the discussion of our prototype implementation.

A *propositional formula* F is built up from propositional variables (denoted as $Var(F)$) and the logical connectives $\vee$, $\wedge$, and $\neg$. An *interpretation* of F is simply a subset X of $Var(F)$, i.e. the variables in X evaluate to true, while all other variables evaluate to false. If F evaluates to true in X, then X is called a *model* of F, written as $X \models F$. Likewise, we write $F_1 \models F_2$ if every model of F_1 is also a model of F_2. Moreover, X is called a *minimal model*, if there exists no model X' of F with $X' \subset X$. The following result is folklore, see e.g. (Baaz, Egly, & Leitsch 2001).

Theorem 2 *For every propositional formula F, there exists a formula F' in Conjunctive Normal Form (CNF), with $Var(F) \subseteq Var(F')$, s.t. the following properties hold:*

1. *$\forall X$ with $X \models F$, there exists an interpretation Y with $X \subseteq Y$ and $(Y \setminus X) \subseteq (Var(F') \setminus Var(F))$, s.t. $Y \models F'$.*
2. *$\forall Y$ with $Y \models F'$, the interpretation $Y \cap Var(F)$ is a model of F.*

Moreover, such an F' can always be found in linear time and, therefore, also the size of F' is linearly bounded by the size of F.

Proof. The negation normal form and a parse tree thereof can be clearly obtained in linear time. The CNF can then be constructed by a bottom-up traversal of the parse tree and by successive applications of the rewrite rule $(A \wedge B) \vee C \Rightarrow (z \vee A) \wedge (z \vee B) \wedge (\neg z \vee C)$ for some fresh variable z. Again this is feasible in linear time. □.

By slight abuse of notation, we shall refer to such an F' in CNF as the *canonical CNF* of F, even though it is not unique. But, of course, it can be easily made unique by fixing the order in which the above rewrite rule has to be applied to subformulae of F.

For a propositional formula F in CNF, there are several possibilities to define a corresponding graph. The most powerful concept (cf. the discussion in (Szeider 2004)) is the *incidence graph* $I(F)$, which contains as vertices the clauses and propositional variables of F; two vertices c and x (corresponding to a clause c and a variable x) are connected in $I(F)$, iff x occurs (either negated or unnegated) in c. More generally, we can represent F by a relational structure $A(F)$ based on the extensional DB predicates $cl(.)$, $var(.)$, $Pos(.,.)$, $Neg(.,.)$ with the following intended meaning: $cl(c)$ (resp. $var(x)$) means that c is a clause (resp. a variable) in F; $Pos(x, c)$ (resp. $Neg(x, c)$) means that x occurs unnegated (resp. negated) in the clause c. Then we define the treewidth of F as $tw(F) = tw(I(F)) = tw(A(F))$. For an arbitrary propositional formula F, we set $tw(F) = tw(F')$, where F' is the canonical CNF of F.

Finally, the notion of the incidence graph I, the relational structure A and the treewidth tw can be naturally extended to more than one formula in CNF, e.g., let F_1 and F_2 be two propositional formulae in CNF. Then the incidence graph of (F_1, F_2) is simply $I(F_1 \wedge F_2)$. Again, (F_1, F_2) can be represented by a relational structure $A(F_1, F_2)$, where $A(F_1, F_2)$ has the extensional DB predicates $cl_i(.)$, $var_i(.)$, $Pos_i(.,.)$, $Neg_i(.,.)$ for $i \in \{1, 2\}$ with the obvious meaning. For the treewidth, we clearly get $tw(A(F_1, F_2)) = tw(F_1 \wedge F_2)$.

From results in (Szeider 2004), the following relationship between CNF-formulae and MSO can be easily derived:

Theorem 3 *Let F be in CNF and X an interpretation, then $X \models F$ holds, iff the following MSO-sentence is valid:*

MSO encoding of $X \models F$ (with F in CNF)

$(\forall c) cl(c) \rightarrow (\exists z)[(Pos(z, c) \wedge z \in X) \vee (Neg(z, c) \wedge z \notin X)]$

Actually, even if F is not in CNF, the property "$X \models F$" can be encoded in terms of MSO:

Theorem 4 *Let F be an arbitrary propositional formula with canonical CNF F' and let X be an interpretation of F. Then "$X \models F$" can be expressed by means of an MSO-sentence.*

Proof. Let F and F' be defined by the extensional DB predicates $var(.)$ (for the variables in F) plus $cl'(.)$, $var'(.)$, $Pos'(.,.)$, and $Neg'(.,.)$ encoding F'. Then we have:

MSO encoding of $X \models F$ (with arbitrary F)

$Ext_F(X, X') \equiv X \subseteq X' \wedge$
$\quad (\forall z)[(z \in X' \wedge z \notin X) \rightarrow (var'(z) \wedge \neg var(z))]$
$X \models F \equiv (\exists X')[Ext_F(X, X') \wedge (X' \models F')]$

The auxiliary predicate $Ext_F(X, X')$ means that X' is an extension of the interpretation X to the variables in F'. Moreover, the subformula $(X' \models F')$ is precisely the CNF-evaluation from Theorem 3. □

Fixed-Parameter Linearity of DLPs

A *disjunctive logic program* (DLP, for short) P is a set of DLP clauses $a_1 \vee \ldots \vee a_n \leftarrow b_1, \ldots, b_k, \neg b_{k+1}, \ldots, \neg b_m$. Let I be an interpretation. Then the *Gelfond-Lifschitz reduct* P^I of P w.r.t. I contains precisely the clauses $a_1 \vee \ldots \vee a_n \leftarrow b_1, \ldots, b_k$, s.t. for all $i \in \{k+1, \ldots, m\}$, $b_i \notin I$. An interpretation I is called a *disjunctive stable model* (DSM, for short), iff I is a minimal model of P^I, see (Gelfond & Lifschitz 1988) and (Przymusinski 1991).

Without any restrictions, the following problems are all on the second level of the polynomial hierarchy, see (Eiter & Gottlob 1995a):

- CONSISTENCY: Does P have a DSM?
- BRAVE REASONING: Is a propositional formula F true in at least one DSM of P (written as $P \models_b F$)?
- CAUTIOUS REASONING: Is a propositional formula F true in all DSMs of P (written as $P \models_c F$)?

In contrast, for bounded treewidth of P and F, the situation changes dramatically. Suppose that a DLP P is given by a database with the extensional DB predicates $var_P(.)$ and $cl_P(.)$ encoding the variables and clauses of P plus the additional predicates $H(.,.)$, $B^+(.,.)$, and $B^-(.,.)$, s.t. $H(x, c)$ means that x occurs in the head of c and $B^+(x, c)$ (resp. $B^-(x, c)$) means that x occurs unnegated (resp. negated) in the body of c. Then we have:

Theorem 5 *The* CONSISTENCY *problem, the* BRAVE REASONING *problem and the* CAUTIOUS REASONING *problem of DLPs can be expressed by means of MSO sentences.*

Proof. Recall from Theorem 4 that $X \models F$ can be encoded by an MSO-formula. In total, we thus have:

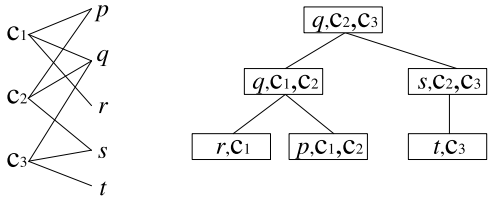

Figure 1: Incidence Graph and Tree Decomp. of Example 6

MSO Encoding of DLP-Reasoning

$GL(X, Y) \equiv (\forall c) cl_P(c) \rightarrow (\exists z)[(H(z, c) \wedge z \in X) \vee$
$\quad (B^+(z, c) \wedge z \notin X) \vee (B^-(z, c) \wedge z \in Y)]$
$DSM(X) \equiv GL(X, X) \wedge (\forall Z)[Z \subset X \rightarrow \neg GL(Z, X)]$

CONSISTENCY: $(\exists X) DSM(X)$

BRAVE REASONING: $(\exists X)[DSM(X) \wedge X \models F]$

CAUTIOUS REASONING: $(\forall X)[DSM(X) \rightarrow X \models F]$

The predicates defined above have the following meaning:

$GL(X, Y)$ = "X is a model of the Gelfond-Lifschitz reduct of the program P w.r.t. the interpretation Y".
$DSM(X)$ = "X is a disjunctive stable model of P". □

Example 6 Consider the following DLP:
$P = \ c_1 : p \vee q \leftarrow \neg r, \ \ c_2 : q \leftarrow \neg p \wedge \neg s, \ \ c_3 : s \vee t \leftarrow q.$
Clearly, P is consistent since, for instance, $\{p\}$ is a DSM.

The incidence graph $I(P)$ of P and a tree decomposition $\mathcal{T}$ of $I(P)$ are given in Figure 1. Note that $\mathcal{T}$ has width 2 (i.e., the maximum width of the labels of $\mathcal{T}$ minus 1). Actually, $I(P)$ cannot have a tree decomposition of width 1 since only trees have $tw = 1$. Hence, we have $tw(P) = 2$.

The structure $\mathcal{A}(P)$ is given by the following set of ground atoms:

$\mathcal{A}(P) = \{var_P(p), var_P(q), var_P(r), var_P(s), var_P(t),$
$\quad cl_P(c_1), cl_P(c_2), cl_P(c_2),$
$\quad H(p, c_1), H(q, c_1), H(q, c_2), H(s, c_3), H(t, c_3),$
$\quad B^+(r, c_1), B^-(p, c_2), B^-(s, c_2), B^-(q, c_3)\}.$

It can be easily checked that the tree decomposition $\mathcal{T}$ in Figure 1 is also a tree decomposition of $\mathcal{A}(P)$; in fact, every tuple in $\mathcal{A}(P)$ is covered by the label of some node in $\mathcal{T}$.

The MSO-formula $GL(X, Y)$ from the proof of Theorem 5 clearly evaluates to true over $\mathcal{A}(P)$ for $X = \{p\}$ and $Y = \{p\}$. Moreover, for $X = \{\}$ and $Y = \{p\}$, it evaluates to false. Hence, $DSM(X)$ evaluates to true for $X = \{p\}$ and, therefore, the consistency of P is correctly established via the MSO-formula $(\exists X) DSM(X)$.

Theorem 7 *The* CONSISTENCY *problem of DLPs P is solvable in linear time, if the incidence graph of P has bounded treewidth. Likewise,* BRAVE REASONING $P \models_b F$ *and* CAUTIOUS REASONING $P \models_c F$ *are solvable in linear time, if the incidence graph of (P, F') has bounded treewidth, where F' denotes the canonical CNF of F.*

Proof. The treewidth of an input DB with the extensional predicates $var_P(.)$, $cl_P(.)$, $H(.,.)$, $B^+(.,.)$, and $B^-(.,.)$ encoding P is identical to the treewidth of the incidence graph $I(P)$. Likewise, if the input DB additionally contains the predicate $var(.)$ for the variables of F and the predicates

$cl'(.)$, $var'(.)$, $Pos'(.,.)$, and $Neg'(.,.)$ encoding F', then the treewidth coincides with the treewidth of the incidence graph $I(P, F')$. The rest follows immediately from Courcelle's Theorem. □

Fixed-Parameter Linearity of CWR and Circumscription

Several forms of *closed world reasoning* (CWR, for short) are proposed in the literature, namely CWA (Closed World Assumption), GCWA (Generalized CWA), EGCWA (Extended GCWA), CCWA (Careful CWA), and ECWA (Extended CWA). They are defined in terms of the following terminology: Let T (a "theory") and F be propositional formulae and let $\langle P; Q; Z\rangle$ be a partition of $Var(T)$. Then we write $M(T)$ (resp. $MM(T)$) to denote the set of all models (resp. of all minimal models) of T. Moreover, we write $MM(T; P; Q; Z)$ to denote the set of $\langle P; Q; Z\rangle$-minimal models of T, i.e.: $X \in MM(T; P; Q; Z)$, iff $X \models T$ and there exists no model Y of T with $(Y \cap P) \subset (X \cap P)$ and $(Y \cap Q) = (X \cap Q)$.

In (Cadoli & Lenzerini 1990), several equivalent characterizations of the closure of a theory T under the various CWR rules are provided. Below, we recall those characterizations which are best suited for our purposes here:

- CWA(T) = $T \cup \{\neg K \mid K$ positive literal s.t. $T \not\models K\}$
- GCWA(T) = $T \cup \{\neg K \mid K$ positive literal and $\forall X \in MM(T)$: $X \not\models K\}$
- EGCWA(T) $\models F$ iff $\forall X \in MM(T)$: $X \models F$
- CCWA($T; P; Q; Z$) = $T \cup \{\neg K \mid K$ positive literal and $\forall X \in MM(T; P; Q; Z)$: $X \not\models K\}$
- ECWA($T; P; Q; Z$) $\models F$ iff $\forall X \in MM(T; P; Q; Z)$: $X \models F$

The DEDUCTION problem of CWR-rule C with C $\in$ { CWA, GCWA, EGCWA, CCWA, ECWA } is as follows: Given T and F (and possibly P, Q, Z), does C(T) $\models F$ (resp. C($T; P; Q; Z$) $\models F$) hold? In (Eiter & Gottlob 1993), this problem is shown to be Π_2^p-complete or even harder for all rules C $\neq$ CWA. Note that in the propositional case, CIRCUMSCRIPTION coincides with the ECWA-rule, see (Gelfond, Przymusinska, & Przymusinski 1989).

Again, for bounded treewidth, we get much better complexity results. Let T and F be arbitrary propositional formulae with canonical CNFs T' and F', respectively. Suppose that the input DB contains the extensional predicates $var_T(.)$ and $var_F(.)$ for the variables in the original formulae T and F. Moreover, the CNFs are encoded as usual by means of the predicates $var_{T'}(.)$, $var_{F'}(.)$, $cl_{T'}(.)$, $cl_{F'}(.)$, etc. Finally, the partition $\langle P; Q; Z\rangle$ of $Var(T)$ is encoded in the input DB via unary predicates P, Q, Z. Then we have:

Theorem 8 *For all of the CWR-rules CWA, GCWA, EGCWA, CCWA, ECWA, the* DEDUCTION *problem (and, hence, also* CIRCUMSCRIPTION*) can be expressed by means of MSO sentences.*

Proof. The rules GCWA and EGCWA are special cases of CCWA and ECWA, resp., with $Q = Z = \emptyset$. Moreover, as mentioned above, CIRCUMSCRIPTION is equivalent to the ECWA-rule. The remaining cases are encoded as follows:

MSO Encoding of CWR-Deduction

$MM(X) \equiv X \models T \wedge \neg(\exists Y)[(Y \cap P) \subset (X \cap P) \wedge (Y \cap Q) = (X \cap Q) \wedge Y \models T]$
$var(z) \equiv var_T(z) \vee var_F(z)$
$clo_1(\neg z) \equiv var(z) \wedge \neg(T \models z)$
$clo_2(\neg z) \equiv var(z) \wedge (\forall Y)[MM(Y) \rightarrow \neg(Y \models z)]$

DEDUCTION with CWA-rule and CCWA-rule:
$(\forall X)[(X \models T \wedge (\forall z)(clo_i(\neg z) \rightarrow X \models \neg z)) \rightarrow (X \models F)]$
DEDUCTION with ECWA-rule:
$(\forall X)[MM(X) \rightarrow (X \models F)]$

The above predicates have the following meaning:

$clo_i(\neg z)$ with $i \in \{1, 2\}$ means that z is in the closure of T w.r.t. the CWA-rule (for $i = 1$) or CCWA-rule (for $i = 2$), respectively. $MM(X)$ means $X \in MM(T; P; Q; Z)$. □

Analogously to the previous section, we thus get the following FPT-results:

Theorem 9 *Consider propositional formulae T and F with canonical CNFs T' and F'. Then for all of the CWR-rules $C \in$ { CWA, GCWA, EGCWA, CCWA, ECWA }, the corresponding* DEDUCTION *problem $T \models_C F$ (and, hence, also* CIRCUMSCRIPTION*) is solvable in linear time, if the incidence graph of (T', F') has bounded treewidth.*

Fixed-Parameter Linearity of Abduction

A *propositional abduction problem* (PAP, for short) is given by a tuple $\mathcal{P} = \langle V, H, M, T\rangle$ where V is a set of propositional variables, $H \subseteq V$ (the "hypotheses"), $M \subseteq V$ (the "manifestations"), and T is a consistent propositional formula (the "theory"). A *solution* of $\mathcal{P}$ is a subset $S \subseteq H$, s.t. $T \cup S$ is consistent and $T \cup S \models M$. Given a PAP $\mathcal{P}$, the basic problems of propositional abduction are the following:

- SOLVABILITY: Does there exist a solution of $\mathcal{P}$?
- RELEVANCE: Given $h \in H$, is h contained in at least one solution of $\mathcal{P}$?
- NECESSITY: Given $h \in H$, is h contained in every solution of $\mathcal{P}$?

In (Eiter & Gottlob 1995b), the former two problems were shown to be Σ_2^p-complete while the latter is Π_2^p-complete. However, for bounded treewidth of T, we can easily establish the fixed-parameter tractability: Let the input DB contain the usual predicates $var_T(.)$, $var_{T'}(.)$, $cl_{T'}(.)$, etc. encoding T and its canonical CNF T'. Moreover, the DB contains the unary predicates V, H, and M encoding the corresponding variable sets. Then we have:

Theorem 10 *The basic PAP-problems* SOLVABILITY, RELEVANCE, *and* NECESSITY *can be expressed by means of MSO sentences.*

Proof. The MSO encoding is straightforward:

MSO Encoding of the Basic Abduction Problems

$Sol(S) \equiv S \subseteq H \wedge (\exists X)[X \models T \wedge X \subseteq S] \wedge (\forall Y)[(Y \models T \wedge Y \subseteq S) \rightarrow Y \models M]$

SOLVABILITY: $(\exists S)Sol(S)$

RELEVANCE: $(\exists S)[Sol(S) \wedge h \in S]$

NECESSITY: $(\forall S)[Sol(S) \rightarrow h \in S]$

The predicate $Sol(S)$ is a straightforward encoding of the property "S is a solution of $\mathcal{P}$", namely (i) S is a subset of the propositional variables in H, (ii) $(T \cup S)$ has at least one model X, and (iii) every model Y of $(T \cup S)$ is also a model of M. □

Usually, a refined version of the RELEVANCE (resp. NECESSITY) problem is considered. Rather than asking whether h is contained in some (resp. every) solution, one is interested if h is contained in some (resp. every) *acceptable* solution. In this context, "acceptable" means "minimal" w.r.t. some preorder $\preceq$ on the powerset 2^H. Consequently, one speaks of $\preceq$-RELEVANCE (resp. $\preceq$-NECESSITY). The above treated basic abduction problems RELEVANCE and NECESSITY correspond to the special case where $\preceq$ is equality. The other preorders thus studied are the following:

- subset-minimality "$\subseteq$"
- prioritization "$\subseteq_P$" for a fixed number p of priorities: H is partitioned into "priorities" $H_1, \ldots, H_p$. Then $A \subseteq_P B$, iff $A = B$ or there exists a k s.t. $A \cap H_i = B \cap H_i$ for all $i < k$ and $A \cap H_k \subset B \cap H_k$.
- minimum cardinality "$\leq$": $A \leq B$ iff $|A| \leq |B|$.
- penalization "$\sqsubseteq_p$" (also referred to as "weighted abduction"): To each element $h \in H$, a weight $w(h)$ is attached. Then $A \leq B$ iff $\sum_{h \in A} w(h) \leq \sum_{h \in B} w(h)$.

In all of these cases, the resulting $\preceq$-RELEVANCE (resp. $\preceq$-NECESSITY) problem is on the second or third level of the polynomial hierarchy, see (Eiter & Gottlob 1995b). The last two cases turn out to be a bit tricky. Below, we establish the desired FPT-result for the first two ones:

Theorem 11 *For $\preceq \in \{\subseteq, \subseteq_P\}$, both the $\preceq$-RELEVANCE problem and the $\preceq$-NECESSITY problem can be expressed by means of MSO sentences.*

Proof. It suffices to provide an MSO encoding of the predicates $Acc_{\subseteq}(S)$ and $Acc_{\subseteq_P}(S)$, which mean that S is an acceptable solution for the preorders $\subseteq$ and $\subseteq_P$, resp.

MSO Encoding of Acceptability w.r.t. $\subseteq$ and $\subseteq_P$

$Acc_{\subseteq}(S) \equiv Sol(S) \wedge (\forall X)[(X \subset S) \rightarrow \neg Sol(X)]$

$X \subset_P S \equiv (X \cap H_1 \subset S) \vee (X \cap H_1 \subseteq S \wedge X \cap H_2 \subset S) \vee (X \cap H_1 \subseteq S \wedge X \cap H_2 \subseteq S \wedge X \cap H_3 \subset S)$

$Acc_{\subseteq_P}(S) \equiv Sol(S) \wedge (\forall X)[(X \subset_P S) \rightarrow \neg Sol(X)]$

By $X \subset_P S$ we mean that $X \subseteq_P S$ and $X \neq S$. Note that we are only considering 3 priority levels H_1, H_2, and H_3 above. However, the generalization to an arbitrary but fixed number p of priority levels is clear. □

By Courcelle's Theorem and Theorems 10 and 11 above, we immediately get the following FPT-result:

Theorem 12 *Consider PAP-problems $\langle V, H, M, T \rangle$ where T is a propositional formula with canonical CNF T'. Then all of the following problems are solvable in linear time, if the incidence graph of T' has bounded treewidth:* SOLVABILITY, RELEVANCE, *and* NECESSITY *as well as* $\preceq$-RELEVANCE *and* $\preceq$-NECESSITY *with* $\preceq \in \{\subseteq, \subseteq_P\}$.

It remains to consider the cases $\preceq \in \{\leq, \sqsubseteq_p\}$. Actually, $\leq$ is a special case of $\sqsubseteq_p$, where every $h \in H$ is assigned the same weight. Unfortunately, MSO is not powerful enough to express the cardinality-comparison $\leq$ (cf. the discussion in (Arnborg, Lagergren, & Seese 1991)). Nevertheless, also for $\preceq \in \{\leq, \sqsubseteq_p\}$, it is possible to establish the FPT-property in case bounded treewidth. Due to space limitations, we can only give a very rough sketch here:

In (Arnborg, Lagergren, & Seese 1991), so-called *extended monadic second order* problems (EMS, for short) are defined, which – in addition to MSO – are allowed to use weight functions, sums and minimum/maximum-functions. This additional expressive power can be shown to suffice for expressing the acceptability of PAP-solutions w.r.t. $\leq$ and $\sqsubseteq_p$. Moreover, in (Arnborg, Lagergren, & Seese 1991), Theorem 5.4, it is shown that deciding EMS problems is fixed-parameter tractable (though not necessarily solvable in linear time) if the treewidth of the graphs considered is bounded by some fixed constant.

Use and Implementation: Proof of Concept

System Description.

Practical work realizing Courcelle's theorem was first proposed in (Arnborg, Lagergren, & Seese 1991), which was further developed in (Flum, Frick, & Grohe 2002). Our implementation, which is based on (Flum, Frick, & Grohe 2002), has the following overall structure:

General Program for KR & R problems
Input: MSO-formula φ, treewidth w, structure $\mathcal{A}$
Output: if $tw(\mathcal{A}) \leq w$ then output "Yes" / "No"
else output "$tw(\mathcal{A}) > w$".

Transformation of the MSO-formula φ.
I. from φ and w, compute the MSO-formula φ^*;

Computation of a colored, binary tree $\mathcal{T}^*$.
II. compute a tree decomposition $\mathcal{T}$ of $\mathcal{A}$ with width w;
if $tw(\mathcal{A}) > w$ then HALT with result "$tw(\mathcal{A}) > w$";
III. from $\mathcal{A}$, compute a colored, binary tree $\mathcal{T}^*$;

MSO-model checking.
IV. check if φ^* evaluates to true over $\mathcal{T}^*$;

The problem of evaluating an MSO formula φ over relational structures $\mathcal{A}$ is transformed into an equivalent problem of evaluating an MSO formula φ^* over colored, binary trees $\mathcal{T}^*$. Note that φ^* (which depends on the original formula φ and the fixed treewidth w), has to be computed only once *for every problem* described in this paper. In contrast, the computation of a tree decomposition $\mathcal{T}$ and the transformation of $\mathcal{T}$ into $\mathcal{T}^*$ has to be done *for every problem instance*. The final step is the the actual model-checking.

So far, we have only implemented the DLP-consistency problem. The input MSO-formula φ from the proof of Theorem 5 was transformed into φ^* in an ad hoc manner – depending on the chosen treewidth w. For the computation of the tree decomposition $\mathcal{T}$, we are planning to implement Bodlaender's algorithm, see (Bodlaender 1996). For the time being, we are considering the tree decomposition as an additional part of the input. Our transformation of $\mathcal{T}$ into $\mathcal{T}^*$ essentially implements the algorithm from (Flum, Frick, & Grohe 2002) – apart from some simplifications that are possible here (e.g., due to the fact that we have no predicates over sets of constants in the input database). For the last step, we decided to take advantage of an existing MSO-model checking tool. We have thus chosen MONA, see (Klarlund, Møller, & Schwartzbach 2002). To the best of our knowledge, MONA is the only existing such tool.

Practical Experience and Discussion.

We have experimented with our prototype implementations on several instances of the DLP-consistency problem. Some experimental results are reported in Table 1, where "tw" means the treewidth, "# var." (resp. "# cl.") means the number of variables (resp. clauses), "# tn" refers to the number of nodes in the tree decomposition, and "# rel. ins." stands for the number of relational instances in the relational structure. The computation time (in seconds) was measured on a PC under Linux with an Intel Pentium M processor 1.60GHz.

So in principle, it has been proved that the reduction to a model checking problem and the assembling of components as described above is a viable way of solving the desired KR & R problems. However, as far as the observed runtime is concerned, the experimental results are far from the linear time behavior according to the theoretical results shown in the previous section. An analysis of the various components of our program has revealed that MONA is the weak point of the program. In fact, the way how MONA evaluates an MSO formula φ^* over a tree $\mathcal{T}^*$ is very problematical (and, in a sense, contradicts the spirit of model checking). The correct way of handling this model checking task would be as follows, see (Flum, Frick, & Grohe 2002): The MSO formula φ^* has to be compiled "once and for all" into a finite tree automaton (FTA, for short) and this FTA has to be run on any input tree $\mathcal{T}^*$. Instead, MONA also considers the ground atoms encoding $\mathcal{T}^*$ as part of the formula and then compiles the resulting MSO-formula (consisting of φ^* plus the encoding of $\mathcal{T}^*$) into an FTA. In other words, the size of the FTA grows (exponentially!) with the size of $\mathcal{T}^*$. Consequently, on the one hand, the runtime needed by MONA is far from linear. On the other hand – even worse – the state explosion of the FTA for an increasing size of $\mathcal{T}^*$ led to runtime errors of MONA caused by "memory leak". In the experiments described in Table 1, this negative effect already happened for # var. = 10 and # cl. = 7.

MONA undoubtedly has its merits in other areas, notably in verification. However, with its current strategy of considering the input structure as part of the formula (and, therefore, mixing up data complexity and query complexity), MONA is not suited for the KR & R problems studied here. Hence, in the long run, we either need a general model checking tool that does distinguish between the formula and the data (unfortunately, such a tool currently does not exit) or we have to implement the model checking part (i.e., step IV of our program) ourselves – based on the algorithm described in (Flum, Frick, & Grohe 2002). The resulting system architecture of our approach is depicted in Figure 2: The steps I, II, and III are clear. However, it is crucial to divide step IV into two substeps: Step IVa, which computes a finite tree automaton from the MSO-formula φ^*, and Step IVb, which runs the FTA on every tree $\mathcal{T}^*$. The important point to notice is that steps I and IVa (depicted on the left-hand side of Figure 2) are carried out once and for all while the steps I, II, and III (shown on the right-hand side) have to be repeated for every single input structure $\mathcal{A}_i$.

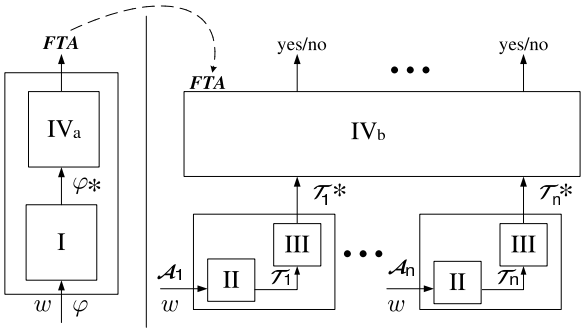

Figure 2: Proposed System Architecture

tw	# var.	# cl.	# tn	# rel. ins.	time
4	4	1	2	8	0.10
4	5	2	4	12	0.17
4	6	3	6	16	0.52
4	7	4	8	20	0.65
4	8	5	10	24	1.27
4	9	6	12	28	3.54

Table 1: Experiments with the DLP-Reasoning Problem

Conclusions and Future Work

In this paper, we have introduced a new approach (based on Courcelle's Theorem) to solving a whole range of problems in knowledge representation and reasoning. Despite the high worst case complexity of these problems, we have thus managed to prove that they are all solvable in linear time if the treewidth of the considered formulae (resp. of the DLPs) is bounded by some constant. We have also experimented with a prototype implementation based on the algorithm of (Flum, Frick, & Grohe 2002) but using the general tool MONA for the actual model checking part. The results obtained by these experiments show the feasibility of our approach in principle.

However, the experience with this prototype also clearly suggests the need for a dedicated system that is built from scratch. For the reasons explained in the previous section, MONA is not suitable for our purposes. Hence, MONA should be replaced in our program by an implementation of

the model checking algorithm from (Flum, Frick, & Grohe 2002). Only with the strict separation of data complexity and query complexity guaranteed by the latter algorithm, we can obtain the desired linear time behavior of the overall program. Anyway, even then, searching for further optimizations is an important target of future research.

Note that a dedicated system has a further advantage: It would allow us to implement extensions like the *extended monadic second order* language from (Arnborg, Lagergren, & Seese 1991), which is needed to encode several problems in the context of abduction. With MONA used a black box, such extensions are impossible.

As was mentioned in the previous section, we are currently considering the tree decomposition of the relational structure as part of the input. In the future, we want to automate the computation of a tree decomposition of given width w. Theoretically, this is a well-studied problem, which can be solved in linear time for fixed w, see (Bodlaender 1996). However, the large constant factor of Bodlaender's algorithm (i.e., exponential w.r.t. w^3) may turn out to be problematical in practice. Hence, in order to cope not only with really small values of w, we have to be careful with the selection of an appropriate algorithm. In particular, there exist quadratic time tree decomposition algorithms (Reed 1992) which – depending on the concrete situation – may be preferable to the linear one. Hence, extensive experimenting with various tree decomposition algorithms is needed.

We have shown here one way how KR & R problems can be represented as graphs and that the idea of graph decompositions and the notion of some width can be fruitfully applied to this area. However, also other mappings of the KR & R problems to graphs – in particular, to directed graphs – are conceivable. Moreover, recently, other interesting notions of decompositions and width have been developed, see e.g. (Berwanger & Grädel 2005),(Berwanger *et al.* 2006). We are planning to investigate their applicability to the KR & R problems studied here.

References

Arnborg, S.; Lagergren, J.; and Seese, D. 1991. Easy Problems for Tree-Decomposable Graphs. *J. Algorithms* 12(2):308–340.

Arnborg, S. 1985. Efficient Algorithms for Combinatorial Problems with Bounded Decomposability - A Survey. *BIT* 25(1):2–23.

Baaz, M.; Egly, U.; and Leitsch, A. 2001. Normal Form Transformations. In Robinson, J. A., and Voronkov, A., eds., *Handbook of Automated Reasoning*, volume 1. Elsevier Science. chapter 5, 273–333.

Berwanger, D., and Grädel, E. 2005. Entanglement - A Measure for the Complexity of Directed Graphs with Applications to Logic and Games. In *Proc. LPAR 2004*, volume 3452 of *LNCS*, 209–223.

Berwanger, D.; Dawar, A.; Hunter, P.; and Kreutzer, S. 2006. DAG-Width and Parity Games. In *Proc. STACS 2006*, LNCS.

Bodlaender, H. L. 1993. A tourist guide through treewidth. *Acta Cybern.* 11(1-2):1–22.

Bodlaender, H. L. 1996. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM J. Comput.* 25(6):1305–1317.

Cadoli, M., and Lenzerini, M. 1990. The Complexity of Closed World Reasoning and Circumscription. In *Proc. AAAI-90*, 550–555.

Courcelle, B. 1990. Graph Rewriting: An Algebraic and Logic Approach. In *Handbook of Theoretical Computer Science, Volume B*. Elsevier Science Publishers. 193–242.

Downey, R. G., and Fellows, M. R. 1999. *Parameterized Complexity*. New York: Springer.

Eiter, T., and Gottlob, G. 1993. Propositional Circumscription and Extended Closed World Reasoning are Π_2^p-complete. *Theoretical Computer Science* 114:231–245.

Eiter, T., and Gottlob, G. 1995a. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Annals of Math. and Artif. Intell.* 15(3/4):289–323.

Eiter, T., and Gottlob, G. 1995b. The Complexity of Logic-Based Abduction. *J. ACM* 42(1):3–42.

Flum, J.; Frick, M.; and Grohe, M. 2002. Query evaluation via tree-decompositions. *J. ACM* 49(6):716–752.

Gelfond, M., and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In *Proc. ICLP/SLP*, 1070–1080.

Gelfond, M.; Przymusinska, H.; and Przymusinski, T. C. 1989. On the Relationship Between Circumscription and Negation as Failure. *Artif. Intell.* 38(1):75–94.

Gottlob, G.; Scarcello, F.; and Sideri, M. 2002. Fixed-Parameter Complexity in AI and Nonmonotonic Reasoning. *Artif. Intell.* 138(1-2):55–86.

Klarlund, N.; Møller, A.; and Schwartzbach, M. I. 2002. MONA Implementation Secrets. *International Journal of Foundations of Computer Science* 13(4):571–586. World Scientific Publishing Company. Earlier version in Proc. CIAA '00, LNCS vol. 2088.

Przymusinski, T. C. 1991. Stable Semantics for Disjunctive Programs. *New Generation Comput.* 9(3/4):401–424.

Reed, B. 1992. Finding approximate separators and computing treewdith quickly. In *ACM Symposium on Theory of Computing*.

Szeider, S. 2004. On Fixed-Parameter Tractable Parameterizations of SAT. In *Proc. 6th Int. Conf. SAT 2003, Selected Revised Papers*, volume 2919 of *LNCS*, 188–202.