Efficient Datalog Abduction through Bounded Treewidth

Georg Gottlob
Computing Laboratory
Oxford University
Oxford OX1 3QD,UK

Reinhard Pichler
Information Systems Institute
Vienna University of Technology
A-1040 Vienna, Austria

Fang Wei
Information Systems Institute
Vienna University of Technology
A-1040 Vienna, Austria

Abstract

Abductive diagnosis is an important method for identifying possible causes which explain a given set of observations. Unfortunately, abduction suffers from the fact that most of the algorithmic problems in this area are intractable. We have recently obtained very promising results for a strongly related problem in the database area. Specifically, the PRIMALITY problem becomes efficiently solvable and highly parallelizable if the underlying functional dependencies have bounded treewidth (Gottlob, Pichler, & Wei 2006b). In the current paper, we show that these favorable results can be carried over to logic-based abduction. In fact, we even show a further generalization of these results.

Introduction

Abductive diagnosis aims at an explanation of some observed symptoms in terms of minimal sets of hypotheses (like failing components) which may have led to these symptoms (de Kleer, Mackworth, & Reiter 1992). Unfortunately, most of the decision problems in logic-based abduction are intractable (Eiter & Gottlob 1995). For instance, the *relevance* problem (i.e., deciding if a hypothesis is part of a possible explanation) is NP hard, even if the system description consists of propositional, definite Horn clauses only (Friedrich, Gottlob, & Nejdl 1990). Hence, it is an important task to search for sufficient conditions under which these practically important problems become efficiently solvable.

The concept of treewidth and related notions have been successfully applied to many areas of Computer Science. In recent years, several intractable problems in the database field and in AI (such as e.g., conjunctive query equivalence and CSP problems) have been shown to become solvable in polynomial time or even highly parallelizable if the underlying graph or hypergraph structure has bounded treewidth or hypertree width (Gottlob, Leone, & Scarcello 2002). In fact, it is generally believed that many practically relevant problems actually do have low treewidth, see e.g. the discussion of applications in (Bodlaender 1993).

We have recently devised new algorithms for several longstanding intractable problems in the database area, such as the PRIMALITY problem (i.e., testing if a given attribute is part of a key). Moreover, we showed that these algorithms work in linear time and are amenable to parallelization if the underlying functional dependencies have bounded treewidth (Gottlob, Pichler, & Wei 2006b). This property is also known as fixed-parameter tractability. Note that the high inherent complexity of these problems was a serious obstacle to the automation of database design.

It turns out that our results have interesting applications to logic-based abduction. By the well-known relationship with the PRIMALITY problem, we shall make our new algorithms also applicable to the relevance problem of propositional abduction if the system description is given by a set of propositional, definite Horn clauses with bounded treewidth.

Moreover, in this paper, we also present a significant extension of these results to abductive datalog. It is thus possible to actually tackle many real-world problems in AI with these methods. Suppose that a system description (e.g., of an electronic circuit) is given in form of a finite structure (i.e., the "extensional database", EDB) of bounded treewidth and a guarded *non-ground* datalog program that describes the propagation of faulty behavior. We will show that, also in this case, it can be determined efficiently whether the misbehavior of some system component is a possible cause for the observed symptoms. It should be noted that this extension to the non-ground case is by no means obvious, notwithstanding the clear analogy between the functional dependencies in database design and *propositional* abduction. The difficulty in proving our new result consists in showing that the abductive datalog problem is equivalent to a propositional abduction problem *and* the bounded treewidth of the EDB indeed propagates to the propositional rules.

Another possibility to prove the fixed-parameter tractability (FPT) of the above mentioned problems is to show that they can be expressed by monadic second-order (MSO) formulae. The FPT thus follows immediately via Courcelle's Theorem (Courcelle 1990). A concrete algorithm can be obtained by constructing a finite tree automaton (FTA) corresponding to the MSO formula and by checking whether a tree obtained from the tree decomposition is accepted by the FTA. In a recent paper (Gottlob, Pichler, & Wei 2006a), we discussed the feasibility of this method for the propositional case. However, it turns out that such an MSO-to-FTA transformation severely suffers from a state explosion and tends to be excessively complicated. Thus – as was pointed out in

(Grohe 1999) – it is clearly preferable to have a dedicated algorithm rather than just an MSO-encoding.

Tractability Results on Primality

In this section, we recall some basic notions and results on database design and tree decompositions. Moreover, we briefly review some main results from (Gottlob, Pichler, & Wei 2006b). In particular, we recall that the PRIMALITY problem can be solved very efficiently if the functional dependencies have bounded treewidth.

Primality Problem

A *relational schema* is denoted as (R, F) where R is the set of attributes and F is the set of *functional dependencies* (FDs, for short) over R. A functional dependency f is written in the form $X \to A$, where $X \subseteq R$ and $A \in R$. We write $lhs(f)$ and $rhs(f)$ to denote the left-hand side X and the right-hand side A of f, respectively.

The set of all FDs that hold in a given schema (R, F) can be derived from F via Armstrong's Axioms, see (Armstrong 1974; Mannila & Räihä; 1992). The derivation of new functional dependencies can be characterized via the notion of a *derivation sequence* in the following way: A functional dependency $Y \to B$ holds in a schema (R, F) iff there exists a sequence of the form $Y \to Y \cup \{B_1\} \to Y \cup \{B_1, B_2\} \to \ldots \to Y \cup \{B_1, \ldots, B_n\}$, s.t. $B_n = B$ and for every $i \in \{1, \ldots, n\}$, there exists an FD $f \in F$ with $lhs(f) \subseteq Y \cup \{B_1, \ldots, B_{i-1}\}$ and $rhs(f) = B_i$.

In other words, the derivation of all functional dependencies that hold in a given schema (R, F) works pretty much like reasoning with propositional Horn clauses. If an FD $Y \to B$ can be derived from F, then we write $F \models Y \to B$ and say that "Y determines B" in the schema (R, F).

Given a relational schema (R, F) and a subset $X \subseteq R$, if X determines all attributes $A \in R$, then X is called a *superkey*. If X is minimal with this property, then X is a *key*. The set of all keys in (R, F) is denoted as $K(R, F)$. An attribute A is called *prime* in (R, F), if it is contained in at least one key in $K(R, F)$.

The following example, which is a slightly modified version of an example given in (Lucchesi & Osborn 1978), will help to illustrate these notions:

Example 1 Consider a student record database consisting of the relational schema with attributes $R = \{$student_ID, student_name, course, professor, time$\}$ and with the following functional dependencies:

- $\text{student\_name} \to \text{student\_ID}$
- $\text{student\_ID} \to \text{student\_name}$
- $\text{student\_name}, \text{course} \to \text{professor}$
- $\text{student\_name}, \text{course} \to \text{time}$
- $\text{professor}, \text{time} \to \text{course}$

It is easy to verify that $X = \{\text{student\_name}, \text{course}\}$ is a key of this schema: On the one hand, the functional dependencies $X \to \text{student\_ID}$, $X \to \text{professor}$, $X \to \text{time}$ clearly hold and, therefore, X is a superkey. On the other hand, X is minimal with this property since neither $\{\text{student\_name}\}$ nor $\{\text{course}\}$ is a superkey.

Note that $Y = \{\text{student\_ID}, \text{course}\}$ is another key of this schema. For instance, the FD $Y \to \text{time}$ can be derived via the derivation sequence $Y \to Y \cup \{\text{student\_name}\} \to Y \cup \{\text{student\_name}, \text{time}\}$. The FDs $Y \to \text{professor}$ and $Y \to \text{student\_name}$ can be derived by similar arguments. It is easy to check that Y is a *minimal* superkey, i.e., a key.

Similarly, we can find two more keys $\{\text{student\_name}, \text{time}, \text{professor}\}$ and $\{\text{student\_ID}, \text{time}, \text{professor}\}$. □

Let (R, F) be a relational schema and $X \subseteq R$. We define the *projection* of F onto X as $F[X] = \{Y \to Z \mid F \models Y \to Z \text{ and } Y \subseteq X, Z \in X\}$. Let $R' \subseteq R$. Then the schema $(R', F[R'])$ is referred to as a *subschema* of (R, F). Let further $A \in R'$ be an attribute. A is prime in the subschema $(R', F[R'])$ if it is contained in one key in $K(R', F[R'])$. It is well-known that, in general, the FDs $F[R']$, which hold in a subschema, have no polynomial-size representation. The "classical" example for this phenomenon is as follows, see (Fischer, Jou, & Tsou 1983; Mannila & Räihä; 1992):

$$R = \{A_1, \ldots, A_n, B_1, \ldots, B_n, C_1, \ldots, C_n, D\}$$
$$F = \{A_i \to C_i, B_i \to C_i \mid 1 \leq i \leq n\} \cup \{C_1, \ldots, C_n \to D\}$$
$$R' = \{A_1, \ldots, A_n, B_1, \ldots, B_n, D\}$$

It is easy to verify that in $(R', F[R'])$ all 2^n dependencies of the form $Z_1, \ldots, Z_n \to D$ with $Z_i \in \{A_i, B_i\}$ hold. Moreover, $F[R']$ admits no representation of polynomial size.

Tree Decompositions and Treewidth

A *hypergraph* is a pair $\mathcal{H} = \langle V, H \rangle$ consisting of a set V of vertices and a set H of hyperedges. A hyperedge $h \in H$ is a subset of V. A measure for the "tree-likeness" of a hypergraph $\mathcal{H}$ is its treewidth defined below. A *tree decomposition* $\mathcal{T}$ of $\mathcal{H}$ is a pair $\langle T, \lambda \rangle$, 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. $\forall h \in H$, there exists a node N in T, s.t. $h \subseteq \lambda(N)$.
3. "connectedness condition": $\forall v \in V$, the set of nodes $\{N \mid v \in \lambda(N)\}$ induces a connected subtree of T.

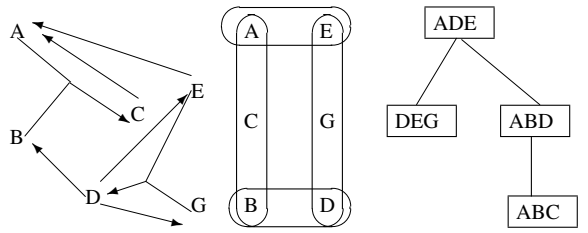
The sets $\lambda(N)$ are referred to as *bags*. The *width* of a tree decomposition $\langle T, \lambda \rangle$ is defined as $\max(\{|\lambda(N)| - 1 : N \text{ node in } T\})$. The *tree-width* $tw(\mathcal{H})$ of a hypergraph $\mathcal{H}$ is the minimum width over all its tree decompositions. Note that trees are precisely the hypergraphs with treewidth = 1.

The notions of *tree decomposition* and *treewidth* of a relational schema, a logic program, or a finite structure are defined in the obvious way via the *corresponding hypergraph*.

For a relational schema (R, F), we define the corresponding hypergraph as $\mathcal{H} = \langle V, H \rangle$ with $V = R$ and $H = \{\{A_1, \ldots, A_n, B\} \mid (A_1 \ldots A_n \to B) \in F\}$.

For a propositional logic program, we proceed analogously by defining the hyperedges $\{A_1, \ldots, A_n, B\}$ in $\mathcal{H}$ via the rules $B \leftarrow A_1 \ldots A_n$ rather than the FDs.

For a finite structure $\mathfrak{A}$ with universe A we define the corresponding hypergraph as $\mathcal{H} = \langle V, H \rangle$ with $V = A$ and $H = \{\{a_1, \ldots, a_n\} \mid \mathfrak{A} \text{ contains a ground atom } P(a_1 \ldots a_n) \text{ for some predicate symbol } P\}$.



(a)FDs (b)Hypergraph (c)Tree deomposition

Figure 1: FDs, hypergraph and decomposition of Example 2

Example 2 Consider the relational schema (R, F) with attributes $R = \{A, B, C, D, E, G\}$ and FDs $F = \{AB \rightarrow C, C \rightarrow A, D \rightarrow B, EG \rightarrow D, D \rightarrow E, D \rightarrow G, E \rightarrow A\}$. The FDs and the corresponding hypergraph $\mathcal{H}$ are shown in Figure 1 together with a possible tree decomposition. This tree decomposition has width 2. Note that $tw(\mathcal{H}) \geq 2$ since $\mathcal{H}$ is not a tree. Hence, the decomposition in Figure 1 is optimal, and we have $tw(R, F) = 2$. □

Complexity

LogCFL is the class of decision problems which are log-space reducible to a context-free language. The relationship between LogCFL and other well-known complexity classes is summarized as follows:

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{LogCFL} \subseteq \mathsf{AC}^1 \subseteq \mathsf{NC}^2 \subseteq \mathsf{P}$$

By L (resp. NL) we denote the problems that can be decided in deterministic (resp. non-deterministic) log-space. The class P contains the problems decidable in deterministic polynomial time. The classes AC^i and NC^i are logspace-uniform circuit-based parallel computation classes. For details on these classes, see e.g. (Johnson 1990). Since $\mathsf{LogCFL} \subseteq \mathsf{AC}^1 \subseteq \mathsf{NC}^2$, the problems in LogCFL are highly parallelizable.

Bounded Treewidth & Primality

It has already been mentioned that testing whether some attribute A is prime in a relational schema (R, F) is NP-complete (Mannila & Räihä; 1992). In (Gottlob, Pichler, & Wei 2006b) we showed that if the relational schema has bounded treewidth, the PRIMALITY test becomes tractable. In fact, we even showed that this problem falls into the highly parallelizable complexity class LogCFL.

Theorem 3 *(Gottlob, Pichler, & Wei 2006b) Let (R, F) be a relational schema whose treewidth is bounded by some constant $k \geq 1$ and let $A \in R$ be an attribute. It can be decided in linear time whether A is prime in (R, F). Moreover, this decision problem is in* LogCFL.

Theorem 4 *(Gottlob, Pichler, & Wei 2006b) Let (R, F) be a relational schema whose treewidth is bounded by some constant $k \geq 1$ and let $(R', F[R'])$ be a subschema with $R' \subseteq R$, and $A \in R'$ an attribute. Then it can be decided in linear time whether A is prime in $(R', F[R'])$. Moreover, this decision problem is in* LogCFL.

Propositional Abduction

In this work, we study *propositional abduction problems* (PAPs) of the following form.

Definition 5 A *propositional abduction problem* (a PAP, for short) $\mathcal{P}$ consists of a tuple $\langle V, Hyp, Obs, SD\rangle$, where V is a finite set of *variables*, $Hyp \subseteq V$ is the set of *hypotheses*, $Obs \subseteq V$ is the set of *observed symptoms*, and SD is a *system description* in the form of a set of definite, propositional Horn clauses with $SD \cup Hyp \models Obs$. Moreover, we assume that $|Obs|$ is bounded by some fixed constant k.

A set $\Delta \subseteq Hyp$ is a *diagnosis* (also called *solution*) to $\mathcal{P}$ if Δ is minimal s.t. $SD \cup \Delta \models Obs$ holds. A hypothesis $h \in Hyp$ is called *relevant* if h is contained in at least one diagnosis Δ of $\mathcal{P}$.

Note that the restriction on the cardinality of Obs is not a severe one since $|Obs|$ is often even assumed to be 1 (e.g., an alarm bell ringing, a bulb ligthing up, etc.). The condition $SD \cup Hyp \models Obs$ is a prerequisite to ensure that there exists at least one diagnosis. This condition can be checked in linear time, see (Dowling & Gallier 1984; Minoux 1988). Deciding the relevance of a hypothesis h in a PAP $\mathcal{P}$ is NP-complete even with the above restrictions on SD, Hyp, and Obs (Friedrich, Gottlob, & Nejdl 1990).

Example 6 Consider the following PAP describing problems of a football team (Hermann & Pichler 2007).

$$\begin{aligned} SD &= \{\ \text{weak\_defense} \vee \text{weak\_attack} \rightarrow \text{match\_lost},\\ &\quad\ \text{match\_lost} \rightarrow \text{manager\_sad} \wedge \text{press\_angry}\\ &\quad\ \text{star\_injured} \rightarrow \text{manager\_sad} \wedge \text{press\_sad}\ \}\\ Obs &= \{\ \text{manager\_sad, press\_angry}\ \}\\ Hyp &= \{\ \text{weak\_defense, weak\_attack, star\_injured}\ \} \end{aligned}$$

It is convenient to abbreviate the propositional variables weak_defense, weak_attack, star_injured, match_lost, manager_sad, press_angry, and press_sad in this order as A_1, A_2, A_3, A_4, A_5, A_6, A_7. Obviously, SD is equivalent to the following set of definite Horn clauses

$$\begin{aligned} SD' = \{\ & A_4 \leftarrow A_1, A_4 \leftarrow A_2, A_5 \leftarrow A_4,\\ & A_6 \leftarrow A_4, A_5 \leftarrow A_3, A_7 \leftarrow A_3\}. \end{aligned}$$

This PAP has two diagnoses, $\Delta_1 = \{A_1\}$ and $\Delta_2 = \{A_2\}$ (i.e., weak_defense and weak_attack, respectively). □

We are now ready to show that the tractability results from PRIMALITY can indeed be carried over to abduction.

Theorem 7 *Let $\mathcal{P} = \langle V, Hyp, Obs, SD\rangle$ be a PAP where SD has bounded treewidth and let $h \in Hyp$. It can be decided in linear time whether h is relevant in $\mathcal{P}$. Moreover, this decision problem is in* LogCFL.

Proof. By Theorem 4, it suffices to show that there is a log-space reduction from the relevance problem to the PRIMALITY problem in a subschema s.t. the increase of the treewidth is bounded by a constant. By slight abuse of notation, we identify any definite Horn rule $B \leftarrow A_1 \ldots A_n$ with the FD $A_1 \ldots A_n \rightarrow B$. Then we reduce an arbitrary PAP $\mathcal{P} = \langle V, Hyp, Obs, SD\rangle$ to the relational schema (R, F) with $R = V$ and $F = SD \cup \{Obs \rightarrow B | B \in Hyp\}$. Moreover, we set $R' = Hyp$.

This reduction is clearly feasible in log-space. Moreover, we can obtain a tree decomposition of F from a tree decomposition of SD by adding Obs to every bag. Since we are only considering PAPs with $|Obs| \leq k$ for some constant k, we have $tw(F) \leq tw(SD) + k$. It remains to prove the correctness of this reduction, i.e. $h \in Hyp$ is relevant in the PAP $\mathcal{P}$ iff h is prime in the subschema $(R', F[R'])$. It suffices to show that, for every $\Delta \subseteq Hyp$, the following equivalence holds:

Δ is a diagnosis of $\mathcal{P} \Leftrightarrow \Delta$ is a key of $(R', F[R'])$.

Before we prove the two directions of this equivalence, we comment on the notation used below. Recall that we are identifying the set of definite Horn rules SD with the set of functional dependencies SD and vice versa. Hence, we shall write $SD \cup \Delta \models A$ in order to denote that the propositional atom A is implied by the propositional logic program $SD \cup \Delta$. Alternatively, we shall write $SD \models \Delta \rightarrow A$ in order to denote that the FD $\Delta \rightarrow A$ can be derived from the FDs SD. It is easy to check that these two conditions are equivalent.

"$\Rightarrow$" Suppose that $\Delta \subseteq Hyp$ is a solution of the PAP $\mathcal{P}$. We show that then (i) Δ is a superkey in $(R', F[R'])$ and (ii) Δ is minimal with this property.

(i) Since Δ is a solution of $\mathcal{P}$, we have $SD \cup \Delta \models A$ for every atom $A \in Obs$. Hence, in (R, F), the FD $\Delta \rightarrow A$ can be derived for every attribute $A \in Obs$. Thus, by the additional FDs $Obs \rightarrow B$, we can derive in (R, F) also the FD $\Delta \rightarrow B$ for every $B \in Hyp$. Note that $\Delta \subseteq Hyp$ and $B \in Hyp$. Hence, we actually have $F[R'] \models \Delta \rightarrow B$ for all $B \in Hyp$. Thus, Δ is a superkey in $(R', F[R'])$.

(ii) We prove the minimality of Δ indirectly. Suppose to the contrary that there exists a strictly smaller key $\Delta' \subset \Delta$ of $(R', F[R'])$. Exactly as in part (i) of the "$\Leftarrow$"-direction treated below, one can show that then $SD \cup \Delta' \models Obs$ holds, contradicting the minimality of the solution Δ of $\mathcal{P}$.

"$\Leftarrow$" Suppose that $\Delta \subseteq Hyp$ is a key of $(R', F[R'])$. We show that then (i) $SD \cup \Delta \models Obs$ and (ii) Δ is minimal with this property.

(i) By assumption, $F[R'] \models \Delta \rightarrow B$ for all $B \in Hyp$. First suppose that for at least one B, the rule $Obs \rightarrow B$ is used in the derivation of B from Δ. This means that all FDs $\Delta \rightarrow A$ for all $A \in Obs$ can be derived from SD. In terms of the PAP $\mathcal{P}$, we thus have the desired implication $SD \cup \Delta \models Obs$. On the other hand, suppose that all of the attributes B can be derived from Δ by only using the FDs in SD, i.e.: $SD \models \Delta \rightarrow B$ for every $B \in Hyp$ or, equivalently, $SD \cup \Delta \models Hyp$. Recall from Definition 5 that, in any PAP, the condition $SD \cup Hyp \models Obs$ holds. Hence, in total, we have the desired implication $SD \cup \Delta \models Obs$.

(ii) It remains to show the minimality of Δ. Suppose to the contrary that there exists a strictly smaller solution $\Delta' \subset \Delta$ of the PAP $\mathcal{P}$. Exactly as in part (i) of the "$\Rightarrow$"-direction, one can show that then Δ' is also a superkey of $(R', F[R'])$, which contradicts the minimality of the key Δ of $(R', F[R'])$. □

Example 8 Let us revisit the PAP $\mathcal{P}$ from Example 6. By the proof of Theorem 7, we can reduce $\mathcal{P}$ to the relational schema (R, F) with $R = V = \{A_1, \ldots, A_7\}$ and

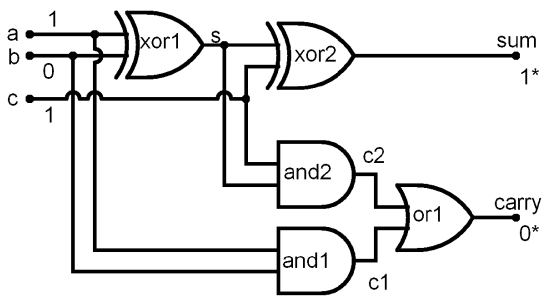


Figure 2: Full-adder with incorrect output values.

$$F = \{\, A_1 \rightarrow A_4, A_2 \rightarrow A_4, A_4 \rightarrow A_5, \\ A_4 \rightarrow A_6, A_3 \rightarrow A_5, A_3 \rightarrow A_7, \\ A_5A_6 \rightarrow A_1, A_5A_6 \rightarrow A_2, A_5A_6 \rightarrow A_3\}.$$

Moreover, let $R' = \{A_1, A_2, A_3\}$. Then the subschema $(R', F[F'])$ has two keys, namely $K_1 = \{A_1\}$ and $K_2 = \{A_2\}$, which correspond to the diagnoses of the PAP $\mathcal{P}$. □

Datalog Abduction

In recent years, the datalog language has been successfully applied as a knowledge representation mechanism in the area of abductive diagnosis (Balsa, Dahl, & Lopes 1995; Koshutanski & Massacci 2003; Bonatti & Samarati 2000). In this paper, we will restrict our attention to the so-called guarded fragment of datalog:

Definition 9 (Gottlob, Grädel, & Veith 2002) A *guard* of a datalog rule is an atom A whose predicate symbol occurs in the input structure (i.e., the "EDB") s.t. all variables of the rule occur in A. A datalog rule is *guarded* if its body contains a guard. A guarded datalog program is a datalog problem whose rules are guarded.

Then we define datalog abduction as follows.

Definition 10 A *datalog abduction problem* consists of a tuple $\mathcal{P} = \langle EDB, P, Hyp, Obs \rangle$, where EDB is an input structure (i.e., set of ground atoms), P is a set of definite Horn datalog rules, Hyp and Obs are sets of ground atoms with $EDB \cup P \cup Hyp \models Obs$. As in the propositional case, we assume $|Obs| \leq k$ for some constant k. Moreover, we consider the set of predicate symbols occurring in $\mathcal{P}$ as arbitrarily chosen but fixed.

A *diagnosis* is a minimal subset $\Delta \subseteq Hyp$ with $EDB \cup P \cup \Delta \models Obs$. An atom $h \in Hyp$ is called *relevant*, if there exists at least one diagnosis Δ with $h \in \Delta$.

Note that in a datalog abduction problem, the system description consists of an input structure (the EDB) and a (normally non-ground) datalog program P. A PAP as defined in the previous section corresponds to the special case where the EDB is omitted and P contains only ground rules.

Example 11 Consider the the full-adder in Figure 2, which has faulty output bits (indicated by *). We describe this diagnosis problem by a datalog abduction problem:

$$EDB = \{\mathsf{one}(a), \mathsf{zero}(b), \mathsf{one}(c), \mathsf{xor}(a, b, s, xor_1), \\ \mathsf{xor}(s, c, sum, xor_2), \mathsf{and}(a, b, c_1, and_1), \\ \mathsf{and}(s, c, c_2, and_2), \mathsf{or}(c_1, c_2, carry, or_1)\}$$

The guarded datalog program P contains rules that model the *normal* and the *faulty* behavior for each gate type (i.e., and, or, and xor). We only show the datalog rules for the gate type xor. The other types are handled analogously. For the normal behavior, we have the following four rules:

$$\begin{aligned}
\mathsf{zero}(O) &\leftarrow \mathsf{xor}(I_1, I_2, O, G), \mathsf{one}(I_1), \mathsf{one}(I_2).\\
\mathsf{one}(O) &\leftarrow \mathsf{xor}(I_1, I_2, O, G), \mathsf{one}(I_1), \mathsf{zero}(I_2).\\
\mathsf{one}(O) &\leftarrow \mathsf{xor}(I_1, I_2, O, G), \mathsf{zero}(I_1), \mathsf{one}(I_2).\\
\mathsf{zero}(O) &\leftarrow \mathsf{xor}(I_1, I_2, O, G), \mathsf{zero}(I_1), \mathsf{zero}(I_2).
\end{aligned}$$

where the atom $\mathsf{xor}(I_1, I_2, O, G)$ is the *guard* in all rules. The faulty behavior is modeled by another set of four rules:

$$\begin{aligned}
\mathsf{one}(O) &\leftarrow \mathsf{xor}(I_1, I_2, O, G), \mathsf{one}(I_1), \mathsf{one}(I_2), \mathsf{faulty}(G).\\
\mathsf{zero}(O) &\leftarrow \mathsf{xor}(I_1, I_2, O, G), \mathsf{one}(I_1), \mathsf{zero}(I_2), \mathsf{faulty}(G).\\
\mathsf{zero}(O) &\leftarrow \mathsf{xor}(I_1, I_2, O, G), \mathsf{zero}(I_1), \mathsf{one}(I_2), \mathsf{faulty}(G).\\
\mathsf{one}(O) &\leftarrow \mathsf{xor}(I_1, I_2, O, G), \mathsf{zero}(I_1), \mathsf{zero}(I_2), \mathsf{faulty}(G).
\end{aligned}$$

Finally, we set $Obs := \{\mathsf{one}(sum), \mathsf{zero}(carry)\}$ and $Hyp := \{\mathsf{faulty}(xor_1), \mathsf{faulty}(xor_2), \mathsf{faulty}(and_1), \mathsf{faulty}(and_2), \mathsf{faulty}(or_1)\}$.

Of course, if one has already verified that some component c works properly, then one will remove the atom $\mathsf{faulty}(c)$ from Hyp. This abduction problem has three diagnoses, namely $\Delta_1 = \{\mathsf{faulty}(xor_1)\}$, $\Delta_2 = \{\mathsf{faulty}(xor_2), \mathsf{faulty}(or_1)\}$, and $\Delta_3 = \{\mathsf{faulty}(xor_2), \mathsf{faulty}(and_2)\}$. □

We now extend the fixed-parameter tractability result of propositional abduction to datalog abduction.

Theorem 12 *Let $\mathcal{P} = \langle EDB, P, Hyp, Obs\rangle$ be a datalog abduction problem, s.t. the input structure EDB has bounded treewidth. Then, for any $h \in Hyp$, it can be decided in polynomial time whether h is relevant in $\mathcal{P}$.*

Proof. By Theorem 7, it suffices to show the following two facts: First we have to show that the datalog program P is equivalent to a ground program of polynomial size, and second we have to show that the bounded treewidth propagates from the input structure to the ground program.

1. Size of an equivalent ground program. The grounding of a datalog program Π relatively to some structure $\mathfrak{A}$ is obtained by computing all possible instantiations of the variables occurring in Π by all constants in the active domain (Ceri, Gottlob, & Tanca 1990). The resulting program $ground(\Pi)$ is equivalent to Π, i.e., for any atom A, we have $\Pi \cup \mathfrak{A} \models A \Leftrightarrow ground(\Pi) \cup \mathfrak{A} \models A$. In general, the program $ground(\Pi)$ is exponentially big. However, if Π is a guarded datalog program, then one can restrict $ground(\Pi)$ to an equivalent set $ground'(\Pi)$ which can be computed in quadratic time and whose size is also quadratically bounded, namely $O(|\mathfrak{A}| \cdot |\Pi|)$ (Gottlob, Grädel, & Veith 2002). Intuitively, this can be seen as follows: For each rule r in Π, there are at most $|\mathfrak{A}|$ instantiations for the guard of r which actually exist in $\mathfrak{A}$. On the other hand, all ground rules where the guard is instantiated to an atom outside $\mathfrak{A}$ can be simply deleted (because this body will never be true).

2. Bounded treewidth of the ground program. By assumption, there exists a tree decomposition $\mathcal{T}$ of EDB of width $< k$ for some constant k. Note that the bags $\lambda(N)$ in $\mathcal{T}$ are sets of domain elements. It is convenient to denote the domain elements in a ground atom A as $dom(A)$. Moreover, we refer to the ground atoms obtained by instantiating a guard atom as "ground guards". For every ground guard A, we may assume w.l.o.g., that $\mathcal{T}$ has a leaf node N s.t. $\lambda(N) = dom(A)$, since we may clearly append a new leaf node N with this property to a node N' with $\lambda(N) \supseteq dom(A)$. Then we construct a tree decomposition $\mathcal{T}'$ of the ground program $ground'(P)$ as follows.

$\mathcal{T}'$ has the same tree structure as $\mathcal{T}$. Note that now the bags $\lambda'(N)$ consist of ground atoms occurring in $ground'(P)$. Let N be a leaf node with $\lambda(N) = dom(A)$ for some ground guard A. Then we insert into $\lambda'(N)$ the atom A plus all ground atoms B that occur in at least one rule $r \in ground'(P)$, s.t. A is the ground guard of r. By construction, we have $\lambda(N) = dom(A)$ and $dom(B) \subseteq dom(A)$. Hence, $dom(B) \subseteq \lambda(N)$ holds as well.

Suppose that now a ground atom B occurs in two distinct bags $\lambda'(N_1)$ and $\lambda'(N_2)$. In order to preserve the connectedness condition, we thus also have to add B to any bag $\lambda'(M)$ on the path from N_1 to N_2. As we have argued above, we know that $dom(B) \subseteq \lambda(N_1)$ and $dom(B) \subseteq \lambda(N_2)$ holds. Thus, by the connectedness condition on the tree decomposition $\mathcal{T}$, we may conclude that also $dom(B) \subseteq \lambda(M)$ holds for all nodes M on the path from N_1 to N_2. In other words, the following implication holds for every ground atom B: If $B \in \lambda'(N)$ then $dom(B) \subseteq \lambda(N)$. Hence, as a (very rough) upper bound on the width of $\mathcal{T}'$, we get $m * k^l$, where m is the number of predicate symbols in EDB and l is their maximum arity. □

We conclude this section with some heuristics which will normally lead to a much more efficient reduction from Datalog abduction to propositional abduction (even though the worst-case complexity from the proof of Theorem 12 is not affected).

1. Further simplification of $ground'(P)$. In the proof of Theorem 12 we were contented with deleting all ground rules where the ground guard is not contained in the EDB. Of course, this idea can be extended to any ground atoms with an EDB-predicate symbol. We thus apply the following simplification. Let r be a ground rule in $ground'(P)$ and let A be an atom occurring in the body of r. Moreover suppose that A has a predicate symbol from the EDB. Then we may apply the following simplifications: If $A \in EDB$, then A may be deleted from the rule (since it is clearly true). If $A \notin EDB$ then r may be deleted from $ground'(P)$ (since the body of r will never be true).

2. Simplification of the tree decomposition $\mathcal{T}'$. In the proof of Theorem 12 we started off with a tree decomposition $\mathcal{T}$ of EDB and constructed a tree decomposition $\mathcal{T}'$ of the ground logic program $ground'(P)$. A close inspection of the proof reveals that we could have started off with a tree decomposition of the subset $EDB' \subseteq EDB$ with

$$EDB' = \{A \in EDB \mid A \text{ is a ground guard}\}$$

Moreover, the tree decomposition will in general become much simpler if we first apply the above simplifications to

$ground'(P)$.

Example 13 Consider again the PAP $\mathcal{P}$ from Example 11. If we first ground P and then apply the above simplifications, then we get the following ground program (which is equivalent to $EDB \cup P$):

$\mathsf{one}(s).$ $\quad$ $\mathsf{zero}(s) \leftarrow \mathsf{faulty}(xor_1).$
$\mathsf{zero}(c_1).$ $\quad$ $\mathsf{one}(c_1) \leftarrow \mathsf{faulty}(and_1).$
$\mathsf{one}(sum) \leftarrow \mathsf{zero}(s).$ $\quad$ $\mathsf{zero}(sum) \leftarrow \mathsf{one}(s).$
$\mathsf{one}(sum) \leftarrow \mathsf{one}(s), \mathsf{faulty}(xor_2).$
$\mathsf{zero}(sum) \leftarrow \mathsf{zero}(s), \mathsf{faulty}(xor_2).$
$\mathsf{zero}(c_2) \leftarrow \mathsf{zero}(s).$ $\quad$ $\mathsf{zero}(c_2) \leftarrow \mathsf{one}(s), \mathsf{faulty}(and_2).$
$\mathsf{one}(c_2) \leftarrow \mathsf{one}(s).$ $\quad$ $\mathsf{one}(c_2) \leftarrow \mathsf{zero}(s), \mathsf{faulty}(and_2).$
$\mathsf{one}(carry) \leftarrow \mathsf{one}(c_1), \mathsf{zero}(c_2).$
$\mathsf{zero}(carry) \leftarrow \mathsf{one}(c_1), \mathsf{zero}(c_2), \mathsf{faulty}(or_1).$
$\mathsf{one}(carry) \leftarrow \mathsf{zero}(c_1), \mathsf{one}(c_2).$
$\mathsf{zero}(carry) \leftarrow \mathsf{zero}(c_1), \mathsf{one}(c_2), \mathsf{faulty}(or_1).$
$\mathsf{zero}(carry) \leftarrow \mathsf{one}(c_1), \mathsf{one}(c_2).$
$\mathsf{one}(carry) \leftarrow \mathsf{one}(c_1), \mathsf{one}(c_2), \mathsf{faulty}(or_1).$
$\mathsf{zero}(carry) \leftarrow \mathsf{zero}(c_1), \mathsf{zero}(c_2).$
$\mathsf{one}(carry) \leftarrow \mathsf{zero}(c_1), \mathsf{zero}(c_2), \mathsf{faulty}(or_1).$ □

Conclusions and Future Work

In (Gottlob, Pichler, & Wei 2006b) we presented new algorithms for several fundamental decision problems in database design, like the PRIMALITY problem.

In the current paper, we showed how these results can be carried over from the database area to AI and even further extended. We have thus established that the notion of treewidth can be fruitfully applied to many important real-world problems in both areas. Moreover, these new results provide yet another nice example for the often observed potential of cross-fertilization between the DB- and AI-area.

It should be noted that there are several ways of defining the treewidth of a hypergraph (notably: via the "primal graph" or via the "incidence graph"). Moreover, there are also other notions of treewidth (like directed treewidth). Striving for further fixed-parameter tractability results in the database and artificial intelligence field via these related concepts is an interesting target for future research.

References

Armstrong, W. 1974. Dependency structures of data base relationships. In *IFIP Congress*, 580–583.

Balsa, J.; Dahl, V.; and Lopes, J. P. 1995. Datalog grammars for abductive syntactic error diagnosis and repair. In *Proc. NLULP'95*.

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

Bonatti, P., and Samarati, P. 2000. Regulating service access and information release on the web. In *Proc. CCS'00*, 134–143.

Ceri, S.; Gottlob, G.; and Tanca, L. 1990. *Logic programming and databases*. Springer-Verlag New York, Inc.

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

de Kleer, J.; Mackworth, A. K.; and Reiter, R. 1992. Characterizing diagnoses and systems. *Artif. Intell.* 56(2-3):197–222.

Dowling, W. F., and Gallier, J. H. 1984. Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae. *J. Log. Program.* 1(3):267–284.

Eiter, T., and Gottlob, G. 1995. The complexity of logic-based abduction. *J. ACM* 42(1):3–42.

Fischer, P. C.; Jou, J. H.; and Tsou, D.-M. 1983. Succinctness in dependency systems. *Theor. Comput. Sci.* 24:323–329.

Friedrich, G.; Gottlob, G.; and Nejdl, W. 1990. Hypothesis classification, abductive diagnosis and therapy. In *Proc. Expert Systems in Engineering*, 69–78.

Gottlob, G.; Grädel, E.; and Veith, H. 2002. Datalog lite: a deductive query language with linear time model checking. *ACM Trans. Comput. Logic* 3(1):42–79.

Gottlob, G.; Leone, N.; and Scarcello, F. 2002. Hypertree decompositions and tractable queries. *J. Comput. Syst. Sci.* 64(3):579–627.

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

Gottlob, G.; Pichler, R.; and Wei, F. 2006b. Tractable database design through bounded treewidth. In *Proc. PODS'06*, 124–133.

Grohe, M. 1999. Descriptive and Parameterized Complexity. In *Proc. CSL'99*, volume 1683 of *LNCS*, 14–31.

Hermann, M., and Pichler, R. 2007. Counting complexity of propositional abduction. In *Proc. IJCAI-07*, 417–422.

Johnson, D. S. 1990. A catalog of complexity classes. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*. Elsevier Science Publishers B.V. (North-Holland). 67–161.

Koshutanski, H., and Massacci, F. 2003. An access control framework for business processes for web services. In *Proc. XMLSEC '03*, 15–24.

Lucchesi, C. L., and Osborn, S. L. 1978. Candidate keys for relations. *J. Comput. Syst. Sci.* 17(2):270–279.

Mannila, H., and Räihä;, K.-J. 1992. *The design of relational databases*. Addison-Wesley.

Minoux, M. 1988. LTUR: A Simplified Linear-Time Unit Resolution Algorithm for Horn Formulae and Computer Implementation. *Inf. Process. Lett.* 29(1):1–12.