In: Proceedings of the Thirteenth ACM SIGACT SIGMOD-SIGART Symposium on Principles of Database Systems (PODS-94), pp. 267–278, 1994.

Adding Disjunction to Datalog

(Extended Abstract)

Thomas Eiter **Georg Gottlob** **Heikki Mannila***

Christian Doppler Laboratory for Expert Systems, Information Systems Department

Technical University of Vienna, Paniglgasse 16, A-1040 Wien, Austria

(eiter|gottlob|mannila)@vexpert.dbai.tuwien.ac.at

Abstract

We study the expressive power and complexity of disjunctive datalog, i.e., datalog with disjunctive rule heads, under three different semantics: the minimal model semantics, the perfect models semantics, and the stable model semantics. We show that the brave variants of these semantics express the same set of queries. In fact, they precisely capture the complexity class Σ_2^P. The combined complexity of disjunctive datalog is shown to be $\mathrm{NEXPTIME}^{\mathrm{NP}}$-complete.

1 Introduction

Strong interest in enhancing logic programs by the capability of disjunction emerged in the logic programming and databases community. This interest is documented by a number of publications (cf. [LMR92]) and even workshops dedicated to this subject (cf. [IFI94]).

A disjunctive logic program (DLP) is a logic program that allows for disjunction in rule heads. Several semantics for such programs have been developed. Most of them are extensions of well-known semantics of logic programming (with and without negation). As plain logic programming corresponds to datalog, various extensions of datalog can be formulated for the different semantics of disjunctive logic programming.

In a previous paper [EG93], two of the authors analyzed the complexity of *propositional* disjunctive logic programming. The present paper is dedicated to the more difficult and involved task of analyzing both the complexity and the expressive power of *disjunctive datalog*.

The main questions we address in this paper are the following:

- ▷ How to define the precise semantics of disjunctive datalog based on different well-known semantics for DLPs ?
- ▷ What is the expressive power of these variants of disjunctive datalog? In particular, how is the expressive power affected by allowing disjunction in combination with other constructs like inequality $\neq$ and negation $\neg$?
- ▷ What is the complexity of disjunctive datalog, measured by data and expression complexity for the various formalisms ?
- ▷ Is disjunctive datalog besides its theoretical interest of practical relevance, i.e., do there exist queries arising in practice for which the power of disjunction in datalog is needed?

Our answers to these questions and the main contributions of this paper are summarized as follows:

- We give a precise definition of the syntax and the semantics of three major variants of disjunctive datalog. These semantics are: the minimal model semantics (*MM*) [Min82], also termed circumscription [McC80] and EGCWA [YH85]; the disjunctive stable model semantics (*SM*) [GL88, Prz91]; and the perfect model semantics (*PM*) [Prz88]. We consider both the brave and the cautious version of these semantics, for Boolean queries and for queries computing output relations.
- Plain datalog augmented with disjunction ($\mathrm{DATALOG}^{\vee}$) allows under all considered semantics (*MM*,*SM*,*PM*) expressing some Σ_2^P-hard queries, but not all queries in Σ_2^P. In fact, it does not allow expressing all queries computable in polynomial time. (This is shown by proving a weak form of preservation under supermodels.)
- We derive interesting new normal forms for second order Σ_k^1 sentences, $k \geq 2$, on finite structures. The salient feature of these normal forms is that there is

*On leave from the University of Helsinki, Finland. Current address: Department of Computer Science, University of Helsinki, P.O. Box 26, SF-00014 University of Helsinki, Finland. mannila@cs.helsinki.fi

SIGMOD/PODS 94 - 5/94 Minneapolis, Minnesota USA
© 1994 ACM 0-89791-639-5/94/0005..$3.50

no quantifier alternation in the first order part. Note that a similar normal form for Σ_1^1 sentences does not exist. The normal forms are useful for simplifying our expressiveness proofs.

- We show that disjunctive datalog with $\neq$ captures precisely the Σ_2^P queries under the brave version of all considered semantics. Thus, adding $\neq$ (inequality) to DATALOG$^\vee$ (giving DATALOG$^{\vee,\neq}$) suffices to gain full expressive power. The same holds for adding $\neg$ in case of both the *SM* and the *PM* semantics. In the more relevant cases, a symmetric result (Π_2^P-expressiveness) holds for cautious reasoning. In particular, the results show that for disjunctive datalog the *SM*-semantics and the *PM*-semantics express precisely the same set of queries; this is probably not true for DATALOG$^\neg$.
- We show that disjunctive datalog queries are Σ_2^P-complete in the data size and NEXPTIME$^{\text{NP}}$-complete in the program size under the brave version of all considered semantics. Notice that NP $\subset$ NEXPTIME; few natural NEXPTIME$^{\text{NP}}$-complete problems are known. In fact, there are Σ_2^P-hard resp. NEXPTIME$^{\text{NP}}$-hard queries defined by plain disjunctive datalog programs without $\neq$ and $\neg$.
- We demonstrate the practical relevance of disjunctive datalog by exhibiting queries (cf. the end of this section) which can be expressed in disjunctive datalog but not in standard DATALOG$^{\neq,\neg}$ unless the polynomial hierarchy collapses.

Our results complement and extend previous studies of important extensions of datalog and related query languages (cf. [CH85, Van87, KV90, ASV90, AV91, KP91, HS91, VVAG92] and many others [Kan90]). Due to space limitations, we will comment on these extensions in the full paper. We only note that all major versions of DATALOG$^{\neq,\neg}$ [KP91, Sch90, vRS91] express database queries in NP(resp. co-NP). Thus, disjunction adds expressive power (unless Σ_2^P = NP).

The following two examples of practical queries illustrate the need for higher expressive power than provided by DATALOG$^\neg$.

Example 1.1 (Strategic companies) Suppose a holding has a relational database with information about products and companies; a relation *Prod*(x, y) contains a tuple (p, c) if product p is produced by company c. The holding faces a crisis and is forced to sell some companies, but still wants to produce all goods. Company c is called *strategic* if it belongs to some minimal set of companies (wrt inclusion) that can produce all goods. Clearly, the managers want to avoid selling strategic companies. Thus, the database query if a certain company c is strategic is highly relevant.

Now consider the slightly more complex situation that accounts for company control. Several companies $c_1, c_2, \ldots, c_n$ may together have control over a company c, i.e., the holding cannot own all c_i's without holding c as well; this is stored in the relation *Cont*(x, y), which contains a tuple (c, c_i) for each i. Deciding if a company c is strategic in this realistic setting is Σ_2^P-complete (cf. [CEG94]). Hence, this query is not expressible in DATALOG$^{\neq,\neg}$ under stable semantics (which captures co-NP [Sch90] resp. NP) unless the polynomial hierarchy collapses.

Example 1.2 (Maximal eigenvector) The need for doing matrix computations in databases was pointed out by David Maier in his invited talk at PODS '93 [MV93]. Recall that an *eigenvector* of an $n \times n$ boolean matrix M is a Boolean n-vector e that $e = M \cdot e$. Eigenvectors have important applications in many areas such as control theory and game theory.

Impose a pre-order (reflexive, transitive binary relation) $\leq$ on Boolean n-vectors which can be decided in polynomial time, for all $n \geq 1$, and consider computing maximal eigenvectors. Given a Boolean matrix M, deciding whether M has a maximal eigenvector with first component 0, order $\leq$ fixed, can be shown to be Σ_2^P-complete. Thus, if M is stored in a relational database, a query deciding this problem is not expressible in DATALOG$^{\neq,\neg}$ under stable model semantics. However, by our results the problem can be solved with disjunctive datalog.

2 Preliminaries

For the required concepts of complexity theory, we refer to [Joh90], and for disjunctive logic programming to [LMR92]. For any disjunctive logic program (DLP) P (i.e., a logic program with negation where the head is a disjunction $a_1 \vee \cdots \vee a_n$ of atoms), we denote by HB_P its Herbrand base, by $MM(P)$ the set of its minimal Herbrand models, and by P^U the grounded version of P with respect to universe U. All considered DLPs are assumed to be function-free.

A *relational scheme* over a domain *Dom* is a (possibly empty) list $\mathbf{R} = R_1, \ldots, R_k$ of relation symbols R_i, $i = 1, \ldots, k$ of arity $a(R_i) \geq 0$. We assume that *Dom* is fixed and countable. An *instance* of $\mathbf{R}$ (*relational database* over $\mathbf{R}$) is a finite structure $D = (U, R_1, \ldots, R_k)$ of a finite universe $U \subseteq Dom$ and relations $R_i \subseteq D^{a(R_i)}$. (As common, we use relation symbols also to denote relation instances.) $\mathcal{D}(\mathbf{R})$ denotes the set of all instances of $\mathbf{R}$. By $U(D)$ we denote the universe of D and by $D[\mathbf{R}']$ the projection of D to $\mathbf{R}'$ for any subscheme $\mathbf{R}'$ of $\mathbf{R}$.

A *database property over* $\mathbf{R}$ is a predicate $\mathcal{P}$ on $\mathcal{D}(\mathbf{R})$ which associates with each $D \in \mathcal{D}(\mathbf{R})$ a truth value $\mathcal{P}(D)$ from $\{true, false\}$ and which is closed under isomorphisms.

A *database mapping* or *query* is a (partial) recursive function $q : \mathcal{D}(\mathbf{R}) \rightarrow \mathcal{D}(\mathbf{S})$, i.e., which maps instances D of $\mathbf{R}$ (intuitively, the input relations) to instances $q(D)$ of $\mathbf{S}$ (the output relations), where $\mathbf{R}$ and $\mathbf{S}$ are disjoint, such that $U(D) = U(q(D))$ and q is invariant under isomorphisms.The pair $(\mathbf{R}, \mathbf{S})$, denoted by $IO(q)$, is called the *input/output scheme of* q.

Recognizability of a query q is defined in terms of the following *query output tuple (QOT) problem*: Given $D \in \mathcal{D}(\mathbf{R})$ and a tuple $\mathbf{t}$ matching the format of $S_i \in \mathbf{S}$, decide if $q(D)[S_i] \models S_i(\mathbf{t})$. Query q is $\mathbf{C}$-*recognizable*, $\mathbf{C}$ a complexity class, iff the QOT-problem is in $\mathbf{C}$ (cf. [Gur88]). Similarly, a database property $\mathbf{P}$ is $\mathbf{C}$-*recognizable* (or, *in* $\mathbf{C}$) iff deciding $\mathbf{P}(D) = \mathit{true}$ is in $\mathbf{C}$.

A query q is *Boolean* if $\mathbf{S}$ is a single 0-ary relation symbol, i.e., a propositional letter P; we say that such a P is *true* if P (as a relation) consists of the empty tuple (); otherwise, P is *false* (in that case, $P = \emptyset$). A Boolean query corresponds to a database property in the obvious way.

A *query language* is a set $\mathcal{L}$ of query expressions E and a function μ defining the semantics of $\mathcal{L}$, such that for each $E \in \mathcal{L}$, $\mu(E)$ is a query. The *expressive power* $E(\mathcal{L})$ *of* $\mathcal{L}$ is the class of all queries definable by query expressions from $\mathcal{L}$, i.e., $E(\mathcal{L}) = \{q : q = \mu(E),\ E \in \mathcal{L}\}$.

As well-known, second order definability over finite structures is closely connected to computability within the polynomial hierarchy $PH = \Sigma_1^P \cup \Sigma_2^P \cup \cdots$ (Σ_1^P = NP) [Sto77, Lyn82]. For vocabulary σ denote by $\Sigma_k^1(\sigma)$ the second order sentences of the form $(\exists \mathbf{S}_1)(\forall \mathbf{S}_2)\cdots(Q\mathbf{S}_k)\varphi(\mathbf{S}_1, \ldots, \mathbf{S}_k)$, where the quantifiers alternate and φ is first order; we write Σ_k^1 if σ is understood.

Proposition 2.1 [Fag74, Sto77, Lyn82] A database property $\mathcal{P}$ over $\mathbf{R}$ is in Σ_k^P, $k \geq 1$, iff there exists a $\Sigma_k^1(\mathbf{R})$-sentence Ψ such that for each $D \in \mathcal{D}(\mathbf{R})$, $D \models \Psi$ iff $\mathcal{P}(D) = \mathit{true}$.

3 Disjunctive datalog

3.1 Syntax

A *disjunctive datalog rule* is a clause of the form

$$a_1 \vee \cdots \vee a_n \leftarrow t_1, \cdots, t_m, \qquad 1 \leq n, \quad 0 \leq m, \quad (1)$$

where the a_i's are atoms $S(x_1, \ldots, x_l)$ forming the *head* of the clause, and the t_j's are terms of the form $P(x_1, \ldots, x_l)$, $\neg P(x_1, \ldots, x_a)$, equality atoms $x_i = x_j$, or inequality atoms $x_i \neq x_j$ that constitute the *body* of the clause, where S resp. P are relations and all x_k are variables (cf. [KP91]); we allow that literals involve 0-ary relation symbols (i.e., propositional letters). The formalism can be easily extended for constants.

A DATALOG$^{\vee,\neq,\neg}$ *program* π is a finite collection of clauses of form (1). Let $RS(\pi)$ denote the relations occurring in π. We assume that π has attached an *input/output schema* $IO(\pi)$, which is a pair $(\mathbf{R}, \mathbf{S})$ such that $\mathbf{R}$ are the *extensional database relations* (i.e., those relations which can appear only in the body), and $\mathbf{S}$ is a *subset* of the *intensional relations* (i.e., those relations which appear in the head of at least one rule). Intuitively, the relations in $\mathbf{R}$ are the input relations and the relations in $\mathbf{S}$ are the output relations of π. The relations in $RS(\pi) - \mathbf{RS}$ are referred to as *auxiliary relations*.

We define a family DATALOG^X, $X \subseteq \{\vee, \neq, \neg\}$, of sublanguages of DATALOG$^{\vee,\neq,\neg}$ as follows. DATALOG (i.e. $X = \emptyset$) allows only for clauses with no disjunction ($n = 1$), negation $\neg$ only in front of input relations, and no atoms $x_i = x_j$, $x_i \neq x_j$; $\vee \in X$ (resp. $\neq$, $\neg \in X$) means that disjunction in heads (i.e. $n > 1$) (resp. $x_i = x_i$ and $x_i \neq x_j$, $\neg P(x_1, \ldots, x_l)$ for all P) is allowed, cf. [KV90]. For example, DATALOG$^\vee$, which we refer to as *plain disjunctive datalog*, allows disjunction in the head but no form of negation except in front of input predicates.

3.2 Semantics

There are several approaches in logic programming to capture the meaning of DLPs (cf. [LMR92]). We consider the minimal model semantics [Min82] (also termed circumscription [McC80] or EGCWA [YH85]), the (disjunctive) stable model semantics [GL88, Prz91], and the perfect model semantics [Prz88], which are among the most well-known disjunctive semantics. In [BF87] a semantics of disjunctive datalog (with stratified negation) was proposed using default logic; it is equivalent to the stable semantics. We formulate these semantics for full DATALOG$^{\vee,\neq,\neg}$ programs, which implicitly also provides definitions for the sublanguages.

With each DATALOG$^{\vee,\neq,\neg}$ program π and input database D, we associate a grounded DLP P_π^D. This program consists of all ground clauses C' which can be obtained from any clause C of $\pi^{U(D)}$, i.e., the ground instantiation of π (seen as DLP) from $U(D)$, by cancelling each literal l such that $D \models l$ and l is an atom $a = b$, $a \neq b$, or l involves an input relation of π. Notice that in P_π^D no $=$, $\neq$ and no literal of an input relation of π occurs. We define models of π in terms of Herbrand models of P_π^D. In the sequel, by model we always mean Herbrand model.

Definition 3.1 Let π be a DATALOG$^{\vee,\neq,\neg}$ program and D an input database for π. The *minimal (resp. stable, perfect) models of* π *on* D, $MM(\pi, D)$ (resp. $SM(\pi, D)$, $PM(\pi, D)$), are the minimal models (resp. the stable models, perfect models) of the DLP P_π^D.

The stable models of a grounded DLP P are the interpretations I of P such that I is a minimal model of the Gelfond-Lifschitz transform P^I [GL88, Prz91]; for the definition of perfect models, cf. [Prz88].

Since a disjunctive datalog program π has multiple minimal (resp. stable, perfect) models in general, the result of the computed query can be defined as either the union or the intersection of those models. The former approach (union) is usually referred to as *brave (or credulous) reasoning* and the latter (intersection) as *cautious (or skeptical) reasoning*. More formally, let $IO(\pi) = (\mathbf{R}, \mathbf{S})$ and let $\mathcal{S}$ stand for any of MM, SM, PM:

brave semantics: $q^{\pi}_{b,\mathcal{S}}$ denotes the query with I/O scheme ($\mathbf{R}$,$\mathbf{S}$) which maps $D \in \mathcal{D}(\mathbf{R})$ into the union of all $\mathcal{S}$-models of π on D, i.e.,

$$\mathbf{t} \in q^{\pi}_{b,\mathcal{S}}(D)[S_i] \quad \text{iff} \quad S_i(\mathbf{t}) \in \bigcup_{M \in \mathcal{S}(\pi,D)} M,$$

for each tuple $\mathbf{t} \in U(D)^{a(S_i)}$ and each S_i from $\mathbf{S}$;

cautious semantics: $q^{\pi}_{c,\mathcal{S}}$ denotes the query with input/output scheme ($\mathbf{R}$,$\mathbf{S}$) which maps $D \in \mathcal{D}(\mathbf{R})$ into the intersection of all $\mathcal{S}$-models of π on D, i.e.,

$$\mathbf{t} \in q^{\pi}_{c,\mathcal{S}}(D)[S_i] \quad \text{iff} \quad S_i(\mathbf{t}) \in \bigcap_{M \in \mathcal{S}(\pi,D)} M,$$

for each $\mathbf{t} \in U(D)^{a(S_i)}$ and each S_i from $\mathbf{S}$.

We denote by $\text{DATALOG}^{\vee,\neq,\neg}_{b,\mathcal{S}}$ (resp. $\text{DATALOG}^{\vee,\neq,\neg}_{c,\mathcal{S}}$) the query language of $\text{DATALOG}^{\vee,\neq,\neg}$ programs π with meaning $q^{\pi}_{b,\mathcal{S}}$ (resp. $q^{\pi}_{c,\mathcal{S}}$). For example, $\text{DATALOG}^{\vee,\neq,\neg}_{b,SM}$ denotes $\text{DATALOG}^{\vee,\neq,\neg}$ with brave stable model semantics. The notation for sublanguages is analogous.

The following relation between models of the semantics from above follows easily from the definitions and the well-known coincidence of minimal, perfect, and stable models of a $\neg$-free disjunctive program [Prz88].

Lemma 3.1 For every $\text{DATALOG}^{\vee,\neq}$ program π and input database D, $MM(\pi, D) = SM(\pi, D) = PM(\pi, D)$.

Hence, the query languages $\text{DATALOG}^{\vee,\neq}_{b,\mathcal{S}}$ (resp. $\text{DATALOG}^{\vee,\neq}_{c,\mathcal{S}}$) for $\mathcal{S} = MM, SM, PM$ coincide.

In this paper, we will disregard the cautious variant of the MM-semantics (c, MM) and restrict our attention to the brave variant (b, MM). The reason is that a well-known theorem states that a positive fact is in all minimal models iff it is in all models. Thus, the cautious MM-semantics does not provide model selection, and falls back to logical formalisms which are well-understood (cf. FN-queries in [CH85]). For the other semantics (SM,PM), the cautious variant is precisely symmetric to the brave variant if $\neg$ is allowed. For instance, the Boolean $\text{DATALOG}^{\vee,\neg}_{b,SM}$ queries express the Σ^P_2 database properties and the Boolean $\text{DATALOG}^{\vee,\neg}_{c,SM}$ queries express the Π^P_2 database properties. Therefore, we will not explicitly consider cautious SM-semantics (resp. cautious PM-semantics) here.

4 Limited capability of plain disjunctive datalog

We start with considering the effect of adding disjunction to (plain) DATALOG. The expressive power increases a lot: $\text{DATALOG}^{\vee}_{b,MM}$ (and $\text{DATALOG}^{\vee}_{b,SM}$, $\text{DATALOG}^{\vee}_{b,PM}$ as well) can express some Σ^P_2-hard queries (cf. Section 7). Notice that $\text{DATALOG}_{b,MM}$, which coincides with classical DATALOG plus negation of input relations, is limited to a strict subclass of the queries computable in polynomial time. However, $\text{DATALOG}^{\vee}_{b,MM}$ does not express all Σ^P_2-recognizable queries, and can in fact even not express all polynomial time computable database properties.

We show that no $\text{DATALOG}^{\vee,\neg}$ program with brave minimal model semantics is able to decide if the universe of the input database is even. This can be easily derived from the following lemma, which states a weak form of monotonicity for this query language.

Lemma 4.1 Let π be a $\text{DATALOG}^{\vee,\neg}$ program with $IO(\pi) = (\emptyset, W)$, where W is an arbitrary 0-ary predicate. Let $D_1, D_2 \in \mathcal{D}(\emptyset)$ (i.e., $D_1 = (U_1), D_2 = (U_2)$) such that $U_1 \subseteq U_2$ and $|U_1| \geq 1$. Then, $W \in M_1$ for some $M_1 \in MM(\pi, D_1)$ only if $W \in M_2$ for some $M_2 \in MM(\pi, D_2)$.

Proof. (Sketch) Note that, since π has no input relations, $P^{D_1}_{\pi} = \pi^{U_1}$ and $P^{D_2}_{\pi} = \pi^{U_2}$, hence $MM(\pi, D_1) = MM(\pi^{U_1})$ and $MM(\pi, D_2) = MM(\pi^{U_2})$. In what follows, let $P_1 = \pi^{U_1}$ and $P_2 = \pi^{U_2}$.

It is sufficient to show that the claim holds for $U_2 = U_1 \cup \{a\}$ where $a \notin U_1$. The result then follows by a simple induction argument.

Let $U_2 = U_1 \cup \{a\}$, $a \notin U_1$. Assume that $W \in M$ for $M \in MM(P_1)$. Let c be an arbitrary constant from U_1, and denote for any formal object X (formula, rule, term, etc) by $X[a/c]$ the object resulting from X by uniformly replacing a with c. Let $M^{a/c} = \{L \in HB_{P_2} : L[a/c] \in M\}$. Of course, $M^{a/c}$ is an interpretation of P_2. One can show that $M^{a/c}$ is a model of P_2. There must exist a minimal model $M' \subseteq M^{a/c}$ of P_2. Clearly, $M' \cap HB_{P_1}$ is a model of P_1. Since $M' \cap HB_{P_1} \subseteq M$, but M is already minimal, it holds that $M' \cap HB_{P_1} = M$. Since $W \in M$, also $W \in M'$. Since $M' \in MM(P_2)$, we are done. □

Theorem 4.2 A query deciding if the universe of a database is even is not expressible in $\text{DATALOG}^{\vee,\neg}_{b,MM}$.

Proof. Assume that query $q^{\pi}_{b,MM}$ defined by the $\text{DATALOG}^{\vee,\neg}$ program π with $IO(\emptyset, W)$ decides if the universe of input database D is even. Let $D = (U) \in \mathcal{D}(\emptyset)$ be such that $|U| > 0$ is even. Then π computes W true on D. By Lemma 4.1, π also computes W true on $D' = (U \cup \{a\})$, where a is not in U. However, $|U \cup \{a\}|$ is odd. Contradiction. □

Corollary 4.3 A query deciding if the universe of a database is even is not expressible in DATALOG$^{\vee}_{b,SM}$ and not expressible in DATALOG$^{\vee}_{b,PM}$.

Analogous results hold for the cautious semantics. Note that for nondisjunctive datalog, nonexpressiveness results have been recently shown by using infinitary logic (cf. [KV90, KV92]). It would be interesting if similar techniques can also be applied to disjunctive datalog.

5 Useful normal forms for Σ^1_2-sentences on finite structures

In this section, we present new normal forms for Σ^1_2 sentences over finite structures. In particular, (2) refines the normal form $(\exists\mathbf{S})(\forall\mathbf{T})(\exists\mathbf{x})(\forall\mathbf{y})\varphi(\mathbf{x},\mathbf{y})$, where φ is quantifier-free first order, for Σ^1_2 sentences (cf. [KP91]) over all structures, by taking into account that only finite structures are considered. The new normal forms appear to be helpful for deriving the result of Σ^P_2-expressiveness of DATALOG$^{\vee,\neq}_{b,MM}$ in the next section.

Theorem 5.1 Let σ be a (finite) vocabulary and Φ a $\Sigma^1_2(\sigma)$ sentence.

(i) There exists a $\Sigma^1_2(\sigma)$ sentence of the form

$$(\exists\mathbf{S})(\forall\mathbf{T}Z)[(\forall\mathbf{y})(Z(\mathbf{y}))\vee(\exists\mathbf{x})(\vartheta_1(\mathbf{x})\vee\cdots\vee\vartheta_k(\mathbf{x}))] \quad (2)$$

where the ϑ_i's are conjunctions of literals, which is equivalent to Φ over all finite structures.

(ii) Let $\sigma'=\sigma\cup\{F,S,L\}$, where F and L, unary, and S, binary, are new predicate symbols. There exists a $\Sigma^1_2(\sigma')$ sentence Φ' of the form

$$(\exists\mathbf{S})(\forall\mathbf{T})(\exists\mathbf{x})(\vartheta_1(\mathbf{x})\vee\cdots\vee\vartheta_k(\mathbf{x})) \quad (3)$$

where the ϑ_i's are conjunctions of literals in which F,L, and S occur only positively, such that for every finite structure D for σ, we have $D\models\Phi$ iff $D'\models\Phi'$, for any extension D' of D to σ' in which F,S, and L are interpreted as the relations first, successor, and last of a linear order on the universe.

Proof. (Sketch) We first show (ii).
Applying the method in [KP91] one can easily show that every $\Sigma^1_2(\sigma)$ sentence is equivalent to a $\Sigma^1_2(\sigma)$ sentence

$$(\exists\mathbf{S})(\forall\mathbf{T})(\exists\mathbf{x})(\forall\mathbf{y})\varphi(\mathbf{x},\mathbf{y}). \quad (4)$$

Assume that $F(x)$, $S(x,y)$, and $L(x)$ are the predicates which describe the first element, the successor relation, and the last tuple of some linear order $<$ on the universe. The predicates $F^n(\mathbf{y}), S^n(\mathbf{y},\mathbf{y}')$, and $L^n(\mathbf{y})$ that describe the first tuple, the successor ($\mathbf{y}'$ is the successor of $\mathbf{y}$) and the last tuple of the lexicographic order $<^n$ derived from $<$ on all n-tuples over the universe can be easily described using F,S and L as follows: let $F^1(x)=F(x)$, $S^1(x,y)=S(x,y)$, $L^1(x)=L(x)$ and for $n>1$,

$$\begin{aligned} F^n(\mathbf{x}) &\equiv F(x_1)\wedge F^{n-1}(\mathbf{x}'),\\ S^n(\mathbf{x},\mathbf{y}) &\equiv (x_1=y_1\wedge S^{n-1}(\mathbf{x}',\mathbf{y}'))\vee(S(x_1,y_1)\wedge\\ &\qquad L^{n-1}(\mathbf{x}')\wedge F^{n-1}(\mathbf{y}')),\\ L^n(\mathbf{y}) &\equiv L(y_1)\wedge L^{n-1}(\mathbf{y}'),\end{aligned}$$

where $\mathbf{x}=(x_1,\ldots,x_n)$, $\mathbf{x}'=(x_2,\ldots,x_n)$, $\mathbf{y}=(y_1,\ldots,y_n)$ and $\mathbf{y}'=(y_2,\ldots,y_n')$. Thus, $F^n(\mathbf{x})$, $S^n(\mathbf{x},\mathbf{y})$, and $L^n(\mathbf{x})$ can be described by disjunctions of conjunctions of equality atoms $x_i=y_i$ and atoms from F, S, and L.

Using F^n, S^n and L^n, the universal first order quantifier in (4) can be eliminated as follows. Define the formula Ψ as

$$\begin{aligned}(\forall Z)(\exists\mathbf{xy})[\,&F^n(\mathbf{y})\wedge\varphi(\mathbf{x},\mathbf{y})\wedge\neg Z(\mathbf{x},\mathbf{y})\,]\\ \vee(\exists\mathbf{xy})[\,&Z(\mathbf{x},\mathbf{y})\wedge L^n(\mathbf{y})\,]\\ \vee(\exists\mathbf{xyy}')[\,&Z(\mathbf{x},\mathbf{y})\wedge\neg Z(\mathbf{x},\mathbf{y}')\wedge\\ &S^n(\mathbf{y},\mathbf{y}')\wedge\varphi(\mathbf{x},\mathbf{y}')\,],\end{aligned}$$

where Z is a new predicate variable. Intuitively, the formula is false if Z consists of the union of $\{(\mathbf{a},\mathbf{b}_1),\ldots,(\mathbf{a},\mathbf{b}_{l_{\mathbf{a}}})\}$ for all $\mathbf{a}$, where $\mathbf{b}_1,\ldots,\mathbf{b}_{l_{\mathbf{a}}}$ is the maximum initial segment of the order $<^n$ such that $\varphi(\mathbf{a},\mathbf{b}_i)$ is true, $1\le i\le l_{\mathbf{a}}$, and each of those segments is strict. We claim that $\Psi\equiv(\exists\mathbf{x})(\forall y)\varphi(\mathbf{x},\mathbf{y})$.

Assume that for any finite structure D, $D\not\models(\exists\mathbf{x})(\forall\mathbf{y})\varphi(\mathbf{x},\mathbf{y})$, i.e., $D\models(\forall\mathbf{x})(\exists\mathbf{y})\neg\varphi(\mathbf{x},\mathbf{y})$. Let $min(\mathbf{a})$ be the first $\mathbf{b}'$ such that $D\not\models\varphi(\mathbf{a},\mathbf{b}')$; notice that $min(\mathbf{a})$ is well-defined. Define $Z_0=\{(\mathbf{a},\mathbf{b}):\mathbf{b}$ is ordered before $min(\mathbf{a})\}$. We check that $(D,Z_0)\not\models A$, where A is the first order part of Ψ. First

$$(D,Z_0)\not\models(\exists\mathbf{xy})[\,F^n(\mathbf{y})\wedge\varphi(\mathbf{x},\mathbf{y})\wedge\neg Z(\mathbf{x},\mathbf{y})\,],$$

because if $(D,Z_0)\models\varphi(\mathbf{a},\mathbf{b})\wedge F^n(\mathbf{b})$, then by definition $(\mathbf{a},\mathbf{b})\in Z_0$. Second,

$$(D,Z_0)\not\models(\exists\mathbf{xy})[\,Z(\mathbf{x},\mathbf{y})\wedge L^n(\mathbf{y})\,];$$

for, if $(D,Z_0)\models Z(\mathbf{a},\mathbf{b})$, $\mathbf{b}$ is ordered by definition of Z_0 before $min(\mathbf{a})$, hence $(D,Z_0)\not\models L^n(\mathbf{b})$. Third,

$$\begin{aligned}(D,Z_0)\not\models(\exists\mathbf{xyy}')[\,&Z(\mathbf{x},\mathbf{y})\wedge\neg Z(\mathbf{x},\mathbf{y}')\wedge\\ &S^n(\mathbf{y},\mathbf{y}')\wedge\varphi(\mathbf{x},\mathbf{y}')\,];\end{aligned}$$

Assume to the contrary that $(D,Z_0)\models Z(\mathbf{a},\mathbf{b})\wedge\neg Z(\mathbf{a},\mathbf{b}')\wedge S^n(\mathbf{b},\mathbf{b}')\wedge\varphi(\mathbf{a},\mathbf{b}')$ for $\mathbf{a},\mathbf{b}$, and $\mathbf{b}'$. By the definition of Z_0, it follows that $\mathbf{b}'=min(\mathbf{a})$. Since $D\models\neg\varphi(\mathbf{a},min(\mathbf{a}))$, a contradiction is reached. It follows that $(D,Z_0)\not\models A$; hence, $D\not\models\Psi$.

Now assume that $D\models(\exists\mathbf{x})(\forall\mathbf{y})\varphi(\mathbf{x},\mathbf{y})$, i.e., $D\models(\forall\mathbf{y})\varphi(\mathbf{a},\mathbf{y})$ for some $\mathbf{a}$. Consider any Z, and let $\mathbf{b}_1$ be

the first and the $\mathbf{b}_l$ be the last tuple in the order $<^n$. There are three cases.
Case 1: $(\mathbf{a}, \mathbf{b}_1) \notin Z$. Then, $(D, Z) \models F^n(\mathbf{b}_1) \wedge \varphi(\mathbf{a}, \mathbf{b}_1) \wedge \neg Z(\mathbf{a}, \mathbf{b}_1)$; hence, $(D, Z) \models A$.
Case 2: $(\mathbf{a}, \mathbf{b}_l) \in Z$. Then, $(D, Z) \models Z(\mathbf{a}, \mathbf{b}_l) \wedge L^n(\mathbf{b}_l)$; hence, $(D, Z) \models A$.
Case 3: $(\mathbf{a}, \mathbf{b}_1) \in Z$ and $(\mathbf{a}, \mathbf{b}_l) \notin Z$. Hence, there is a $\mathbf{b}$ such that $(\mathbf{a}, \mathbf{b}) \notin Z$ and $(\mathbf{a}, \mathbf{b}') \in Z$ for each $\mathbf{b}'$ that is ordered before $\mathbf{b}$. Let $\mathbf{b}'$ be the predecessor of $\mathbf{b}$. (Note that $\mathbf{b}'$ exists.) Then,

$$(D, Z) \models Z(\mathbf{a}, \mathbf{b}') \wedge \neg Z(\mathbf{a}, \mathbf{b}) \wedge S^n(\mathbf{b}', \mathbf{b}) \wedge \varphi(\mathbf{a}, \mathbf{b});$$

hence, $(D, Z) \models A$. $\diamond$

Consequently, $(D, Z) \models A$. It follows that $D \models \Psi$. The claim is proved.

Thus, if F, S, and L are provided, (4) is equivalent to

$$(\exists \mathbf{S})(\forall \mathbf{T} Z)(\exists \mathbf{z})\mu(\mathbf{z}), \qquad (5)$$

where $(\exists \mathbf{z})\mu(\mathbf{z})$ is the first order part of Ψ transformed into prenex normal form. Now $\mu(\mathbf{z})$ can be easily rewritten as a disjunction of conjunctions of literals, in which F, S, and L occur only positively. Thus, part (ii) of the theorem follows.

We continue to show (i). Clearly, F, S and L can be characterized from $<$ by a first order sentence, which is equivalent to some second order sentence of the form

$$(\exists \mathbf{Q})(\forall R)[(\forall \mathbf{x})(R(\mathbf{x})) \vee (\exists \mathbf{u})\beta(\mathbf{u})], \qquad (6)$$

where $\beta(\mathbf{xy}) = \neg R(\mathbf{x}) \wedge \varphi(\mathbf{x}, \mathbf{y})$. To show this, every Σ^1_1 sentence is equivalent to a Σ^1_1 sentence of the form

$$(\exists \mathbf{S})(\forall \mathbf{x})(\exists \mathbf{y})\varphi(\mathbf{x}, \mathbf{y}), \qquad (7)$$

over the same vocabulary, cf. [KP91]; as easily checked, this formula is equivalent to

$$(\exists \mathbf{S})(\forall T)[(\forall \mathbf{x})(T(\mathbf{x})) \vee (\exists \mathbf{xy})(\neg T(\mathbf{x}) \wedge \varphi(\mathbf{x}, \mathbf{y}))], \qquad (8)$$

where T is a new predicate variable.

Consequently, the sentence (4) is equivalent to the existential closure of the conjunction of (5) and (6) with respect to $<, F, S$, and L, which is equivalent to

$$(\exists \mathbf{S} < FSL\mathbf{Q})(\forall R\mathbf{T}) \\ \{[(\forall \mathbf{x})(R(\mathbf{x})) \vee (\exists \mathbf{u})\beta(\mathbf{u})] \wedge (\exists \mathbf{z})\mu(\mathbf{z})\}. \qquad (9)$$

This sentence is in turn equivalent to

$$(\exists \mathbf{S} < FSL\mathbf{Q})(\forall R\mathbf{T}) \\ \{(\forall \mathbf{x})(R(\mathbf{x})) \vee [(\exists \mathbf{u})\beta(\mathbf{u}) \wedge (\exists \mathbf{z})\mu(\mathbf{z})]\}. \qquad (10)$$

Here $(\exists \mathbf{u})\beta(\mathbf{u}) \wedge (\exists \mathbf{z})\mu(\mathbf{z})$ can be rewritten to

$$(\exists \mathbf{uz})[\gamma_1(\mathbf{u}, \mathbf{z}) \vee \cdots \vee \gamma_m(\mathbf{u}, \mathbf{z})],$$

where each γ_i is a conjunction of literals. Hence, for (4) there exists an equivalent formula of the form of (2). Part (ii) of the theorem follows. $\square$

The theorem easily generalizes from Σ^1_2 sentences to Σ^1_k sentences for $k \geq 2$. Notice that it does not hold for $k = 1$, i.e., Σ^1_1 sentences. We remark that neither of the disjuncts in (2) can be eliminated; one can show that if the existential (resp. universal) formula is not present, the formula is preserved under submodels (resp. supermodels; cf. [CK73] for model preservation properties). It is currently not known if the form (2) is valid over all structures. Analogous normal forms for Π^1_k sentences follow.

6 Expressive power of disjunctive datalog plus $\neq$

In this section, we show that adding $\neq$ to plain disjunctive datalog under the brave minimal model semantics suffices to gain full expressive power for Σ^P_2.

We consider first Boolean queries and address then general queries. The proof that $\mathrm{DATALOG}^{\vee,\neq}_{b,MM}$ expresses Σ^P_2 utilizes the equivalent form 3 for Σ^1_2 sentences in Theorem 5.1. In order to do so, we first describe a $\mathrm{DATALOG}^{\vee,\neq}$ program $\pi_<$ that computes the relations F, S, and L under brave minimal semantics. A special 0-ary predicate *Good* will indicate in each minimal model whether F, S, and L have the intended interpretation.

First, we state an easy lemma on minimal models of DLPs, which is useful for showing that a subprogram P_1 of a DLP P behaves as desired under minimal models. Similar modularity properties of logic programs have been considered in [Sch90] and, for disjunctive logic programs, in [Dix92].

Definition 6.1 Let P_1 and P_2 be DLPs. We write $P_2 \triangleright P_1$ iff no predicate occurring in P_1 occurs in the head of any rule of P_2.

Lemma 6.1 Let P_1 and P_2 be $\neg$-free DLPs such that $P_2 \triangleright P_1$. Then,

(i) $\forall M \in MM(P_1 \cup P_2)$: $M \cap HB_{P_1} \in MM(P_1)$.

(ii) $MM(P_1 \cup P_2) = \bigcup_{M \in MM(P_1)} MM(M \cup P_2)$. $\square$

Program $\pi_<$ is as follows; $<$ is a binary relation, $\tilde{F}, \tilde{S}, \tilde{L}$ are intended to be the complements of F, S, and L respectively, in minimal models, and G is an auxiliary relation.

(1) $x < y \vee y < x \leftarrow x \neq y.$
(2) $x < z \leftarrow x < y,\ y < z.$
(3) $\tilde{S}(x, z) \leftarrow z < x.$
(4) $\tilde{S}(x, z) \leftarrow x < y, y < z.$
(5) $\tilde{S}(x, x) \leftarrow.$
(6) $S(x, y) \vee \tilde{S}(x, y) \leftarrow.$

(7) $\tilde{F}(x) \leftarrow y < x.$
(8) $F(x) \vee \tilde{F}(x) \leftarrow.$
(9) $\tilde{L}(x) \leftarrow x < y.$
(10) $L(x) \vee \tilde{L}(x) \leftarrow.$
(11) $G(x) \leftarrow F(x).$
(12) $G(y) \leftarrow G(x), S(x, y).$
(13) $Good \leftarrow G(x), L(x).$

Lemma 6.2 Let U be any finite universe. For every linear order $\prec$ on U, let $M^U_\prec$ be the interpretation of $\pi^U_<$ such that $\{Good, G(a) : a \in U\} \subseteq M^U_\prec$, $<$ is interpreted as $\prec$, and F, S, and L (resp. $\tilde{F}, \tilde{S}, \tilde{L}$) are interpreted as the relations describing the first element, the successor relation, and the last element of $\prec$ (resp. the complements of those relations). Then,

(i) $M^U_\prec \in MM(\pi^U_<)$, for each linear order $\prec$ on U;

(ii) if $M \in MM(\pi^U_<)$ such that $Good \in M$, then $M = M^U_\prec$ for some linear order $\prec$ on U.

Proof. (Sketch) Let P denote $\pi^U_<$. It is easy to see that $M^U_\prec$ is a model of P, and that no atoms can be removed from $M^U_\prec$ without loosing the model property. Hence, *(i)* follows. To show *(ii)*, let $M \in MM(P)$ such that $Good \in M$. (Thus, $U \neq \emptyset$.) Since M is minimal, it is easy to see that $\tilde{F}, \tilde{S}$ and $\tilde{L}$ are defined in M as the complement of F, S and L, respectively. Let P_1 consist of all instances of (1) (2) in P and P_2 of the remaining clauses of P. By Lemma 6.1, it follows easily that $<$ is interpreted in M as a linear order $\prec$ on U. Since F and $\tilde{F}$ etc are complementary in M, it follows from the clauses of P that F, S and L must be interpreted in M as subsets of the relations describing the first element, the successor, and last element for $\prec$, respectively. Since $Good \in M$, it follows that $G(a), L(a) \in M$ for some $a \in U$; it follows that M interprets L as relation describing the last element of $\prec$. By induction downwards the order $\prec$, we obtain that $G(a) \in M$ for each $a \in M$ and that M interprets S and F such that they describe the successor relation and the first element of $\prec$, respectively. It follows that $M = M^U_\prec$. □

The following lemma collects already known results on minimal, stable, and perfect models of propositional DLPs.

Lemma 6.3 [EG93] Let P be a propositional DLP and p be a propositional atom. Deciding if $p \in M$ for some $M \in \mathcal{S}(P)$ is Σ^P_2-complete for $\mathcal{S} = MM, SM, PM$.

Now we are ready, using the results from Section 5, to show that DATALOG$^{\vee,\neq}_{b,MM}$ can express all, and only, the Σ^P_2-recognizable queries.

Theorem 6.4 Let $\mathcal{S}$ be any of MM, SM, PM. The DATALOG$^{\vee,\neq,\neg}_{b,\mathcal{S}}$ queries are Σ^P_2-recognizable.

Proof. The QOT problem (cf. Section 2) for query $q^\pi_{b,\mathcal{S}}$ defined by the DATALOG$^{\vee,\neq,\neg}$ program π trivially reduces to deciding if a grounded atom a belongs to some $M \in \mathcal{S}(\pi, D) = \mathcal{S}(P^D_\pi)$. For fixed π, the size of $\pi^{U(D)}$ is polynomial in the size of the input database D of π, hence the grounded DLP P^D_π can be constructed in polynomial time. Thus the theorem follows from Lemma 6.3. □

Let us first note the expressiveness of Boolean queries, which is useful for deriving the expressiveness of queries with output relations.

Theorem 6.5 The Boolean DATALOG$^{\vee,\neq}_{b,MM}$ queries precisely capture the class of all database properties in Σ^P_2.

Proof. (Sketch) By Theorem 6.4, it remains to show that every Σ^P_2-recognizable database property $\mathcal{P}$ can be expressed by a Boolean DATALOG$^{\vee,\neq}_{b,MM}$ query.

For that, we describe a Boolean DATALOG$^{\vee,\neq}$ program π with $IO = (\mathbf{R}, W)$, where W is a 0-ary predicate, such that, for every $D \in \mathcal{D}(\mathbf{R})$, the query $q^\pi_{b,MM}$ computes W true on D iff $\mathcal{P}(D) = true$. From Proposition 2.1, we have that $\mathcal{P}$ is equivalent to some $\Sigma^1_2(\sigma)$ sentence Ψ. Let Ψ' be the $\Sigma^1_2(\sigma')$ sentence, $\sigma' = \mathbf{R} \cup \{F, S, L\}$, of the form $(\exists \mathbf{S})(\forall \mathbf{T})(\exists \mathbf{x})(\vartheta_1(\mathbf{x}) \vee \cdots \vee \vartheta_k(\mathbf{x}))$ as described in Theorem 5.1(*i*). Assume that $\mathbf{S} = S_1, \ldots, S_n$ and $\mathbf{T} = T_1, \ldots, T_m$. Let the DATALOG$^{\vee,\neq}$ program π_0 consist of the following clauses (arities of tuple variables are understood):

(1)	$S_i(\mathbf{x}) \vee \tilde{S}_i(\mathbf{x}) \leftarrow$,	for $i = 1, \ldots, n$;
(2)	$T_j(\mathbf{x}) \vee \tilde{T}_j(\mathbf{x}) \leftarrow$,	for $j = 1, \ldots, m$;
(3)	$T_j(\mathbf{x}) \leftarrow W$,	for $j = 1, \ldots, m$;
(4)	$\tilde{T}_j(\mathbf{x}) \leftarrow W$,	for $j = 1, \ldots, m$;
(5)	$W \leftarrow T_j(\mathbf{x}), \tilde{T}_j(\mathbf{x})$,	for $j = 1, \ldots, m$;
(6)	$W \leftarrow \eta_i(\mathbf{x})$,	for $i = 1, \ldots, k$;

In this program, the $\tilde{S}_i$ and $\tilde{T}_j$ are new relations of the arity of S_i and T_j, respectively; W is a new 0-ary relation. Further, for each $i = 1, \ldots, k$, $\eta_i(\mathbf{x}) = l'_{i,1}, \ldots, l'_{i,n_i}, Good$, where $\vartheta_i = l_{i,1} \wedge \cdots \wedge l_{i,n_i}$ and $l'_{i,j}$ is the literal $l_{i,j}$ where, in case, each $\neg S_h$ (resp. $\neg T_h$) is replaced by $\tilde{S}_h$ (resp. $\tilde{T}_h$).

Let $\mathbf{R}' = \mathbf{R} \cup \{F, S, L, Good\}$ and let $IO(\pi_0) = (\mathbf{R}', W)$. The following can be shown. For every $D' \in \mathcal{D}(\mathbf{R}')$ such that $Good$ is true and F, S, L describe the relations first, successor, and last for a linear order $\prec$ of the universe $U(D)$, there exists an $M \in MM(\pi_0, D')$ with $W \in M$ iff $D' \models \Psi'$, i.e., $\mathcal{P}(D'[\mathbf{R}]) = true$.

Now let π be the union of π_0 and $\pi_<$ from above. Let $D \in \mathcal{D}(\mathbf{R})$. Define that for each linear order $\prec$ on $U(D)$, $D_\prec \in \mathcal{D}(\mathbf{R}')$ is the database such that $D_\prec[\mathbf{R}] = D$, $Good$ is true, and F, S and L describe the relations first, successor and last for $\prec$. Then, by Lemmas 6.1 and 6.2

it follows that

$$\{M \in MM(\pi, D) : W \in M\} = \\ \{M_1 \cup M_2 : M_1 \in MM(\pi_0, D_\prec), W \in M_1, \\ M_2 = M_\prec^{U(D)}, \prec \text{ a linear order on } U(D)\} .$$

It follows that for every $D \in \mathcal{D}(\mathbf{R})$, the query $q^\pi_{b,MM}$ computes W true on D iff $\mathcal{P}(D) = \mathit{true}$. □

Corollary 6.6 The Boolean DATALOG$^{\vee,\neq}_{b,SM}$ (resp. DATALOG$^{\vee,\neq}_{b,PM}$) queries precisely capture the class of all database properties in Σ_2^P.

A similar result can be derived for DATALOG$^{\vee,\neg}$ queries under the brave *SM* (resp. *PM*) -semantics. Notice that the expressive power of DATALOG$^{\vee,\neg}$ queries under the *MM*-semantics is the same as for DATALOG$^{\vee}$ queries, since negative literals in the body can be moved to the head of a clause.

Theorem 6.7 The Boolean DATALOG$^{\vee,\neg}_{b,SM}$ (resp. DATALOG$^{\vee,\neg}_{b,PM}$) queries precisely capture the class of all database properties in Σ_2^P.

Proof. (Sketch) This follows from Corollary 6.6 and the fact that $\neq$ can be easily computed using negation in stable (resp. perfect) models. Let π be a DATALOG$^{\vee,\neq,\neg}$ program and let Neq, Eq be binary relations not occurring in π. Let π' be the program resulting after replacing in π each $x_i = x_j$ by $Eq(x_i, x_j)$, each $x_i \neq x_j$ by $Neq(x_i, x_j)$, and by adding the clauses $Neq(x, y) \leftarrow \neg Eq(x, y)$ and $Eq(x, x) \leftarrow$. It holds that for every $D \in \mathcal{D}(\mathbf{R})$, $SM(\pi, D) = \{M \cap HB_{P_\pi^D} : M \in SM(\pi', D)\}$ (resp. $PM(\pi, D) = \{M \cap HB_{P_\pi^D} : M \in PM(\pi', D)\}$); the result follows. □

Since Σ_2^P is closed under conjunction, it is not difficult to extend our results for Boolean queries to the case of general I/O-queries. However, proofs of such generalizations are rarely found in the literature. Therefore, we will give a proof here.

Theorem 6.8 A query q is Σ_2^P-recognizable iff it is expressible in DATALOG$^{\vee,\neq}_{b,MM}$.

Proof. (Sketch) By Theorem 6.4, it remains to show the if-direction.

Let $q : \mathcal{D}(\mathbf{R}) \to \mathcal{D}(\mathbf{S})$ be a Σ_2^P-recognizable query. Let $\mathcal{P}$ be the database property over $\mathbf{R}' = \mathbf{RS}$ defined as follows: for every $D' \in \mathcal{D}(\mathbf{R}')$, $\mathcal{P}(D') = \mathit{true}$ iff $D'[\mathbf{S}] \subseteq q(D'[\mathbf{R}])$, where $\subseteq$ is understood componentwise. ($\mathcal{P}$ is in fact a database property.) Note that for any instance D' of $\mathbf{R}'$, $\mathcal{P}(D')$ is equivalent to a conjunction of QOT-problems for q as follows:

$$\mathcal{P}(D') = \bigwedge_{S_i \in \mathbf{S}} \bigwedge_{\mathbf{t} \in D'[S_i]} (\, \mathbf{t} \in q(D'[\mathbf{R}])[S_i] \,).$$

Since q is Σ_2^P-recognizable, it follows that $\mathcal{P}$ is in Σ_2^P. Thusreftheo:ddlneq-b,mm, there exists a DATALOG$^{\vee,\neg}$ program π' where $IO(\pi') = (\mathbf{R}', W)$ for some 0-ary relation W such that $q^{\pi'}_{b,MM}$ computes W true over $D' \in \mathcal{D}(\mathbf{R}')$ iff $\mathcal{P}(D') = \mathit{true}$. We may assume that π' has exactly the form of the DATALOG$^{\vee,\neq}$-program in the proof of Theorem 6.5. Now let π be the DATALOG$^{\vee,\neq}$ program with $IO(\pi) = (\mathbf{R}, \mathbf{S})$ obtained by modifying π' as follows. For each $S_i \in \mathbf{S}$, add the rules

$$S_i(\mathbf{x}) \leftarrow W, S_i'(\mathbf{x}). \\ S_i'(\mathbf{x}) \vee \tilde{S}_i'(\mathbf{x}) \leftarrow \ .$$

where S_i' and $\tilde{S}_i'$ are new relations of the same arity as S_i, and replace each positive (resp. negative) occurrence of S_i in π with S_i' (resp. $\tilde{S}_i'$). Intuitively, the first rule allows derivation of S_i only if W is in the interpretation, and the second rule simulates all possible instances of S_i'.

For every $D \in \mathcal{D}(\mathbf{R})$, $q^\pi_{b,MM}$ essentially computes the union of all $M \in MM(\pi, D)$ with $W \in M$. It can be seen that those M correspond precisely to the $D' \in \mathcal{D}(\mathbf{R}')$ such that $D'[\mathbf{R}] = D$ and $\mathcal{P}(D') = \mathit{true}$, i.e., $D'[\mathbf{S}] \subseteq q(D'[\mathbf{R}])$, where each M corresponds to the D' such that M and D' are equivalent on $\mathbf{S}$. It follows that $q^\pi_{b,MM}(D) = q(D)$. □

Corollary 6.9 A query q is Σ_2^P-recognizable iff it is expressible in DATALOG$^{\vee,\neq}_{b,SM}$ iff it is expressible in DATALOG$^{\vee,\neq}_{b,PM}$ iff it is expressible in DATALOG$^{\vee,\neg}_{b,SM}$ iff it is expressible in DATALOG$^{\vee,\neg}_{b,PM}$.

Symmetric results hold for the cautious semantics.

7 Complexity of disjunctive datalog

The complexity of a query language is traditionally measured by data and expression (resp. data/ expression) complexity, cf. [Var82]. The *data* (resp. *expression*, *data/expression*) *complexity* of a query language $\mathcal{L}$ is defined as the complexity of the QOT-problem in Section 2 for the queries defined by expressions $E \in \mathcal{L}$, where E is part of the input and E (resp. D, neither E nor D) is *fixed*.

7.1 Data complexity

From Theorem 6.4, we have

Theorem 7.1 The data complexity of DATALOG$^{\vee,\neq,\neg}_{b,\mathcal{S}}$ queries is in Σ_2^P for $\mathcal{S} = MM, SM, PM$.

From the proof of Theorem 6.5 and Proposition 2.1, it follows that DATALOG$^{\vee,\neq}_{b,MM}$ queries are Σ_2^P-hard in the data size. In fact, this can be sharpened to plain disjunctive datalog queries defined by programs without $\neq$ and $\neg$. Thus, adding disjunction to pure Horn clauses allows one to express Σ_2^P-hard queries. The proof is via a program π for deciding whether an atom a occurs in some minimal model of a DLP program of propositional clauses $a_1 \vee a_2 \leftarrow t_1, t_2, t_3$ ([EG93]).

Theorem 7.2 The Boolean $\mathrm{DATALOG}^{\vee}_{b,MM}$ queries are Σ_2^P-hard in the data size, even if no negation occurs in query programs.

Proof. Let P be a DLP consisting of propositional clauses of the form

$$a_1 \vee a_2 \leftarrow t_1, t_2, t_3 \qquad (*)$$

where all a_1, a_2 and t_1, t_2, t_3 are not necessarily distinct atoms from propositional variables $x_1, \ldots, x_n$ and t, which is a constant for *true*. By [EG93, Theorem 3.1] it follows that, given P and an atom a, deciding if a occurs in some minimal model of P is Σ_2^P-hard.

We define a fixed $\mathrm{DATALOG}^{\vee}$ program π with $IO(\pi) = (CAT, W)$, where C is 5-ary, A and T are unary, and W is 0-ary, such that π solves this problem on an input database D which is constructed from P as follows. The universe U of D is the set $\{x_1, \ldots, x_n, t\}$.

The clauses of P are encoded in the relation C, which consists of the tuples $(a_1, a_2, t_1, t_2, t_3)$ for all clauses $(*)$ in P; the relations A and T are used to designate the atom a and the constant t, i.e. $A = \{a\}$ and $T = \{t\}$. In addition, π uses a unary auxiliary relation V which intuitively corresponds to a valuation of the atoms.

Program π consists of the following three clauses.

$$\begin{aligned} V(y) &\leftarrow T(y). \\ V(y_1) \vee V(y_2) &\leftarrow C(y_1, y_2, y_3, y_4, y_5), \\ & V(y_3), V(y_4), V(y_5). \\ W &\leftarrow A(y), V(y). \end{aligned}$$

It is easily checked that π has a minimal model on D in which W occurs (i.e., the query $q^{\pi}_{b,MM}$ computes W true on D) iff P has a minimal model in which a occurs. Since D is polynomial time constructible from P, the result follows. $\Box$

Corollary 7.3 The Boolean $\mathrm{DATALOG}^{\vee}_{b,SM}$ queries and $\mathrm{DATALOG}^{\vee}_{b,PM}$ queries are Σ_2^P-hard in the data size, even if no negation occurs in query programs.

7.2 Expression complexity

Intuitively, the expression complexity of disjunctive datalog is exponentially larger than the data complexity, since the size of the grounded DLP P^D_{π} for the $\mathrm{DATALOG}^{\vee,\neq,\neg}$ program π can be exponential in the size of D and π. In fact, the expression complexity of $\mathrm{DATALOG}^{\vee,\neq,\neg}_{b,MM}$ turns out to be $\mathrm{NEXPTIME}^{\mathrm{NP}}$-complete; even plain disjunctive datalog without $\neq$ and $\neg$ is $\mathrm{NEXPTIME}^{\mathrm{NP}}$-hard in the program size.

Theorem 7.4 The expression complexity of $\mathrm{DATALOG}^{\vee,\neq,\neg}_{b,\mathcal{S}}$ is in $\mathrm{NEXPTIME}^{\mathrm{NP}}$ for $\mathcal{S} = MM, SM, PM$.

Proof. (Sketch) P^D_{π} is polytime constructible from D and $\pi^{U(D)}$, whose size is clearly bounded by $n^n = 2^{n \log n}$, where n is the size of π plus D. By Lemma 6.3, deciding if $S_i(\mathbf{a}) \in M$ for some $M \in \mathcal{S}(P^D_{\pi})$ is nondeterministically possible with an NP oracle in time polynomial in the size P^D_{π}. Thus, deciding whether $S_i(\mathbf{a}) \in M$ for some $M \in \mathcal{S}(\pi, D)$ is possible by a nondeterministic Turing machine in single exponential time with respect to the size of π plus D by using an NP oracle. Thus, the QOT-problem (Section 2) is in $\mathrm{NEXPTIME}^{\mathrm{NP}}$. $\Box$

The proof of $\mathrm{NEXPTIME}^{\mathrm{NP}}$-hardness relies on the use of new complexity upgrade techniques that extend the results in [BLT92, PY85], applied to results in [Ste91], and the simulation of Boolean circuits by a $\mathrm{DATALOG}^{\vee}$ program without $\neq$ and $\neg$ under minimal model semantics. (Cf. [KP91] for a simulation in $\mathrm{DATALOG}^{\neg,\neq}$ using fixpoint semantics.)

Roughly, the proof consists in a reduction of a problem on graphs, co-CERT3COL [Ste91], in its succinct version co-CERT3COL$_S$ [BLT92], i.e., the problem input consists of a Boolean circuit C_w by which the bits of the standard problem input w (a binary string encoding the graph) can be computed. Using the complexity upgrade techniques, we show that co-CERT3COL$_S$ is $\mathrm{NEXPTIME}^{\mathrm{NP}}$-complete. This problem is reduced to evaluating Boolean queries q^{π}_{bMM} over a fixed database D_0, defined by $\mathrm{DATALOG}^{\vee}$ programs π which are constructed in polynomial time

It is well-known that the input representation (e.g., numbers in unary or binary notation) can have a drastic effect on the problem complexity. Such effects were investigated for the "succinct versions" of graph-theoretic problems in [PY85], where a graph whose vertices are elements of $\{0,1\}^n$ is represented by a Boolean circuit with $2n$ input bits; the circuit outputs 1 if and only if the input represents two vertices which are connected by an edge.

Formally, a Boolean circuit is a finite set of triples (gates) $\{g_i = (a_i, j, k) : 1 \leq i \leq t\}$, where $a_i \in \{in, and, or, not\}$ is the kind of the gate, and $g_j, g_k, j, k < i$ are the inputs of the gate, unless g_i is an *in*-gate, in which case, say, $j = k = 0$. For *not*-gates, $j = k$ (cf. [KP91]). As shown in [PY85] showed that the succinct problem setting often leads to an exponential complexity increase.

The results and techniques of [PY85] were generalized in [BLT92] to the succinct version of any decisional problem A encoded as a language over $\{0,1\}$. The succinct representation of a binary word w is a Boolean circuit that on input of a number i (in binary), outputs whether $i \leq |w|$ and in that case, the i-th bit of w ($|w|$

denotes the length of w). The succinct version A_S of A is as follows: Given a Boolean circuit C_w describing a word w, decide whether $w \in A$.

The following theorem shows how to obtain hardness results under polynomial time transformability ($\leq_m^p$) for the succinct problem version from hardness of the standard problem under polylogtime transformability ($\leq_m^{PLT}$). This theorem is a generalization of [BLT92, Theorem 5], where the theorem is formulated for logtime transformability ($\leq_m^{LT}$). Roughly, a $\leq_m^{PLT}$-transformation is a problem mapping f such that, given w and an integer i, the i-th bit of $f(w)$ is computable in polylogarithmic time; note that every such mapping defines a polynomial-time transformation. For problem A, let $long(A)$ be defined as follows: $w \in long(A)$ iff the binary expression of the length of w is in $\{1w : w \in A\}$.

Theorem 7.5 Let **C1** and **C2** be arbitrary complexity classes such that for every $A \in$ **C1**, $long(A) \in$ **C2**. Then, for every B, if B is hard for **C2** under $\leq_m^{PLT}$-transformation, then B_S is hard for **C1** under $\leq_m^p$-transformation.

Proof. (Sketch) The theorem can be easily shown along the lines of the proof of Theorem 5 in [BLT92] by using a generalization of the following Conversion Lemma in [BLT92]: If $A \leq_m^{LT} B$ then $A_S \leq_m^p B_S$. It can be seen, by carefully looking at the proof in [BLT92], that the Conversion Lemma remains valid if $A \leq_m^{LT} B$ is replaced by $A \leq_m^{PLT} B$.

To show the theorem, assume that B is hard for **C2** under $\leq_m^{PLT}$-transformation, and let $A \in$ **C1** be arbitrary. By hypothesis, $long(A) \in$ **C2** and therefore $long(A) \leq_m^{PLT} B$. By the generalized Conversion Lemma, $long(A)_S \leq_m^p B_S$. Furthermore, $A \leq_m^p long(A)_S$ ([BLT92, Lemma 4]); by transitivity, $A \leq_m^p B_S$, and thus B_S is $\leq_m^p$-hard for **C1**. □

The proof of Theorem 7.9 uses this complexity upgrade technique. It gives a polynomial time transformation of the succinct version co-CERT3COL$_S$ of the graph problem co-CERT3COL, to evaluating Boolean DATALOG$^{\vee}_{b,MM}$ queries over a fixed database. The query program involves a subprogram which simulates Boolean circuits by which the relations in the input structure encoding the graph can be computed.

Problem co-CERT3COL is as follows (cf. [Ste91]):

Instance of size n: a graph G on the vertices $\{0, 1, \ldots, n-1\}$ such that every edge is labeled with a disjunction of two literals, where each literal is over the Boolean variables $\{X_{i,j} : i, j = 0, \ldots, n-1\}$.
Yes-instance of size n: an instance G of size n such that for some truth value assignment t on the Boolean variables, the graph $t(G)$ obtained from G by including only those edges set true under t is not 3-colorable.

Stewart showed that the complementary problem CERT3COL (i.e., negated condition for Yes-instances) is complete for Π_2^P under projection translations. It can be easily seen that projection translatability implies $\leq_m^{PLT}$-transformability. Thus, co-CERT3COL is clearly complete for Σ_2^P under $\leq_m^{PLT}$-transformation. Applying the complexity upgrade theorem, we thus obtain

Theorem 7.6 co-CERT3COL$_S$ is complete for NEXPTIME$^{\mathrm{NP}}$ under $\leq_m^p$-transformation.

We next describe disjunctive dalaog programs for deciding co-CERT3COL and for simulating a Boolean circuit. The programs are used in the proof of Theorem 7.9.

labeled graph coloring program

Problem co-CERT3COL can be decided by a Boolean DATALOG$^{\vee}$ program π_{Col} in which no negation occurs using brave stable model semantics. For an instance of co-CERT3COL of size n, the input database of π_{Col} consists in a structure with universe $\{0, \ldots, n-1\}$ and certain relations encoding the labeled graph.

We may assume that co-CERT3COL is encoded using a binary relation E and 4-ary relations P and N, such that $E(x,y) \vee E(y,x)$ (resp. $P(x,y,u,v) \vee P(y,x,u,v)$, $N(x,y,u,v) \vee N(y,x,u,v)$) encodes that an edge between x and y exists (resp. the edge between x and y is labeled by $X_{u,v}$, $\neg X_{u,v}$).

Thus, the input relations of π_{Col} are E, N and P; the output relation of π_{Col} is the 0-ary predicate W. The program π_{Col} uses in addition binary relations V and $\tilde{V}$, where $V(u,v)$ means that $X_{u,v}$ has value *true*; $\tilde{V}$ is intended to be the complement of V. Furthermore, π_{Col} uses unary relations R, B, and G, which encode the color of a node. π_{Col} consists of the following clauses:

(1) $V(x,y) \vee \tilde{V}(x,y) \leftarrow$.
(2) $W \leftarrow R(x), G(x)$.
(3) $W \leftarrow R(x), B(x)$.
(4) $W \leftarrow G(x), B(x)$.
(5) $R(x) \vee G(x) \vee B(x) \leftarrow$.
(6) $C(x) \leftarrow W$.
(7) $W \leftarrow P(x,y,u,v), V(u,v), C(x), C(y)$. $C = R$,
(8) $W \leftarrow N(x,y,u,v), \tilde{V}(u,v), C(x), C(y)$. G, B.

Intuitively, clause (1) chooses a truth value for $X_{u,v}$. Dependent on that choice, W is computed true if the resulting graph $t(G)$ is not 3-colorable.

Lemma 7.7 Let D be the input database for π_{Col} for an instance I of co-CERT3COL. Then, π_{Col} has a minimal model on D that contains W iff I is a Yes-instance.

Boolean circuit program

Let $C = \{g_i : g_i = (a_i, j, l), 1 \leq i \leq t\}$ constitute a Boolean circuit that computes a k-ary relation R over $\{0,1\}$.

We outline a DATALOG$^{\vee}$ program π_C that simulates C using the fixed input database $D_0 = (\{0,1\}, \{1\}, \{0\})$. (A similar DATALOG$^{\vee,\neg}$-program using fixpoint semantics can be found in [KP91].)

The following relations occur in π_C: A 0-ary output relation W, unary input relations One and $Zero$ to emulate constants 1 and 0, and for each gate g_i a k-ary auxiliary relation G_i.

Informally, $G_i(\mathbf{x})$ states that on input of tuple $\mathbf{x}$ to C, the circuit computation sets the output of g_i to 1.

The program π_C consists of the following clauses. For each gate $g_i = (a_i, j, l)$, it contains the clause

(0) $G_i(\mathbf{x}) \leftarrow W;$

Depending on the type a_i of gate g_i, it contains additional clauses as follows.

$$
\begin{array}{lll}
a_i = in: & (1) & G_i(\mathbf{x}) \leftarrow One(x_j)\ . \\
 & (2) & W \leftarrow G_i(\mathbf{x}), Zero(x_j). \\[1ex]
a_i = not: & (3) & G_i(\mathbf{x}) \vee G_j(\mathbf{x}) \leftarrow\ . \\
 & (4) & W \leftarrow G_j(\mathbf{x}), G_i(\mathbf{x}). \\[1ex]
a_i = and: & (5) & G_i(\mathbf{x}) \leftarrow G_j(\mathbf{x}), G_l(\mathbf{x}). \\
 & (6) & G_j(\mathbf{x}) \leftarrow G_i(\mathbf{x}). \\
 & (7) & G_l(\mathbf{x}) \leftarrow G_i(\mathbf{x}). \\[1ex]
a_i = or: & (8) & G_i(\mathbf{x}) \leftarrow G_j(\mathbf{x}). \\
 & (9) & G_i(\mathbf{x}) \leftarrow G_l(\mathbf{x}). \\
 & (10) & G_j(\mathbf{x}) \vee G_l(\mathbf{x}) \leftarrow G_i(\mathbf{x}).
\end{array}
$$

Assume that g_t is the output gate of C. The following can be shown.

Lemma 7.8 π_C has a unique minimal model M on $D_0 = (\{0,1\}, \{1\}, \{0\})$, which has the properties that $W \notin M$ and that $G_t(\mathbf{t}) \in M$ iff C computes 1 on input vector $\mathbf{t}$, for each $\mathbf{t} \in \{0,1\}^k$.

From Theorem 7.6 and Lemmas 7.7 and 7.8, the main theorem of this section can be derived.

Theorem 7.9 The Boolean DATALOG$^{\vee}_{b,MM}$ queries are NEXPTIME$^{\mathrm{NP}}$-hard in the program size (resp. combined data and program size).

Proof. (Sketch) Let C_I be the Boolean circuit of an instance I of co-CERT3COL of size n. From C_I, Boolean circuits C_P and C_N are constructible in polynomial time that compute P and N, respectively, where each argument x, y, u, and v is encoded by a string of $\lceil \log n \rceil$ bits. The circuits output 1 if a tuple is contained in P resp. N and 0 otherwise; for any string representing m with $n \leq m < 2^{\lceil \log n \rceil}$, the output is 0. C_P and C_N can be simulated by DATALOG$^{\vee}$ programs π_{C_P} and π_{C_N} constructible in polynomial time over input database D_0 as shown above. Let G_t^P and G_t^N be the relations corresponding to the output gates g_t^P and g_t^N of C_P and C_N, respectively.

Program π_{Col} can be easily adapted to a variant π^n_{Col} where the elements from $\{0, \ldots, n-1\}$ are represented by binary strings of length $\lceil \log n \rceil$ and all occurrences of P, N are replaced by G_t^P and G_t^N, respectively, in time polynomial in $\log n$.

Let the program π_I with $IO(\pi_I) = (One, Zero; W)$ consist of the union of π^n_{Col}, π_{C_N} and π_{C_P}. It can be shown that π_I has on D_0 a minimal model that contains W, i.e., query $q^{\pi_I}_{b,MM}$ computes W true on D_0 iff the considered instance of co-CERT3COL is a Yes-instance. Since π_I can be constructed from C_I in polynomial time, the result follows. □

Corollary 7.10 The Boolean DATALOG$^{\vee}_{b,SM}$ queries and DATALOG$^{\vee}_{b,PM}$ queries are NEXPTIME$^{\mathrm{NP}}$-hard in the program size (resp. combined data and program size).

Symmetric results (membership and hardness for Π_2^P in the data size and for co-NEXPTIME$^{\mathrm{NP}}$ in the program size resp. the combined data and program size) hold for the cautious variants of *SM* and *PM*, with hardness for Boolean DATALOG$^{\vee,\neg}$ queries.

Acknowledgment We are indebted to P.A. Bonatti, Ph. Kolaitis, and J.S. Schlipf for several important hints and comments.

References

[ASV90] A. Abiteboul, E. Simon, and V. Vianu. Non-Deterministic Languages to Express Deterministic Transformations. In *Proceedings PODS-90*, pp. 218–229, 1990.

[AV91] S. Abiteboul and V. Vianu. Datalog Extensions for Database Updates and Queries. *Journal of Computer and System Sciences*, 43:62–124, 1991.

[BLT92] J.L. Balcázar, A. Lozano, and J. Torán. The Complexity of Algorithmic Problems on Succinct Instances. In R. Baeta-Yates and U. Manber, editors, *Computer Science*, pp. 351–377. Plenum Press, New York, 1992.

[BF87] N. Bidoit and C. Froidevaux. Minimalism Subsumes Default Logic and Circumscription in Stratified Logic Programming. In *Proceedings LICS-87*, pp. 89–97. IEEE Press, 1987.

[CEG94] M. Cadoli, T. Eiter, and G. Gottlob. Using Default Logic as a Query Language. In *Proceedings KR-94*, 1994. Forthcoming.

[CH85] A. Chandra and D. Harel. Horn Clause Queries and Generalizations. *Journal of Logic Programming*, 2:1–15, 1985.

[CK73] C. C. Chang and H. J. Keisler. *Model Theory.* North-Holland, 2nd edition, 1973.

[Dix92] Jürgen Dix. Classifying Semantics of Disjunctive Logic Programs. In *Proceedings JICSLP-92*, pp. 798–812, Washington DC, November 1992.

[EG93] T. Eiter and G. Gottlob. Complexity Aspects of Various Semantics for Disjunctive Databases. In *Proceedings PODS-93*, pp. 158–167, June 1993.

[Fag74] R. Fagin. Generalized First-Order Spectra and Polynomial-Time Recognizable Sets. In R. M. Karp, editor, *Complexity of Computation*, pp. 43–74. AMS, 1974.

[GL88] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proc. Fifth Logic Programming Symposium*, pp. 1070–1080, Cambridge Mass., 1988. MIT Press.

[Gur88] Y. Gurevich. Logic and the Challenge of Computer Science. In E. Börger, editor, *Trends in Theoretical Computer Science*, chapter 1. Computer Science Press, 1988.

[HS91] R. Hull and J. Su. On the Expressive Power of Database Queries with Intermediate Types. *Journal of Computer and System Sciences*, 43:219–267, 1991.

[IFI94] IFIP-GI. Workshop: *Disjunctive Logic Programming and Disjunctive Databases.* To be held at 13^{th} IFIP World Computer Congress, Hamburg, August 1994.

[Joh90] David S. Johnson. A Catalog of Complexity Classes. In van Leeuwen [van90], chapter 2.

[Kan90] P. Kanellakis. Elements of Relational Database Theory. In van Leeuwen [van90], chapter 17.

[KP91] Ph.G. Kolaitis and Ch. H. Papadimitriou. Why Not Negation By Fixpoint ? *Journal of Computer and System Sciences*, 43:125–144, 1991.

[KV90] Ph.G. Kolaitis and M. Vardi. On the Expressive Power of Datalog: Tools and a Case Study. In *Proceedings PODS-90*, pp. 61–71, 1990.

[KV92] Ph.G. Kolaitis and M. Vardi. Fixpoint Logic vs. Infinitary Logic in Finite-Model Theory. In *Proceedings LICS-92*, pages 61–71. IEEE Computer Science Press, 1992.

[LMR92] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming.* MIT Press, Cambridge, MA, 1992.

[Lyn82] J.F. Lynch. Complexity Classes and Theories of Finite Models. *Mathematical Systems Theory*, 15:127–144, 1982.

[McC80] J. McCarthy. Circumscription – A Form of Non-Monotonic Reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[Min82] J. Minker. On Indefinite Data Bases and the Closed World Assumption. In *Proceedings of the 6^{th} Conference on Automated Deduction (CADE)*, pp. 292–308, 1982.

[MV93] D. Maier and B. Vance. A Call to Order. In *Proceedings PODS-93*, pp. 1–16, 1993.

[Prz88] T. Przymusinski. On the Declarative and Procedural Semantics of Stratified Deductive Databases. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pp. 193–216. Morgan Kaufman, 1988.

[Prz91] T. Przymusinski. Stable Semantics for Disjunctive Programs. *New Generation Computing*, 9:401–424, 1991.

[PY85] C.H. Papadimitriou and M. Yannakakis. A Note on Succinct Representations of Graphs. *Information and Computation*, 71:181–185, 1985.

[Sch90] J.S. Schlipf. The Expressive Powers of Logic Programming Semantics. Technical Report CIS-TR-90-3, Computer Science Department, University of Cincinnati, 1990. *Proceedings PODS-90*, pp. 196–204. Full paper in *JCSS*, to appear.

[Ste91] I. Stewart. Complete Problems Involving Boolean Labelled Structures and Projection Transactions. *Journal of Logic and Computation*, 1(6):861–882, 1991.

[Sto77] L. J. Stockmeyer. The Polynomial-Time Hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[Van87] D. Van Gucht. On the Expressive Power of the Extended Relational Algebra for the Unnormalized Relational Model. In *Proceedings PODS-87*, pp. 302–312, 1987.

[van90] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science*. Elsevier Science Pub, 1990.

[Var82] M. Vardi. Complexity of relational query languages. In *Proceedings STOC-82*, pp. 137–146, 1982.

[vRS91] A. van Gelder, K.A. Ross, and J.S. Schlipf. The Well-Founded Semantics for General Logic Programs. *JACM*, 38(3):620–650, 1991.

[VVAG92] J. Van den Bussche, D. Van Gucht, M. Andries, and M. Gyssens. On the Completeness of Object Creating Query Languages. In *Proceedings FOCS-92*, pp. 372–379, 1992.

[YH85] A. Yahya and L. Henschen. Deduction in Non-Horn Databases. *Journal of Automated Reasoning*, 1(2):141–160, 1985.