# Cardinality Constraints in Disjunctive Deductive Databases

Dietmar Seipel[1] and Ulrich Geske[2]

[1] University of Würzburg, Dept. of Computer Science
Am Hubland, D – 97074 Würzburg, Germany
seipel@informatik.uni-wuerzburg.de

[2] GMD–FIRST Berlin
Kekuléstraße 7, D – 12489 Berlin, Germany
geske@first.gmd.de

**Abstract**

We will consider *cardinality constraints* of the form $M \hookrightarrow_\Theta K$, $\Theta \in \{=, \leq, \geq\}$, stating that "(exactly, at most, at least) $K$ elements out of a set $M$ have to be chosen". We will show how a set $\mathcal{C}$ of constraints can be represented by means of a positive–disjunctive deductive database $\mathcal{P}_\mathcal{C}$, such that the models of $\mathcal{P}_\mathcal{C}$ correspond to the solutions of $\mathcal{C}$. This allows for embedding cardinality constraints into applications dealing with *incomplete knowledge*.

We will also present a *sound calculus* represented by a definite logic program $\mathcal{P}_{cc_=}$, which allows for directly reasoning with sets of exactly–cardinality constraints. Reasoning with $\mathcal{P}_{cc_=}$ is very efficient, and it can be used for performance reasons before $\mathcal{P}_\mathcal{C}$ is evaluated. For obtaining *completeness*, however, $\mathcal{P}_\mathcal{C}$ is necessary, since we show the theoretical result that a sound and complete calculus for exactly–cardinality constraints does not exist.

**Keywords**

disjunctive logic programming, constraint logic programming, cardinality constraints, hyperresolution

## 1 Introduction

Practical applications of deductive databases ([1]) often require the use of extended features like *disjunctive information*, *aggregation operators*, *default negation*, *computation of cheapest models*; several extensions of deductive databases have been proposed for these reasons. Applications dealing with *cardinality constraints* may for instance arise in assignment problems, such as course planning, or in games. E.g. there may be rules like "every student has to take 3 of the courses in a given set $M$".

*Constraint logic programming* (CLP) ([4]) offers very efficient methods for solving problems involving cardinality constraints. But if cardinality constraints have to be mixed with other formalisms for representing incomplete knowldege, then it becomes necessary to embed them into a richer framework, e.g., disjunctive logic programming. It might for instance be necessary to represent generic incomplete information such as "every student has to finish two small labs or one big lab, except if he enrolled himself before a certain deadline".

Various *sound and complete calculi* are known from literature in relational databases, e.g. the calculus given by the well–known *Armstrong axioms* for deriving implied functional dependencies. In deductive databases, reasoning with the well–known consequence operator $\mathcal{T}_\mathcal{P}$ of van Emden and Kowalski (cf. [2]) is sound and complete for deriving all atomary consequences of a definite logic program $\mathcal{P}$; it is not necessary to derive other formulas, such as disjunctions of literals, as intermediate results. Similarly, we would like to *limit the search space* in a special calculus for cardinality constraints. We will show that cardinality constraints can be derived based on the disjunctive consequence operator $\mathcal{T}^s_{\mathcal{P}}$ of Minker and Rajasekar (cf. [5]), which generalizes $\mathcal{T}_{\mathcal{P}}$. But we will also show that, unfortunately, there exists no sound and complete calculus that is dealing with cardinality constraints as intermediate results only.

The rest of the paper is organized as follows. In Section 2 we present the theoretical background of disjunctive deductive databases and we define cardinality constraints. In Section 3 we are dealing with calculi for reasoning with cardinality constraints. Some experimental results obtained for reasoning with cardinality constraints based on disjunctive deductive databases are reported in Section 4. Finally, Section 5 presents some conclusions.

## 2 Basic Definitions and Notations

In this section we will define the concepts of disjunctive deductive databases and of cardinality constraints, and we will show how cardinality constraints can be expressed by disjunctive deductive databases.

### 2.1 Disjunctive Deductive Databases

**Syntax**

Given a first order language $\mathcal{L}$, a *disjunctive deductive database* $\mathcal{P}$ (cf. [3]) consists of logical inference rules of the form

$$\begin{aligned} r \;=\; & A_1 \vee \ldots \vee A_k \leftarrow \\ & B_1 \wedge \ldots \wedge B_m \wedge \\ & \mathit{not}\, C_1 \wedge \ldots \wedge \mathit{not}\, C_n, \end{aligned} \quad (1)$$

where $A_i$, $i \in \langle 1, k \rangle$, $B_i$, $i \in \langle 1, m \rangle$, and $C_i$, $i \in \langle 1, n \rangle$, are atoms in the language $\mathcal{L}$, $k, m, n \in I\!N_0$, and *not* is the negation–by–default operator.[1] A rule is called a fact if $m = n = 0$. The set of all *ground instances* of the rules and facts in $\mathcal{P}$ is denoted by $\mathit{gnd}\,(\mathcal{P})$. A rule (or database) is called *positive–disjunctive* if it does not contain default negation (i.e. $n = 0$). It is called *normal* if $k = 1$, and *definite* if $k = 1$ and $n = 0$. It is called a *denial rule* if $k = 0$. A rule $r$ of the form (1) is denoted for short as $r = \alpha \leftarrow \beta \wedge \mathit{not} \cdot \gamma$, where $\alpha = A_1 \vee \ldots \vee A_k$, $\beta = B_1 \wedge \ldots \wedge B_m$, and $\gamma = C_1 \vee \ldots \vee C_n$.[2] $\alpha$ is called the *head*, $\beta$ is called the *positive body*, and $\mathit{not} \cdot \gamma$ is called the *negative body* of $r$.

For positive–disjunctive deductive databases $\mathcal{P}$ the *dual version* (cf. [10]) can been defined: for a positive–disjunctive rule $r = A_1 \vee \ldots \vee A_k \leftarrow B_1 \wedge \ldots \wedge B_m$ the dual version

$$r^d = B_1 \vee \ldots \vee B_m \leftarrow A_1 \wedge \ldots \wedge A_k$$

is obtained by exchanging the head and the body atoms of $r$. The dual version of $\mathcal{P}$ is given by $\mathcal{P}^d = \{\, r^d \mid r \in \mathcal{P} \,\}$.

The Herbrand base $\mathit{HB}_{\mathcal{P}}$ of a disjunctive deductive database $\mathcal{P}$ contains all ground atoms over the language of $\mathcal{P}$. The *disjunctive Herbrand base* $\mathit{DHB}_{\mathcal{P}}$ of $\mathcal{P}$ consists of all ground disjunctions $A_1 \vee \ldots \vee A_k$, $k \in I\!N_0$, of atoms $A_i$, $i \in \langle 1, k \rangle$. The *negative Herbrand base* $\mathit{NHB}_{\mathcal{P}}$ of $\mathcal{P}$ consists of all ground disjunctions $\neg A_1 \vee \ldots \vee \neg A_k$, $k \in I\!N_0$, of negated atoms $\neg A_i$, $i \in \langle 1, k \rangle$. A negative disjunction can equivalently be represented by a denial rule $\leftarrow A_1 \wedge \ldots \wedge A_k$. A subset $S \subseteq \mathit{DHB}_{\mathcal{P}}$ ($S \subseteq \mathit{NHB}_{\mathcal{P}}$) is called a disjunctive (negative) Herbrand state.

**Semantics**

An *Herbrand interpretation* of $\mathcal{P}$ is given by a subset $I \subseteq \mathit{HB}_{\mathcal{P}}$. It defines a mapping $I \colon \mathit{HB}_{\mathcal{P}} \to \{\mathsf{true}, \mathsf{false}\}$ that assigns a truth value to each atom in $\mathit{HB}_{\mathcal{P}}$: $I(A) = \mathsf{true} \Leftrightarrow A \in I$.[3] $I$ *satisfies* of a ground rule $r = \alpha \leftarrow \beta \wedge \mathit{not} \cdot \gamma$, for short $I \models r$, if

$$I(\beta) \wedge \neg I(\gamma) = \mathsf{true} \;\Rightarrow\; I(\alpha) = \mathsf{true}.$$

$I$ is an *Herbrand model* of $\mathcal{P}$ if $I$ satisfies all ground instances $r \in \mathit{gnd}\,(\mathcal{P})$ of the elements of $\mathcal{P}$. $I$ is a *minimal model* of $\mathcal{P}$ if $I$ is an Herbrand model of $\mathcal{P}$ and there is no other Herbrand model $J$ of $\mathcal{P}$ such that $J \subsetneq I$. The set of all *minimal models* of $\mathcal{P}$ is denoted by $\mathcal{MM}(\mathcal{P})$.

For positive–disjunctive deductive databases $\mathcal{P}$, the semantics is given by $\mathcal{MM}(\mathcal{P})$, and a formula $\phi$ is a *consequence* of $\mathcal{P}$ under the minimal model semantics if $\phi$ holds in all minimal

[1] By $I\!N_+$ we denote the set $\{\, 1, 2, 3, \ldots \,\}$ of positive natural numbers, whereas $I\!N_0$ denotes the set $\{\, 0, 1, 2, \ldots \,\}$ of all natural numbers. $\langle\, n, m \,\rangle$ denotes the interval $\{\, n, n+1, \ldots, m \,\}$ of natural numbers.

[2] Note that $\gamma$ is a disjunction, and, according to De Morgan's law, $\mathit{not} \cdot \gamma$ is taken to be a conjunction.

[3] For $A_i \in \mathit{HB}_{\mathcal{P}}$, $i \in \langle 1, k \rangle$, and a connective $\otimes \in \{\vee, \wedge\}$ we define $I(\, A_1 \otimes \ldots \otimes A_k \,) = I(A_1) \otimes \ldots \otimes I(A_k)$. For $k = 0$, the empty disjunction (i.e. $\otimes = \vee$) evaluates to $\mathsf{false}$, whereas the empty conjunction (i.e. $\otimes = \wedge$) evaluates to $\mathsf{true}$.

models of $\mathcal{P}$. The set of all positive ground disjunctions that are consequences of $\mathcal{P}$ can be derived as the least fixpoint $lfp(\mathcal{T}_{\mathcal{P}}^{s})$ of the *disjunctive consequence operator*

$$\mathcal{T}_{\mathcal{P}}^{s} : 2^{D_{HB_{\mathcal{P}}}} \to 2^{D_{HB_{\mathcal{P}}}}$$

of Minker and Rajasekar (cf. [5]):

$$\begin{aligned}\mathcal{T}_{\mathcal{P}}^{s}(S) \;=\; \{\, & \alpha \vee \alpha_1 \vee \ldots \vee \alpha_m \in D_{HB_{\mathcal{P}}} \mid \\ & \exists\, \alpha \leftarrow B_1 \wedge \ldots \wedge B_m \in gnd(\mathcal{P}) : \\ & \forall\, i \in \langle 1, m \rangle : B_i \vee \alpha_i \in S \,\}.\end{aligned}$$

This operator $\mathcal{T}_{\mathcal{P}}^{s}$ is monotonic and continuous (cf. [2]), and thus it reaches its least fixpoint $lfp(\mathcal{T}_{\mathcal{P}}^{s})$ in at most $\omega$ steps.

**Example**

For the positive–disjunctive database $\mathcal{P} = \{\, a \leftarrow a_1 \wedge a_2 \,\}$ and the disjunctive Herbrand state $S = \{\, a_1 \vee b_1, a_2 \vee b_2 \,\}$ we can derive $\mathcal{T}_{\mathcal{P}}^{s}(S) = \{\, a \vee b_1 \vee b_2 \,\}$. The set of minimal models of $\mathcal{P}$ is $\mathcal{MM}(\mathcal{P}) = \{\, \{a_1, a_2, a\}, \{a_1, b_2\}, \{a_2, b_1\} \,\}$.

## 2.2 Cardinality Constraints

In this section we will define the concept of cardinality constraints and explain their relationship to disjunctive logic programming (DLP).

**Syntax**

Given a universe $H_B$ of elements, a subset $M \subseteq H_B$, and a natural number $K \in \langle 0, |M| \rangle$, a *cardinality constraint* is given by $M \hookrightarrow_{\Theta} K$, where $\Theta \in \{=, \leq, \geq\}$ is a comparison relation on $\mathbb{N}_0$. $M \hookrightarrow_{=} K$ is called an *exactly–cardinality constraint* (i.e. $\Theta =$ "$=$"), and it is called *trivial*, if $M = \emptyset$ and $K = 0$.

**Semantics**

An interpretation is a subset $I \subseteq H_B$. $I$ satisfies a cardinality constraint $c = M \hookrightarrow_{\Theta} K$, for short $I \models c$, if $|M \cap I| \,\Theta\, K$. I.e., $c$ means that for the cardinality $|M'|$ of the selected subset $M' = M \cap I$ of $M$ it must hold $|M'| \,\Theta\, K$.

Given a set $\mathcal{C}$ of cardinality constraints. $I$ is a *model* of $\mathcal{C}$, for short $I \models \mathcal{C}$, if $I$ satisfies all constraints $c \in M$. $\mathcal{C}$ implies a single cardinality constraint $c$, for short $\mathcal{C} \models c$, if for all models $I$ of $\mathcal{C}$ it holds $I \models c$.

A set $\mathcal{C}$ of exactly–cardinality constraints *covers* $c = M \hookrightarrow_{=} K$, for short $\mathcal{C} \rhd c$, if there exist constraints $M_i \hookrightarrow_{=} K_i \in \mathcal{C}$, $i \in \langle 1, k \rangle$, such that the sets $M_i$ are pairwise disjoint, $M = \cup_{i=1}^{k} M_i$, and $K = \sum_{i=1}^{k} K_i$. It can be shown that $\mathcal{C} \rhd c$ always implies $\mathcal{C} \models c$, and that $\mathcal{C} \rhd c$ if $c$ is trivial.

**Cardinality Constraints in DLP**

In disjunctive logic programming cardinality constraints can be expressed as follows:

$$\begin{aligned}&\mathcal{C}(M \hookrightarrow_{\leq} K) = \\ &\qquad \{\, \leftarrow A_1 \wedge \ldots \wedge A_{K+1} \mid \\ &\qquad\quad \{A_1, \ldots, A_{K+1}\} \subseteq M \,\}, \\ &\mathcal{C}(M \hookrightarrow_{\geq} K) = \\ &\qquad \{\, A_1 \vee \ldots \vee A_{|M|-K+1} \mid \\ &\qquad\quad \{A_1, \ldots, A_{|M|-K+1}\} \subseteq M \,\}, \\ &\mathcal{C}(M \hookrightarrow_{=} K) = \\ &\qquad \mathcal{C}(M \hookrightarrow_{\leq} K) \cup \mathcal{C}(M \hookrightarrow_{\geq} K).\end{aligned}$$

E.g., for $K = |M|$ it holds $\mathcal{C}(M \hookrightarrow_{\leq} K) = \emptyset$, and for $K = 0$ it holds $\mathcal{C}(M \hookrightarrow_{\geq} K) = \emptyset$, since these cardinality constraints do not pose any restrictions on a model.
For a singleton set $M = \{A_1\}$ we get

$$\begin{aligned}\mathcal{C}(\{A_1\} \hookrightarrow_{=} 0) &= \{\leftarrow A_1\}, \\ \mathcal{C}(\{A_1\} \hookrightarrow_{=} 1) &= \{A_1\},\end{aligned}$$

and for a set $M = \{A_1, A_2\}$ we get

$$\begin{aligned}&\mathcal{C}(M \hookrightarrow_{=} K) = \\ &= \begin{cases} \{\leftarrow A_1, \leftarrow A_2\}, & \text{if } K = 0, \\ \{A_1 \vee A_2, \leftarrow A_1 \wedge A_2\}, & \text{if } K = 1, \\ \{A_1, A_2\}, & \text{if } K = 2. \end{cases}\end{aligned}$$

In general, the sizes of the disjunctive deductive databases $\mathcal{C}(M \hookrightarrow_{\Theta} K)$ are:

$$\begin{aligned}|\,\mathcal{C}(M \hookrightarrow_{\leq} K)\,| &= \binom{|M|}{K+1}, \\ |\,\mathcal{C}(M \hookrightarrow_{\geq} K)\,| &= \binom{|M|}{|M|-K+1} \\ &= \binom{|M|}{K-1}, \\ |\,\mathcal{C}(M \hookrightarrow_{=} K)\,| &= \binom{|M|}{K+1} + \binom{|M|}{K-1}.\end{aligned}$$

The databases $\mathcal{C}(M \hookrightarrow_{\Theta} K)$ are always positive–disjunctive, they consist of ground formulas only, and it holds:

$$\begin{aligned}\mathcal{C}(M \hookrightarrow_{\leq} K) &\subseteq N_{HB_M}, \\ \mathcal{C}(M \hookrightarrow_{\geq} K) &\subseteq D_{HB_M}.\end{aligned}$$

The universe of $M \hookrightarrow_\Theta K$ coincides with the Herbrand universe of $\mathcal{C}(M \hookrightarrow_\Theta K)$, i.e. $H_B = H_{B_{\mathcal{C}(M \hookrightarrow_\leq K)}}$, and also their models coincide:

**Lemma 2.1** *For all $I \subseteq H_B$ it holds:*

$$I \models M \hookrightarrow_\Theta K \Leftrightarrow I \models \mathcal{C}(M \hookrightarrow_\Theta K).$$

**Proof:**

1. $\Theta =$ "$\leq$": Obviously, the set $\mathcal{C}(M \hookrightarrow_\leq K)$ of denial rules is not satisfied by an interpretation $I$ iff there exists some subset $\{A_1, \ldots, A_{K+1}\} \subseteq M$, such that the denial rule $\leftarrow A_1 \wedge \ldots \wedge A_{K+1}$ is not satisfied in $I$, i.e. such that $\{A_1, \ldots, A_{K+1}\} \subseteq I$. This precisely holds iff $|I \cap M| \geq K + 1$. Conversely, $I \models \mathcal{C}(M \hookrightarrow_\leq K)$, iff $|I \cap M| \leq K$, i.e. iff $I \models M \hookrightarrow_\leq K$.

2. $\Theta =$ "$\geq$": Obviously, the set $\mathcal{C}(M \hookrightarrow_\geq K)$ of denial rules is not satisfied by an interpretation $I$ iff there exists some subset $\{A_1, \ldots, A_{|M|-K+1}\} \subseteq M$, such that the fact $A_1 \vee \ldots \vee A_{|M|-K+1}$ is not satisfied in $I$, i.e. $\{A_1, \ldots, A_{|M|-K+1}\} \cap I = \emptyset$. This means that at least $|M| - K + 1$ elements of $M$ are not in $I$, i.e. $|M \setminus I| \geq |M| - K + 1$, which means that $|I \cap M| \leq K - 1$. Conversely, $I \models \mathcal{C}(M \hookrightarrow_\leq K)$, iff $|I \cap M| \geq K$, i.e. iff $I \models M \hookrightarrow_\geq K$.

3. $\Theta =$ "$=$": As a consequence of the previous two items, $I \models \mathcal{C}(M \hookrightarrow_= K)$, iff $I \models M \hookrightarrow_= K$. $\square$

Another possible encoding of $M \hookrightarrow_= K$ would be to represent the alternative models $I = \{ A_1, \ldots, A_K \} \subseteq H_B$ of $M \hookrightarrow_= K$ as conjunctive formulas

$$\phi_I = A_1 \wedge \ldots \wedge A_K \wedge \neg A_{K+1} \wedge \ldots \wedge \neg A_{|M|},$$

where $H_B \setminus I = \{ A_{K+1}, \ldots, A_{|M|} \}$. The conjunctive normal form of the disjunction of the $\phi_I$ yields a positive–disjunctive deductive database that is equivalent to $\mathcal{C}(M \hookrightarrow_= K)$. But it can be shown that $\mathcal{C}(M \hookrightarrow_= K)$ is the most compact representation of $M \hookrightarrow_\Theta K$ in DLP.

Given a set $\mathcal{C}$ of cardinality constraints, we denote the positive–disjunctive deductive database representing $\mathcal{C}$ by $\mathcal{P}_\mathcal{C} = \cup_{cc \in \mathcal{C}} \mathcal{C}(cc)$. It can be shown that $\mathcal{P}_\mathcal{C}$ always is non–recursive. Moreover, as a consequence of Lemma 2.1, the least fixpoints of the disjunctive consequence operators $\mathcal{T}^s_\mathcal{P}$ for $\mathcal{P} = \mathcal{P}_\mathcal{C}$ and its dual version $\mathcal{P}^d = \mathcal{P}^d_\mathcal{C}$ can tell whether a singleton cardinality constraint $\{A\} \hookrightarrow_= K$, $K \in \{0, 1\}$, is implied by $\mathcal{C}$.

**Lemma 2.2** *For a set $\mathcal{C}$ of cardinality constraints and an atom $A \in H_B$ it holds:*

$$(\mathcal{C} \models \{A\} \hookrightarrow_= 1) \Leftrightarrow A \in \mathit{lfp}(\mathcal{T}^s_{\mathcal{P}_\mathcal{C}}),$$
$$(\mathcal{C} \models \{A\} \hookrightarrow_= 0) \Leftrightarrow A \in \mathit{lfp}(\mathcal{T}^s_{\mathcal{P}^d_\mathcal{C}}).$$

In Section 4 we will especially be interested in singleton cardinality constraints. For arbitrary cardinality constraints, testing whether $\mathcal{C} \models M \hookrightarrow_= K$ could be done by checking whether the Herbrand state

$$\mathit{lfp}(\mathcal{T}^s_{\mathcal{P}_\mathcal{C}}) \cup (\mathit{lfp}(\mathcal{T}^s_{\mathcal{P}^d_\mathcal{C}}))^d$$

implies $M \hookrightarrow_= K$, which results in a check for subsumption. But computing the set of all implied cardinality constraints is very complex due to the size of the search space.

## 3 Reasoning with Cardinality Constraints

In this section we will present a definite logic program representing a *sound calculus* for reasoning with exactly–cardinality constraints, and we will show that there exists no definite logic program representing a sound and *complete* calculus.

### 3.1 A Sound Calculus for Cardinality Constraints

In the following we will present a logic program $\mathcal{P}_{cc_=} = \{ r_1, \ldots, r_7 \}$ representing a calculus of some *sound* inference rules for exactly–cardinality constraints, where the predicate symbols are written in infix notation:

$$\begin{aligned} r_1 \;=\;& M_2 \setminus M_1 \hookrightarrow_= K_2 - K_1 \leftarrow \\ & M_1 \hookrightarrow_= K_1 \wedge M_2 \hookrightarrow_= K_2 \wedge \\ & M_1 \subseteq M_2, \\ r_2 \;=\;& M_2 \setminus M_1 \hookrightarrow_= K_2 - K_1 \leftarrow \end{aligned}$$

$$
\begin{aligned}
& M_1 \hookrightarrow_= K_1 \wedge M_2 \hookrightarrow_= K_2 \wedge \\
& \mid M_2 \setminus M_1 \mid = K_2 - K_1, \\
r_3 \;=\; & M_2 \setminus M_1 \hookrightarrow_= K_2 - \mid M_2 \cap M_1 \mid \leftarrow \\
& M_1 \hookrightarrow_= \mid M_1 \mid \wedge M_2 \hookrightarrow_= K_2, \\
r_4 \;=\; & M_2 \cup M_1 \hookrightarrow_= |M_2 \cup M_1| \leftarrow \\
& M_1 \hookrightarrow_= \mid M_1 \mid \wedge M_2 \hookrightarrow_= \mid M_2 \mid, \\
r_5 \;=\; & M_2 \setminus M_1 \hookrightarrow_= K_2 \leftarrow \\
& M_1 \hookrightarrow_= 0 \wedge M_2 \hookrightarrow_= K_2, \\
r_6 \;=\; & M_2 \cup M_1 \hookrightarrow_= K_2 \leftarrow \\
& M_1 \hookrightarrow_= 0 \wedge M_2 \hookrightarrow_= K_2, \\
r_7 \;=\; & \{A\} \hookrightarrow_= 0 \leftarrow \\
& M \hookrightarrow_= 0 \wedge A \in M.
\end{aligned}
$$

Let us for example apply the calculus to the following set $\mathcal{C} = \{c_1, c_2, c_3\}$ :

$$
\begin{aligned}
c_1 &= \{a, b, c, d\} \hookrightarrow_= 1, \\
c_2 &= \{c, d, e\} \hookrightarrow_= 2, \\
c_3 &= \{d, e, f, g\} \hookrightarrow_= 2.
\end{aligned}
$$

From $c_1$ and $c_2$ the rule $r_2$ can conclude $c_4$, and rule $r_1$ can conclude $c_5$ from $c_4$ and $c_2$. From $c_5$ and $c_1$ we can conclude $c_6$ using $r_1$, and $c_7$ can be concluded from $c_4$ and $c_3$ using $r_1$:

$$
\begin{aligned}
c_4 &= \{e\} \hookrightarrow_= 1, \\
c_5 &= \{c, d\} \hookrightarrow_= 1, \\
c_6 &= \{a, b\} \hookrightarrow_= 0, \\
c_7 &= \{d, f, g\} \hookrightarrow_= 1.
\end{aligned}
$$

By applying the rule $r_7$ to $c_6$ we can conclude $c_8 = \{a\} \hookrightarrow_= 0$ and $c_9 = \{b\} \hookrightarrow_= 0$. It can be seen that for the special set $\mathcal{C}$ the calculus is able to derive all singleton cardinality constraints $\{A\} \hookrightarrow_= K$, $K \in \{0, 1\}$, such that $\mathcal{C} \models \{A\} \hookrightarrow_= K$ (i.e. $c_4$, $c_8$, and $c_9$).

It is easy to prove the soundness of $\mathcal{P}_{cc_=}$. Of course there exist inference rules for cardinality constraints with $\Theta \in \{\leq, \geq\}$ as well. E.g. rule $r_2$ can be modified by shortening its body:

$$
\begin{aligned}
r_2' \;=\; & M_2 \setminus M_1 \hookrightarrow_\geq K_2 - K_1 \leftarrow \\
& M_1 \hookrightarrow_= K_1 \wedge M_2 \hookrightarrow_= K_2.
\end{aligned}
$$

In the special case $\mid M_2 \setminus M_1 \mid = K_2 - K_1$, the cardinality constraint $M_2 \setminus M_1 \hookrightarrow_\geq K_2 - K_1$ obviously implies $M_2 \setminus M_1 \hookrightarrow_= K_2 - K_1$. An even stronger inference rule would be

$$
\begin{aligned}
r_2'' \;=\; & M_2 \setminus M_1 \hookrightarrow_\geq K_2 - K_1 \leftarrow \\
& M_1 \hookrightarrow_\geq K_1 \wedge M_2 \hookrightarrow_\leq K_2.
\end{aligned}
$$

In this paper, however, we will limit our attention to the restricted calculus for exactly–cardinality constraints given by $\mathcal{P}_{cc_=}$. The question arises, whether this calculus is also *complete* for inferring exactly–cardinality constraints, and – if not – whether $\mathcal{P}_{cc_=}$ can be extended by other rules to a larger definite logic program $\mathcal{P}'_{cc_=}$ that is sound and complete.

### 3.2 A Generic Example

For answering the questions above we will investigate the following generic set $\mathcal{C}_n$ of cardinality constraints.

$$
\begin{aligned}
\mathcal{C}_n \;=\; & \{\, \{a_1, \ldots, a_n\} \hookrightarrow_= n - 1 \,\} \cup \\
& \{\, \{a_i, b_i, c\} \hookrightarrow_= 2 \,|\, 1 \leq i \leq n \,\},
\end{aligned}
$$

where $a_i$, $b_i$, and $c$ are pairwise different atoms.

**Lemma 3.1** *For the cardinality constraints*

$$
\begin{aligned}
cc_i &= \{a_i, b_i\} \hookrightarrow_= 1, \;\; 1 \leq i \leq n, \\
cc_{n+1} &= \{c\} \hookrightarrow_= 1, \\
cc_{n+2} &= \{b_1, \ldots, b_n\} \hookrightarrow_= 1,
\end{aligned}
$$

*it holds*

$$
\mathcal{C}_n \models cc_i, \quad \textit{for all } i \in \langle 1, n+2 \rangle.
$$

**Proof:**

1. The cardinality constraint $\{c\} \hookrightarrow_= 1$ can be derived by applying the disjunctive consequence operator $\mathcal{T}^s_{\mathcal{P}}$ using the following subsets of $\mathcal{C}_n$:

   $$
   \begin{aligned}
   & \mathcal{C}(\{a_1, \ldots, a_n\} \hookrightarrow_\leq n - 1) = \\
   & \quad \{\, \leftarrow a_1 \wedge \ldots \wedge a_n \,\}, \\
   & \mathcal{C}(\{a_i, b_i, c\} \hookrightarrow_\geq 2) = \\
   & \quad \{\, a_i \vee b_i, a_i \vee c, b_i \vee c \,|\, i \in \langle 1, n \rangle \,\}.
   \end{aligned}
   $$

   Thus, for

   $$
   \begin{aligned}
   \mathcal{P} &= \{\, \leftarrow a_1 \wedge \ldots \wedge a_n \,\}, \\
   S &= \{\, a_i \vee c \,|\, i \in \langle 1, n \rangle \,\}
   \end{aligned}
   $$

   we can conclude $\mathcal{T}^s_{\mathcal{P}}(S) = \{\, c \,\}$, which implies $\{c\} \hookrightarrow_= 1$.

2. By applying the rule $r_1$ of $\mathcal{P}_{cc_=}$ to $\{c\} \hookrightarrow_= 1$ and $\{a_i, b_i, c\} \hookrightarrow_= 2$ we can derive $\{a_i, b_i\} \hookrightarrow_= 1$.

3. The constraint $\{b_1, \ldots, b_n\} \hookrightarrow_= 1$ obviously follows from the constraints $\{a_i, b_i\} \hookrightarrow_= 1$ and $\{a_1, \ldots, a_n\} \hookrightarrow_= n-1$. This can also be derived by applying the disjunctive consequence operator $\mathcal{T}^s_\mathcal{P}$ for suitable $\mathcal{P}$ and $S$. From

$$\begin{array}{rcl} \mathcal{P} & = & \{ \leftarrow a_1 \wedge \ldots \wedge a_n \}, \\ S & = & \{ a_i \vee b_i \mid i \in \langle 1, n \rangle \}, \end{array}$$

we can derive $\mathcal{T}^s_\mathcal{P}(S) = \{ b_1 \vee \ldots \vee b_n \}$, which coincides with $\mathcal{C}(\{b_1, \ldots, b_n\} \hookrightarrow_\geq 1)$. We also use the following subsets of $\mathcal{C}_n$:

$$\begin{array}{l} \mathcal{C}(\{a_1, \ldots, a_n\} \hookrightarrow_\geq n-1) = \\ \quad \{ a_{i_1} \vee a_{i_2} \mid i_1, i_2 \in \langle 1, n \rangle, \, i_1 \neq i_2 \}, \\ \mathcal{C}(\{a_i, b_i\} \hookrightarrow_\leq 1) = \\ \quad \{ \leftarrow a_i \wedge b_i \mid i \in \langle 1, n \rangle \}. \end{array}$$

From the dualized rules in

$$\begin{array}{rcl} \mathcal{P} & = & \{ \leftarrow a_{i_1} \wedge a_{i_2} \mid \\ & & \quad i_1, i_2 \in \langle 1, n \rangle, \, i_1 \neq i_2 \}, \\ S & = & \{ a_i \vee b_i \mid i \in \langle 1, n \rangle \}, \end{array}$$

we can derive $\mathcal{T}^s_\mathcal{P}(S) = \{ b_{i_1} \vee b_{i_2} \mid i_1, i_2 \in \langle 1, n \rangle, \, i_1 \neq i_2 \}$, which is the dual version of $\mathcal{C}(\{b_1, \ldots, b_n\} \hookrightarrow_\leq 1) = \{ \leftarrow b_{i_1} \wedge b_{i_2} \mid i_1, i_2 \in \langle 1, n \rangle, \, i_1 \neq i_2 \}$. □

### 3.3 The Incompleteness Result

Now we will answer the questions of whether the cardinality constraints $cc_i$, $i \in \langle 1, n+2 \rangle$, can be inferred from $\mathcal{C}_n$ using the calculus given by $\mathcal{P}_{cc_=}$ alone, i.e. without using the disjunctive consequence operator $\mathcal{T}^s_\mathcal{P}$, and – more generally – whether there exists a definite logic program $\mathcal{P}'_{cc_=}$ (e.g. an extension of $\mathcal{P}_{cc_=}$) representing a sound and *complete* calculus for inferring exactly–cardinality constraints at all. Note that $\mathcal{P}'_{cc_=}$ should be finite and all of its rules can only contain a finite number of atoms of the form $M \hookrightarrow_= K$ in their bodies. Thus, there exists a rule $r_{max} \in \mathcal{P}'_{cc_=}$ with a maximal number $n_{max}$ of atoms $M \hookrightarrow_= K$ in its body. In the following, however, we will show that for all strict subsets $\mathcal{C}' \subsetneq \mathcal{C}_n$ it holds

$$\begin{array}{rcl} \mathcal{C}' & \not\models & \{a_i, b_i\} \hookrightarrow_= 1, \; 1 \leq i \leq n, \\ \mathcal{C}' & \not\models & \{c\} \hookrightarrow_= 1, \\ \mathcal{C}' & \not\models & \{b_1, \ldots, b_n\} \hookrightarrow_= 1. \end{array}$$

We will even prove a much stronger result, namely: there exists no non–trivial cardinality constraint $M \hookrightarrow_= K$ such that $\mathcal{C}' \models M \hookrightarrow_= K$ and $\neg(\mathcal{C}' \rhd M \hookrightarrow_= K)$.

**Theorem 3.2** *For all strict subsets $\mathcal{C}' \subsetneq \mathcal{C}_n$, where $n \geq 2$, and all non–trivial cardinality constraints $M \hookrightarrow_= K$ it holds:*

$$(\mathcal{C}' \models M \hookrightarrow_= K) \Rightarrow (\mathcal{C}' \rhd M \hookrightarrow_= K).$$

**Proof:**
$\mathcal{C}_n$ has got precisely $n$ models $I_n$, $i \in \langle 1, n \rangle$, given by

$$I_i = \{ a_1, \ldots, a_{i-1}, \, b_i, \, a_{i+1}, \ldots, a_n, \, c \}.$$

Assume that $M \hookrightarrow_= K$ is a cardinality constraint, such that $M$ is minimal w.r.t. set inclusion amoung all cardinality constraints satisfying

$$(\mathcal{C}' \models M \hookrightarrow_= K) \wedge \neg(\mathcal{C}' \rhd M \hookrightarrow_= K).$$

Obviously, the index sets

$$\begin{array}{rcl} \Lambda_{a,b} & = & \{ i \in \langle 1, n \rangle \mid a_i \in M, \, b_i \in M \}, \\ \Lambda_{a,-} & = & \{ i \in \langle 1, n \rangle \mid a_i \in M, \, b_i \notin M \}, \\ \Lambda_{-,b} & = & \{ i \in \langle 1, n \rangle \mid a_i \notin M, \, b_i \in M \}, \\ \Lambda_{-,-} & = & \{ i \in \langle 1, n \rangle \mid a_i \notin M, \, b_i \notin M \}, \end{array}$$

form a partition of $\langle 1, n \rangle$. Let $\lambda_{\mu,\nu} = |\Lambda_{\mu,\nu}|$. E.g., for $M = \{ a_1, b_1, a_3, b_4, a_5 \}$ we get

$$\begin{array}{ll} \Lambda_{a,b} = \{1\}, & \Lambda_{a,-} = \{3, 5\}, \\ \Lambda_{-,b} = \{4\}, & \Lambda_{-,-} = \{2\}, \end{array}$$

and $\lambda_{a,b} = \lambda_{-,b} = \lambda_{-,-} = 1$, $\lambda_{a,-} = 2$.

Since $\mathcal{C}' \subsetneq \mathcal{C}_n$, there exists an element $C \in \mathcal{C}_n \setminus \mathcal{C}'$. In the following we will consider an arbitrary such element $C$.

1. If $C = \{a_k, b_k, c\} \hookrightarrow_= 2$, for some index $k \in \langle 1, n \rangle$, then $\mathcal{C}'$ has got the following $n+2$ additional models, which are not models of $\mathcal{C}$:
   - For $i = k$: $J^1_k = I_k \setminus \{ b_k \}$,

     $$\begin{array}{l} J^2_k = \{ a_1, b_1, \ldots, a_{k-1}, b_{k-1}, \\ \qquad a_{k+1}, b_{k+1}, \ldots, a_n, b_n \}, \end{array}$$

     and $J^3_k = J^2_k \cup \{ b_k \}$.

- For $i \neq k$: $J_i^1 = I_i \cup \{\, b_k \,\}$.

In the following three steps we will prove that $b_k \notin M$, $\lambda_{-,-} \neq n$, and $\lambda_{a,-} \neq n$.

If $b_k \in M$, then

$$|\, M \cap J_k^2 \,| = |\, M \cap J_k^3 \,| - 1,$$

i.e. $M \hookrightarrow_= K$ cannot hold both in $J_k^2$ and in $J_k^3$, i.e. $\mathcal{C} \not\models M \hookrightarrow_= K$. Thus, it holds $b_k \notin M$.

If $\lambda_{-,-} = n$, then $M = \{c\}$, since $M \neq \emptyset$. Then for all $i \in \langle\, 1, n \,\rangle$ it holds

$$|\, M \cap I_i \,| = 1 \neq 0 = |\, M \cap J_k^1 \,|,$$

i.e. $M \hookrightarrow_= K$ cannot hold both in $I_i$ and in $J_k^1$, i.e. $\mathcal{C} \not\models M \hookrightarrow_= K$. Thus, it holds $\lambda_{-,-} \neq n$.

If $\lambda_{a,-} = n$, then $\{a_1, \ldots, a_n\} \subseteq M$. Then $M$ is not minimal in $M \hookrightarrow_= K$, since $M \hookrightarrow_= K$ can be simplified to $M' \hookrightarrow_= K - n + 1$, where $M' = M \setminus \{a_1, \ldots, a_n\} \subsetneq M$. Thus, it holds $\lambda_{a,-} \neq n$.

For the interpretations $I_k$ and $J_k^2$ it holds

$$\begin{aligned} |\, M \cap I_k \,| &= \lambda_{a,b} + \lambda_{a,-} - x + y, \\ |\, M \cap J_k^2 \,| &= 2 \cdot \lambda_{a,b} + \lambda_{a,-} - x + \lambda_{-,b}, \end{aligned}$$

where $x = |M \cap \{a_k\}|$ and $y = |M \cap \{c\}|$. Since $|\, M \cap I_k \,| = |\, M \cap J_k^2 \,|$, we get

$$\lambda_{a,b} + \lambda_{-,b} = y \in \{0, 1\},$$

i.e. there exists at most one $b_i \in M$.

If $c \notin M$, i.e. $y = 0$, then $\lambda_{a,b} = \lambda_{-,b} = 0$, i.e. $\Lambda_{a,-} \cup \Lambda_{-,-} = \langle\, 1, n \,\rangle$. Since $\Lambda_{a,-} \neq \langle\, 1, n \,\rangle \neq \Lambda_{-,-}$, there exist indices $i_1 \in \Lambda_{-,-}$ and $i_2 \in \Lambda_{a,-}$. Then

$$|\, M \cap I_{i_1} \,| = |\, M \cap I_{i_2} \,| + 1,$$

i.e. $M \hookrightarrow_= K$ cannot hold both in $I_{i_1}$ and in $I_{i_2}$, i.e. $\mathcal{C} \not\models M \hookrightarrow_= K$.

Thus, we get $c \in M$, i.e. $y = 1$, i.e. there exists exactly one $b_i \in M$. Now we can distinguish between two cases:

- $\lambda_{a,b} = 1$ and $\lambda_{-,b} = 0$,
- $\lambda_{a,b} = 0$ and $\lambda_{-,b} = 1$.

Firstly, assume $\lambda_{a,b} = 1$ and $\lambda_{-,b} = 0$, and let $\Lambda_{a,b} = \{i_1\}$. If $\lambda_{a,-} \neq 0$, then consider $i_2 \in \Lambda_{a,-}$. Then

$$|\, M \cap I_{i_1} \,| = |\, M \cap I_{i_2} \,| + 1,$$

i.e. $M \hookrightarrow_= K$ cannot hold both in $I_{i_1}$ and in $I_{i_2}$, i.e. $\mathcal{C} \not\models M \hookrightarrow_= K$. Thus, $\lambda_{a,-} = 0$, i.e. $M = \{a_{i_1}, b_{i_1}, c\}$. Then, it holds $i_1 = k$, since otherwise $M \hookrightarrow_= K \in \mathcal{C}'$. But then

$$|\, M \cap J_k^2 \,| = 0 \neq 2 = |\, M \cap I_k \,|.$$

i.e. $M \hookrightarrow_= K$ cannot hold both in $I_{J_k^2}$ and in $I_k$, i.e. $\mathcal{C} \not\models M \hookrightarrow_= K$.

Secondly, assume $\lambda_{a,b} = 0$ and $\lambda_{-,b} = 1$, and let $\Lambda_{-,b} = \{i_1\}$. If $\lambda_{a,-} \neq 0$, then consider $i_2 \in \Lambda_{a,-}$. Then

$$|\, M \cap I_{i_1} \,| = |\, M \cap I_{i_2} \,| + 2,$$

i.e. $M \hookrightarrow_= K$ cannot hold both in $I_{i_1}$ and in $I_{i_2}$, i.e. $\mathcal{C} \not\models M \hookrightarrow_= K$. Thus, $\lambda_{a,-} = 0$, i.e. $M = \{b_{i_1}, c\}$. Then $\lambda_{-,-} \neq 0$, and we can consider $i_2 \in \Lambda_{-,-}$. Then

$$|\, M \cap I_{i_1} \,| = 2 \neq 1 = |\, M \cap I_{i_2} \,|.$$

i.e. $M \hookrightarrow_= K$ cannot hold both in $I_{i_1}$ and in $I_{i_2}$, i.e. $\mathcal{C} \not\models M \hookrightarrow_= K$.

2. If $C = \{a_1, \ldots, a_n\} \hookrightarrow_= n - 1$, then $\mathcal{C}'$ has got the following $2^n + 1$ additional models, which are not models of $\mathcal{C}$:

   - $J_{\langle\, x_1, \ldots, x_n \,\rangle} = \{\, x_1, \ldots, x_n \,\} \cup \{c\}$, for all vectors $\langle\, x_1, \ldots, x_n \,\rangle \in \{a_1, b_1\} \times \ldots \times \{a_n, b_n\}$,
   - $J = \{\, a_1, b_1, \ldots, a_n, b_n \,\}$.

   Firstly, we want to show that the $a_i$ and $b_i$ occur in pairs in $M$, i.e.

   $$\Lambda_{a,b} \cup \Lambda_{-,-} = \langle\, 1, n \,\rangle.$$

   Assume, that there exists $i' \in \langle\, 1, n \,\rangle \setminus (\Lambda_{a,b} \cup \Lambda_{-,-})$, i.e. $i' \in \Lambda_{a,-} \cup \Lambda_{-,b}$. Then we consider the two tuples $t_{a_{i'}}$ and $t_{b_{i'}}$, where

   $$t_{x_{i'}} = \langle\, x_1, \ldots, x_{i'-1}, x_{i'}, x_{i'+1}, \ldots, x_n \,\rangle,$$

   such that $t_{a_{i'}}$ and $t_{b_{i'}}$ coincide on the elements $x_i \in \{a_i, b_i\}$, $i \in \langle\, 1, n \,\rangle \setminus \{i'\}$, and differ on $x_{i'}$. Obviously, it holds

   $$|\, M \cap J_{t_{a_{i'}}} \,| = |\, M \cap J_{t_{b_{i'}}} \,| + 1,$$

   if $i \in \Lambda_{a,-}$, and

   $$|\, M \cap J_{t_{a_{i'}}} \,| = |\, M \cap J_{t_{b_{i'}}} \,| - 1,$$

   if $i' \in \Lambda_{-,b}$. Thus, $M \hookrightarrow_= K$ cannot hold both in $J_{t_{a_{i'}}}$ and in $J_{t_{b_{i'}}}$, i.e. $\mathcal{C} \not\models M \hookrightarrow_= K$.

In the following we will consider the models $J$ and $I_i$ for an arbitrary $i \in \langle 1, n \rangle$, and we will make use of the fact that $|\, M \cap I_i \,| = |\, M \cap J \,|$ must hold.

If $c \in M$, then it holds $|\, M \cap I_i \,| = \lambda_{a,b} + 1$ and $|\, M \cap J \,| = 2 \cdot \lambda_{a,b}$. We get $\lambda_{a,b} = 1$. Thus, $M = \{a_{i'}, b_{i'}, c\}$. $\mathcal{C}' \models M \hookrightarrow_{=} K$ implies $K = 2$. Thus, $M \hookrightarrow_{=} K \in \mathcal{C}'$.

If $c \notin M$, then it holds $|\, M \cap I_i \,| = \lambda_{a,b}$ and $|\, M \cap J \,| = 2 \cdot \lambda_{a,b}$. We get $\lambda_{a,b} = 0$. Thus, $M = \emptyset$, which implies $K = 0$. Thus, $M \hookrightarrow_{=} K$ is trivial. □

Moreover it can be seen that $\mathcal{C}' \rhd M \hookrightarrow_{=} K$ implies that $M \hookrightarrow_{=} K \in \mathcal{C}'$ for non–trivial cardinality constraints. As a consequence of Theorem 3.2, if $n > n_{max}$, then no sound calculus for reasoning with exactly–cardinality constraints that is given by a definite logic program $\mathcal{P}'_{cc_=}$ is able to derive the constraints $cc_i$, $i \in \langle 1, n+2 \rangle$, from the set $\mathcal{C}'$.

These calculi can only derive exactly–cardinality constraints as intermediate results. Note, however, that other calculi which are deriving more general intermediate results could be able to derive these conclusions. E.g., from the positive–disjunctive deductive database $\mathcal{P}_{\mathcal{C}_n}$ the positive disjunctions and the denial rules in the sets $\mathcal{C}(cc_i)$ can be derived due to Lemma 2.1.

## 4 Experimental Results

We have investigated a collection of test sets $\mathcal{C}$ of exactly–cardinality constraints of various sizes. In the following we report on sets that were derived from intermediate constellations of the game Mine Sweeper: each cardinality constraint $M \hookrightarrow_{=} K$ expresses how many of the non–investigated neighbours of a field on an $N \times N$ chess board contains a mine. Thus, the cardinalities of the sets $M$ can range from 0 to 8, since each field has 8 neighbours. Larger sets of cardinality constraints can be obtained by enlarging the size $N$ of the board. The goal of the player is to investigate the fields of the board, such that he never investigates a field with a mine. Each time a field is investigated, the system informs the player about the number of mines on the neighbour fields. Of course, the intial investigations cannot rely on any knowledge, but after a few investigations in many cases it becomes possible to infer that some neighbour fields do not contain a mine, or that other fields must contain a mine.

**Mixing $\mathcal{P}_{\mathcal{C}}$ and $\mathcal{P}_{cc_=}$**

We generated 350 sets $\mathcal{C}$ of cardinality constraints (for boards of the size $N = 8$), and we inferred the set $T_{\mathcal{C}}$ of all elements in the universe that definitely have to be chosen as a consequence of $\mathcal{C}$, and also the set $F_{\mathcal{C}}$ of all elements that definitely cannot be chosen:

$$\begin{aligned} T_{\mathcal{C}} &= \{\, A \in H_B \,|\, \mathcal{C} \models \{A\} \hookrightarrow_{=} 1 \,\}, \\ F_{\mathcal{C}} &= \{\, A \in H_B \,|\, \mathcal{C} \models \{A\} \hookrightarrow_{=} 0 \,\}. \end{aligned}$$

For about 99.33% of the generated sets $\mathcal{C}$ the sets $T_{\mathcal{C}}$ and $F_{\mathcal{C}}$ could be derived extremely efficient using the calculus given by the definite logic program $\mathcal{P}_{cc_=}$.

The computation time needed when working with $\mathcal{P}_{cc_=}$ was orders of magnitude smaller than the time needed for computing with the disjunctive consequence operator $\mathcal{T}^s_{\mathcal{P}}$ of the positive–disjunctive deductive database $\mathcal{P}_{\mathcal{C}}$ according to Lemma 2.2, and yet the same result was derived. This is due to the fact that if $K$ gets close to $|M|/2$, then the size of $\mathcal{C}(M \hookrightarrow_{=} K)$ becomes very large, and the computation time grows drastically with the size of $\mathcal{P}_{\mathcal{C}}$. Only for 0.66% of the test cases, $\mathcal{T}^s_{\mathcal{P}}$ was able to derive knowledge, when $\mathcal{P}_{cc_=}$ did not return any conclusions.

Thus, it is a good strategy first to apply $\mathcal{P}_{cc_=}$ and see if it returns any results, and then to apply disjunctive reasoning only in case that no results have been obtained.

**Comparison of Different Methods**

For all of our generated test sets $\mathcal{C}$ of cardinality constraints, we have applied two semantically equivalent types of methods of disjunctive reasoning to obtain conclusions (cf. [7]):

1. *state generation*, i.e. positive hyperresolution with the disjunctive consequence operator $\mathcal{T}^s_{\mathcal{P}}$, and
2. *model generation* based on an operator $\mathcal{T}^{INT}_{\mathcal{P}}$, which computes the set $lfp(\mathcal{T}^{INT}_{\mathcal{P}}) = \mathcal{MM}(\mathcal{P})$ of minimal models of $\mathcal{P}$ as its least fixpoint first and then derives the conclusions from $\mathcal{MM}(\mathcal{P})$.

The methods are implemented in the system DisLog (cf. [8, 9]) for efficient reasoning in disjunctive deductive databases. In a variant of Case 1, the disjunctive consequence operator $\mathcal{T}_{\mathcal{P}}^{s}$ was speeded up by using the compact *tree data structures* for disjunctive and negative Herbrand states of DisLog (cf. [6]).

It is well–known that in general neither $\mathcal{T}_{\mathcal{P}}^{s}$ nor $\mathcal{T}_{\mathcal{P}}^{INT}$ is consistently superior to the other, which is due to the fact that for some boolean formulas the conjunctive normal form is smaller than the disjunctive normal form, and sometimes it is vice versa. But, interestingly it turned out that for the databases $\mathcal{P}_{\mathcal{C}}$ obtained from cardinality constraints, $\mathcal{T}_{\mathcal{P}}^{s}$ was always much faster than $\mathcal{T}_{\mathcal{P}}^{INT}$. Especially for some dual databases $\mathcal{P}_{\mathcal{C}}^{d}$ the running time of $\mathcal{T}_{\mathcal{P}}^{INT}$ was extremely large.

Based on the minimal models in $\mathcal{MM}(\mathcal{P}_{\mathcal{C}})$ it is possible to estimate the *probability* that an atom $A$ is selected by a model $I$ of $\mathcal{C}$ as $n_1/n_2$, where $n_1 = |\{ I \in \mathcal{MM}(\mathcal{P}_{\mathcal{C}}) \mid A \in I \}|$ and $n_2 = |\mathcal{MM}(\mathcal{P}_{\mathcal{C}})|$.

**Weight Functions**

For obtaining an intuition about the complexity of reasoning with the positive–disjunctive deductive databases $\mathcal{P}_{\mathcal{C}}$ obtained from cardinality constraints, we have related the computation times to the weight of $\mathcal{P}_{\mathcal{C}}$. For this pupose we have considered the *following weight functions*:

$$w_n(\mathcal{C}(M \hookrightarrow_{=} K)) = \binom{|M|}{K+1} \cdot (K+1)^n + \binom{|M|}{K-1} \cdot (|M| - K + 1)^n,$$

and $w_n(\mathcal{C}) = \sum_{c \in \mathcal{C}} w_n(\mathcal{C}(c))$ for a set $\mathcal{C}$ of cardinality constraints. These functions compute the sum of the powers $l^n$, where $l$ ranges over the lengths of the disjunctive rules in the sets $\mathcal{C}(M \hookrightarrow_{=} K)$. The motivation for using the powers $l^n$ is the observation that on average reasoning with one long formula $\phi$ of length $l$ in $\mathcal{C}(M \hookrightarrow_{=} K)$ is much more complex that reasoning with two shorter formulas $\phi_1$ and $\phi_2$ of the length $l_1$ and $l_2$, respectively, where $l_1 + l_2 = l$. Thus, the weight of $\phi$ should be larger than the sum of the weights of $\phi_1$ and $\phi_2$. Indeed this holds for $n \geq 2$, since $(l_1 + l_2)^n > l_1^n + l_2^n$.

**Results**

For the weight function $w_2$ ($n = 2$) it turned out that the average computation time $t(w_2(\mathcal{C}))$ in seconds depending on the weight of $\mathcal{C}$ was given by a polynomial of small degree, which was approximately

$$t(w) \approx 3 \cdot 10^{-5} \cdot w^{3/2},$$

cf. Figure 1, which has got a doubly logarithmic scale. Four curves are shown for state generation, namely two for $\mathcal{T}_{\mathcal{P}}^{s}$ ($t_1^T$ for $T_{\mathcal{C}}$, $t_1^F$ for $F_{\mathcal{C}}$) and two for the variant of $\mathcal{T}_{\mathcal{P}}^{s}$ based on the tree data structure ($t_2^T$ for $T_{\mathcal{C}}$, $t_2^F$ for $F_{\mathcal{C}}$). For smaller values of $w = w_2(\mathcal{C}) < 2 \cdot 10^2$ we get $t_2^T(w) < t_2^F(w) < t_1^T(w) < t_1^F(w)$, whereas for larger values $w = w_2(\mathcal{C}) > 2 \cdot 10^2$ we get $t_1^T(w) < t_2^T(w) < t_2^F(w) < t_1^F(w)$. For extremal values of $w_2(\mathcal{C})$, the deviation from the straight line in Figure 1 can be explained by the *overlap* in the databases $\mathcal{P}_{\mathcal{C}}$, which varied correspondingly and which we have not considered in our formula for $w_2(\mathcal{C})$.

## 5 Conclusions

For reasoning with cardinality constraints the mixing of methods from disjunctive logic programming (DLP), which encode the constraints into disjunctive logic rules, with a calculus for directly dealing with the constraints proved itself to be very useful.

Interestingly, cardinality constraints turned out to be a domain where standard methods of reasoning in disjunctive deductive databases performed totally differently. It is well–known that in general neither state generation nor model generation is superior to the other. But on the restricted domain of disjunctive deductive databases derived from cardinality constraints it seems that state generation dominates model generation in most of the cases.

A comparison with methods from constraint logic programming (CLP) showed that CLP is much faster than DLP. But when incomplete, disjunctive knowledge is envolved next to the cardinality constraints, then CLP is not applicable. In our experiments CLP was comparable in performance to reasoning with our specially designed calculus for directly dealing with the constraints.

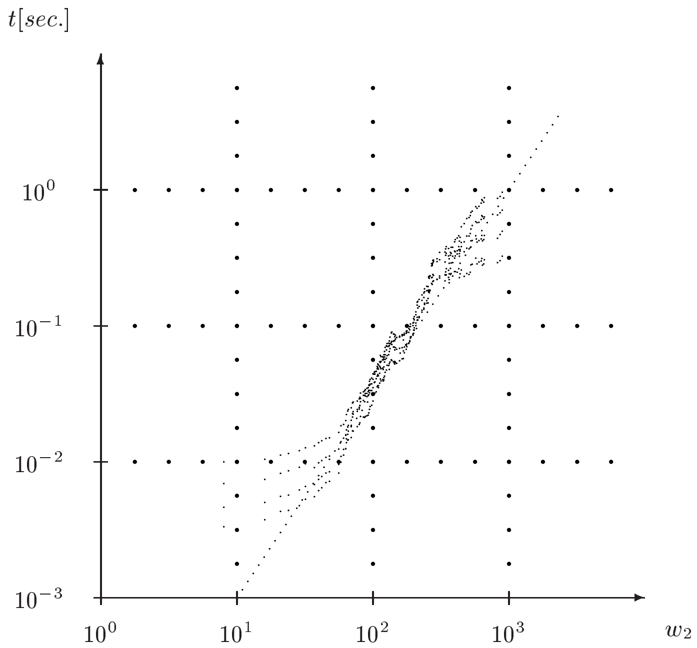


Figure 1: Average Running Times for Disjunctive Reasoning


## Acknowledgements

The authors would like to thank Joachim Baumeister for his useful comments on the paper and Johannes Lönne for giving an incentive to study the concept of cardinality constraints in the context of disjunctive deductive databases.


## References


[1] *S. Ceri, G. Gottlob, L. Tanca:* Logic Programming and Databases, Springer, 1990.

[2] *J.W. Lloyd:* Foundations of Logic Programming, Second Edition, Springer, 1987.

[3] *J. Lobo, J. Minker, A. Rajasekar:* Foundations of Disjunctive Logic Programming, MIT Press, 1992.

[4] *K. Marriott, P. Stuckey:* Programming with Constraints – An Introduction, MIT Press, 1998.

[5] *J. Minker, A. Rajasekar:* A Fixpoint Semantics for Disjunctive Logic Programs. *Journal of Logic Programming*, 9(1), 1990, pages 45–74.

[6] *D. Seipel:* An Efficient Computation of the Extended Generalized Closed World Assumption by Support–for–Negation Sets. Proc. Intl. Conference on Logic Programming and Automated Reasoning (LPAR'94), Springer, LNAI 822, 1994, pages 245–259.

[7] *D. Seipel, J. Minker, C. Ruiz*: Model Generation and State Generation for Disjunctive Logic Programs, *Journal of Logic Programming*, 32(1), 1997, pages 48–69.

[8] *D. Seipel, H. Thöne*: DisLog – A System for Reasoning in Disjunctive Deductive Databases, Proc. Intl. Workshop on the Deductive Approach to Information Systems and Databases 1994 (DAISD'94), pages 325–343.

[9] *D. Seipel:* DisLog – A Disjunctive Deductive Database Prototype, Proc. 12th Workshop on Logic Programming (WLP'97), 1997, pages 136–143. DisLog is available at "http://www-info1.informatik.uni-wuerzburg.de/databases/DisLog".

[10] *A.H. Yahya:* Minimal Model Generation for Refined Answering of Generalized Queries in Disjunctive Deductive Databases. *Journal of Data and Knowledge Engineering*, 34(3), 2000, pages 219–249.