Logic programs with monotone cardinality atoms

Victor W. Marek,
Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA
(*e-mail:* marek@cs.uky.edu)

Ilkka Niemelä,
Department of Computer Science and Engineering
Helsinki University of Technology,
P.O.Box 5400, FI-02015 HUT, Finland
(*e-mail:* Ilkka.Niemela@hut.fi)

Mirosław Truszczyński,
Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA
(*e-mail:* mirek@cs.uky.edu)

Abstract

We investigate *mca-programs*, that is, logic programs with clauses built of monotone cardinality atoms of the form kX, where k is a non-negative integer and X is a finite set of propositional atoms. We develop a theory of mca-programs. We demonstrate that the operational concept of the one-step provability operator generalizes to mca-programs, but the generalization involves nondeterminism. Our main results show that the formalism of mca-programs is a common generalization of (1) normal logic programming with its semantics of models, supported models and stable models, (2) logic programming with cardinality atoms and with the semantics of stable models, as defined by Niemelä, Simons and Soininen, and (3) of disjunctive logic programming with the *possible-model* semantics of Sakama and Inoue.

1 Introduction

We introduce and study logic programs whose clauses are built of *monotone cardinality atoms* (*mc-atoms*), that is, expressions of the form kX, where k is a non-negative integer and X is a finite set of propositional atoms. Intuitively, kX is true in an interpretation M if at least k atoms in X are true in M. Thus, the intended role for mc-atoms is to represent constraints on lower bounds of cardinalities of sets. We refer to programs with mc-atoms as *mca-programs*. We are motivated in this work by the recent emergence and demonstrated effectiveness of logic programming extended with means to model cardinality constraints (Niemelä et al. 1999; Niemelä and Simons 2000; Simons et al. 2002), and by the need to establish sound theoretical basis for such formalisms.

In the paper, we develop a theory of mca-programs. In that we closely follow the development of normal logic programming and lift all its major concepts, techniques and results to the setting of mca-programs. There is, however, a basic difference. Mc-atoms have, by their very nature, a built-in nondeterminism. They can be viewed

as shorthands for certain disjunctions and, in general, there are many ways to make an mc-atom kX true. This nondeterminism has a key consequence. The one-step provability operator is no longer deterministic, as in normal logic programming, where it maps interpretations to interpretations. In the case of mca-programs, the one-step provability operator is nondeterministic. It assigns to an interpretation M a *set* of interpretations, each regarded as possible and equally likely outcome of applying the operator to M.

Modulo this difference, our theory of mca-programs parallels that of normal logic programs. First, we introduce *models* and *supported* models of an mca-program, and describe them in terms of the one-step provability operator in much the same way it is done in normal logic programming. To define *stable* models we first define the class of *Horn* mca-programs by disallowing the negation operator in the bodies of clauses. We show that the nondeterministic one-step provability operator associates with Horn mca-programs a notion of a (nondeterministic) derivation (the counterpart to the bottom-up derivation with normal Horn programs) and a class of *derivable models* (counterparts to the least model of a normal Horn program). We then lift the notion of the Gelfond-Lifschitz reduct (Gelfond and Lifschitz 1988) to the case of mca-programs and use the notions of a derivation and the reduct to define a stable model of an mca-program. A striking aspect of our construction is that all its steps are *literal* extensions of the corresponding steps in the original approach. We show that stable models behave as expected. They are supported and, in case of Horn mca-programs, derivable.

An intended meaning of an mc-atom $1\{a\}$ is that a be true. More formally, $1\{a\}$ is true in an interpretation if and only if a is true in that interpretation. That connection implies a natural representation of normal logic programs as mca-programs. We show that this representation preserves all semantics we discuss in the paper. It follows that the formalism of mca-programs can be viewed as a direct generalization of normal logic programming.

As we noted, an extension of logic programming with direct ways to model cardinality constraints was first proposed in (Niemelä et al. 1999). That work defined a syntax of logic programs with cardinality constraints (in fact, with more general *weight constraints*) and introduced the notion of a *stable model*. We will refer to programs in that formalism as *ca-programs*. One of the results in (Niemelä et al. 1999) showed that ca-programs generalized normal logic programming with the stable-model semantics of Gelfond and Lifschitz (Gelfond and Lifschitz 1988). However, the notion of the reduct underlying the definition of a stable model given in (Niemelä et al. 1999) is different from that proposed by Gelfond and Lifschitz (Gelfond and Lifschitz 1988) and the precise nature of the relationship between normal logic programs and ca-programs was not clear.

Mca-programs explicate this relationship. The formalism of mca-programs parallels normal logic programming. In particular, major concepts, results and techniques in normal logic programming have counterparts in the setting of mca-programs. On the other hand, under some simple transformations, ca-programs are equivalent to mca-programs. Thus, through this connection of ca-programs to mca-programs,

the theory of normal logic programming can be lifted to the setting of ca-programs leading to new characterizations of stable models of ca-programs.

Finally, we show that mca-programs not only provide an overarching framework for both normal logic programs and ca-programs. They are also useful in investigating disjunctive logic programs. In the paper, we show that logic programming with mc-atoms generalize disjunctive logic programming with the possible-model semantics introduced in (Sakama and Inoue 1994).

2 Logic programs with monotone cardinality atoms

Let At be a set of (propositional) *atoms*. An *mc-atom over* At (short for a *monotone cardinality atom over* At) is any expression of the form kX, where k is a non-negative integer and $X \subseteq At$ is a *finite* set such that $k \leq |X|$. We call X the *atom set* of an mc-atom $A = kX$ and denote it by $aset(A)$. An intuitive reading of an mc-atom kX is: *at least k atoms in X are true*. In particular, $1\{a\}$ states that a is true. In other words, intuitively, an mc-atom $1\{a\}$ is equivalent to a. The intended meaning of kX explains the requirement that $k \leq |X|$. Clearly, if $k > |X|$, it is impossible to have in X at least k true atoms and the expression kX is equivalent to a contradiction.

An *mc-literal* is an expression of the form A or $\mathbf{not}(A)$, where A is an mc-atom. An *mca-clause* (short for a *monotone-cardinality-atom clause*) is an expression r of the form

$$H \leftarrow L_1, \ldots, L_m, \tag{1}$$

where H is an mc-atom and L_i, $1 \leq i \leq m$, are mc-literals. We call the mc-atom H the *head* of r and denote it by $hd(r)$. We call the set $\{L_1, \ldots, L_m\}$ the *body* of r and denote it by $bd(r)$. An mca-clause is *Horn* if its body does not contain literals of the form $\mathbf{not}(A)$. Finally, for an mca-clause r, we define the *head set* of r, $hset(r)$, by setting $hset(r) = aset(hd(r))$.

Mca-clauses form *mca-programs*. We define the *head set* of an mca-program P, $hset(P)$, by $hset(P) = \bigcup\{hset(r) \colon r \in P\}$ (if $P = \emptyset$, $hset(P) = \emptyset$, as well). If all clauses in an mca-program P are Horn, P is a *Horn* mca-program.

One can give a declarative interpretation to mca-programs in terms of a natural extension of the semantics of propositional logic. We say that a set M of atoms *satisfies* an mc-atom kX if $|M \cap X| \geq k$, and M *satisfies* an mc-literal $\mathbf{not}(kX)$ if it does not satisfy kX (that is, if $|M \cap X| < k$). A set of atoms M satisfies an mca-clause (1) if M satisfies H whenever M satisfies all literals L_i, $1 \leq i \leq m$. Finally, a set of atoms M satisfies an mca-program P if it satisfies all clauses in P. We often say "is a model of" instead of "satisfies". We use the symbol $\models$ to denote the satisfaction relation.

The following straightforward property of mc-atoms explains the use of the term "monotone" in their name.

Proposition 1
Let A be an mc-atom over a set of atoms At. For every sets $M, M' \subseteq At$, if $M \subseteq M'$ and $M \models A$ then $M' \models A$.

Mca-clauses also have a procedural interpretation in which they are viewed as derivation rules. Intuitively, if an mca-clause r has its body satisfied by some set of atoms M, then r provides *support* for deriving from M any set of atoms M' such that

1. M' consists of atoms mentioned in the head of r (r provides no grounds for deriving atoms that do not appear in its head)
2. M' satisfies the head of r (since r "fires", the constraint imposed by its head must hold).

Clearly, the process of deriving M' from M by means of r is *nondeterministic* in the sense that, in general, there are several sets that are supported by r and M.

This notion of nondeterministic derivability extends to programs and leads to the concept of the nondeterministic one-step provability operator. Let P be an mca-program and let $M \subseteq At$ be a set of atoms. We set $P(M) = \{r \in P : M \models bd(r)\}$. We call mca-clauses in $P(M)$, *M-applicable*.

Definition 1
Let P be an mca-program and let $M \subseteq At$. A set M' is *nondeterministically one-step provable* from M by means of P, if $M' \subseteq hset(P(M))$ and $M' \models hd(r)$, for every mca-clause r in $P(M)$.

The *nondeterministic one-step provability operator* $\mathrm{T}_P^{\mathrm{nd}}$, is a function from $\mathcal{P}(At)$ to $\mathcal{P}(\mathcal{P}(At))$ such that for every $M \subseteq At$, $\mathrm{T}_P^{\mathrm{nd}}(M)$ consists all sets M' that are nondeterministically one-step provable from M by means of P.

We first observe that for every $M \subseteq At$, $\mathrm{T}_P^{\mathrm{nd}}(M)$ is nonempty.

Proposition 2
Let P be an mca-program and let $M \subseteq At$. Then, $hset(P(M)) \in \mathrm{T}_P^{\mathrm{nd}}(M)$. In particular, $\mathrm{T}_P^{\mathrm{nd}}(M) \neq \emptyset$.

Proof
Let $r \in P(M)$. Then, $hset(P(M)) \cap hset(r) = hset(r)$. By the definition of mc-atoms, $hset(r) \models hd(r)$. Thus, $hset(P(M)) \models hd(r)$ and, consequently, $hset(P(M)) \in \mathrm{T}_P^{\mathrm{nd}}(M)$. □

It follows that $\mathrm{T}_P^{\mathrm{nd}}$ can be viewed as a formal representation of a *nondeterministic* operator on $\mathcal{P}(At)$, which assigns to every subset M of At a subset of At arbitrarily selected from the collection $\mathrm{T}_P^{\mathrm{nd}}(M)$ of possible outcomes. Since $\mathrm{T}_P^{\mathrm{nd}}(M)$ is nonempty, this nondeterministic operator is well defined.

The operator $\mathrm{T}_P^{\mathrm{nd}}$ plays a fundamental role in our research. It allows us to formalize procedural interpretations of mca-clauses and identify for them matching classes of models that provide the corresponding declarative account.

Our first result characterizes models of mca-programs. This characterization is a generalization of the familiar description of models of normal logic programs as prefixpoints of T_P in terms of the operator $\mathrm{T}_P^{\mathrm{nd}}$.

Theorem 2
Let P be an mca-program and let $M \subseteq At$. The set M is a model of P if and only if there is $M' \in \mathrm{T}_P^{\mathrm{nd}}(M)$ such that $M' \subseteq M$.

Proof
Let M be a model of P and $M' = M \cap hset(P(M))$. Let $r \in P(M)$. Since M is a model of r, $M \models hd(r)$. Clearly, $hset(r) \subseteq hset(P(M))$. Thus, $M \cap hset(r) = M' \cap hset(r)$ and, consequently, $M' \models hd(r)$. It follows that $M' \in \mathrm{T}_P^{\mathrm{nd}}(M)$. Since $M' \subseteq M$, the assertion follows.

Conversely, let us assume that there is $M' \in \mathrm{T}_P^{\mathrm{nd}}(M)$ such that $M' \subseteq M$. Let $r \in P$ be a clause such that $M \models bd(r)$. Since $M' \in \mathrm{T}_P^{\mathrm{nd}}(M)$, $M' \models hd(r)$. We recall that $hd(r)$ is an mc-atom. Thus, by Proposition 1, $M \models hd(r)$, as well. It follows that M is a model of every clause in P and, consequently, of P. □

A straightforward corollary states that every mca-program has a model.

Corollary 1
Let P be an mca-program. Then, $hset(P)$ is a model of P.

Proof
Let us denote $M = hset(P)$. By Proposition 2, $hset(P(M)) \in \mathrm{T}_P^{\mathrm{nd}}(M)$. Since we have $hset(P(M)) \subseteq hset(P) = M$, the assertion follows by Theorem 2. □

In fact, a stronger property holds. We can show, that like normal logic programs, mca-programs have *minimal* models.

Proposition 3
Every model of an mca-program P contains a minimal model. In particular, every mca-program P has a minimal model.

Proof
Let $\langle M_\xi \rangle_{\xi<\beta}$ be a descending family of models of P (that is, $M_\xi \supseteq M_{\xi'}$, for every $\xi < \xi' < \beta$). We will show that $M^\cap = \bigcap_{\xi<\beta} M_\xi$ is a model of P. To this end, let us consider an arbitrary rule $r \in P$, say

$$r = kX \leftarrow k_1 X_1, \ldots, k_m X_m, \mathbf{not}(l_1\, Y_1), \ldots, \mathbf{not}(l_n\, Y_n)$$

such that $M^\cap \models bd(r)$. Since all sets X, $X_1, \ldots, X_m$ and $Y_1, \ldots, Y_n$ are finite, there is $\eta < \beta$ such that $X \cap M^\cap = X \cap M_\eta$, $X_i \cap M^\cap = X_i \cap M_\eta$, $1 \leq i \leq m$, and $Y_i \cap M^\cap = Y_i \cap M_\eta$, $1 \leq i \leq n$. Thus, $M_\eta \models bd(r)$. Since M_η is a model of P, $M_\eta \models hd(r)$. Since $X \cap M_\eta = X \cap M^\cap$, $M^\cap \models hd(r)$. Thus, $M^\cap$ is a model of P.

Let us now consider a model M of P. From our argument, it follows that the poset of all models of P that are contained in M (ordered by the relation $\supseteq$) satisfies the assumptions of the Zorn Lemma. Consequently, it contains minimal elements. Thus, the assertion follows. □

Models of mca-programs may contain elements that have no support in a program and the model itself, that is, cannot be derived from atoms in the model by means of rules in the program. For instance, let us consider an mca-program P consisting of the clause: $1\{p, q\} \leftarrow \mathbf{not}(1\{q\})$, where p and q are two different atoms. Let $M_1 = \{q\}$. Clearly, M_1 is a model of P. However, M_1 has no support in P and itself. Indeed, $\mathrm{T}^{\mathrm{nd}}_P(M_1) = \{\emptyset\}$ and so, P and M_1 do not provide support for any atom. Similarly, another model of P, the set $M_2 = \{p, r\}$, where $r \in At$ is an atom different from p and q, has no support in P and itself. We have $\mathrm{T}^{\mathrm{nd}}_P(M_2) = \{\{p\}, \{q\}, \{p, q\}\}$ and so, p has support in P and M_2, but r does not. Finally, the set $M_3 = \{p\}$, which is also a model of P, *has support* in P and itself. Indeed, $\mathrm{T}^{\mathrm{nd}}_P(M_3) = \{\{p\}, \{q\}, \{p, q\}\}$ and there is a way to derive M_3 from P and M_3. We formalize now this discussion in the following definition.

Definition 3
Let P be an mca-program. A set of atoms M is a *supported model* of P if $M \in \mathrm{T}^{\mathrm{nd}}_P(M)$.

The use of the term "model" is justified. By Theorem 2, supported models of P are indeed models of P, as stated in the following result.

Corollary 2
Every supported model of an mca-program P is a model of P.

Finally, we have the following characterization of supported models.

Proposition 4
Let P be an mca-program. A set $M \subseteq At$ is a supported model of P if and only if M is a model of P and $M \subseteq hset(P(M))$.

Proof
If M is a supported model of P then it is a model of P. Moreover, since $M \in \mathrm{T}^{\mathrm{nd}}_P(M)$, $M \subseteq hset(P(M))$. Conversely, if M is a model of P then $M \models P(M)$. Since $M \subseteq hset(P(M))$, $M \in \mathrm{T}^{\mathrm{nd}}_P(M)$, that is, M is a supported model of P. □

3 Horn mca-programs

To introduce *stable* models of mca-programs, we need first to study Horn mca-programs. With each Horn mca-program P one can associate the concept of a *P-derivation*. Namely, a *P-derivation* is a sequence $(X_n)_{n=0,1,\ldots}$ such that $X_0 = \emptyset$ and, for every non-negative integer n,

1. $X_n \subseteq X_{n+1}$, and
2. $X_{n+1} \in \mathrm{T}^{\mathrm{nd}}_P(X_n)$.

Given a derivation $t = (X_n)_{n=0,1,\ldots}$, we call $\bigcup_{n=0}^{\infty} X_n$ the *result* of the derivation t and denote it by R_t.

Proposition 5
Let P be a Horn mca-program and let t be a P-derivation. Then $R_t \subseteq hset(P(R_t))$.

Proof
Let $t = (X_n)_{n=0,1,\ldots}$. Clearly $X_0 \subseteq hset(P(R_t))$. Let n be a non-negative integer. Since $X_{n+1} \in \mathrm{T}_P^{\mathrm{nd}}(X_n)$, $X_{n+1} \subseteq hset(P(X_n)) \subseteq hset(P(R_t))$ (the last inclusion follows from the fact that $X_n \subseteq R_t$). Thus, $R_t = \bigcup_{n=0}^{\infty} X_n \subseteq hset(P(R_t))$. □

If P is a Horn mca-program then P-derivations exist. Let M be a model of P. We define the sequence $t^{P,M} = (X_n^{P,M})_{n=0,1,\ldots}$ as follows. We set $X_0^{P,M} = \emptyset$ and, for every $n \geq 0$, $X_{n+1}^{P,M} = hset(P(X_n^{P,M})) \cap M$.

Theorem 4
Let P be a Horn mca-program and let $M \subseteq At$ be its model. The sequence $t^{P,M}$ is a P-derivation.

Proof
We need to show that the conditions (1) and (2) from the definition of a P-derivation hold for the sequence $t^{P,M}$. To prove (1), we proceed by induction on n. For $n = 0$, the condition (1) is clearly satisfied. Let us assume that for some non-negative integer n, $X_n^{P,M} \subseteq X_{n+1}^{P,M}$ holds. Then

$$hset(P(X_n^{P,M})) \subseteq hset(P(X_{n+1}^{P,M})).$$

It follows that

$$X_{n+1}^{P,M} = hset(P(X_n^{P,M})) \cap M \subseteq hset(P(X_{n+1}^{P,M})) \cap M = X_{n+2}^{P,M}.$$

To prove (2), let us consider a non-negative integer n. By the definition, $X_{n+1}^{P,M} \subseteq hset(P(X_n^{P,M}))$. It remains to prove that $X_{n+1}^{P,M} \models P(X_n^{P,M})$. Let $r \in P(X_n^{P,M})$. Then $X_n^{P,M} \models bd(r)$ and, since $X_n^{P,M} \subseteq M$, $M \models bd(r)$. We recall that M is a model of P. Thus, $M \models hd(r)$. It follows that $M \cap hset(r) \models hd(r)$ and, consequently, $M \cap hset(P(X_n^{P,M})) \models hd(r)$. Since $X_{n+1}^{P,M} = M \cap hset(P(X_n^{P,M}))$, it follows that $X_{n+1}^{P,M} \models P(X_n^{P,M})$. □

We call the P-derivation $t^{P,M}$ the *canonical* P-derivation for M. Since every mca-program P has models, we obtain the following corollary.

Corollary 3
Every Horn mca-program has at least one derivation.

The results of derivations are supported models (and, thus, also models) of Horn mca-programs.

Proposition 6
Let P be a Horn mca-program and let t be a P-derivation. Then, the result of t, R_t, is a supported model of P.

Proof
Let $t = (X_n)_{n=0,1,\ldots}$ and let us consider a clause $r \in P(R_t)$. Since r has finitely many mc-atoms in the body, and since for each mc-atom A in $bd(r)$, $aset(A)$ is finite, there is i such that $X_i \models bd(r)$. By the definition of a P-derivation, $X_{i+1} \in \mathrm{T}_P^{\mathrm{nd}}(X_i)$. Thus, $X_{i+1} \models hd(r)$ and, since $X_{i+1} \subseteq R_t$, $R_t \models hd(r)$. It follows that R_t is a model of P. Moreover, by Proposition 5, $R_t \subseteq hset(P(R_t))$. Thus, R_t is a supported model of P (Proposition 4). □

We use the concept of a derivation to identify a certain class of models of Horn mca-programs.

Definition 5
Let P be a Horn mca-program. We say that a set of atoms M is a *derivable model* of P if there exists a P-derivation t such that $M = R_t$.

Derivable models can be obtained as results of their own canonical derivations.

Proposition 7
A model M of a Horn mca-program P is a derivable model of P if and only if $M = R_{t^{P,M}}$.

Proof
We assume first that M is a derivable model of P. Let $(X_n)_{n=0,1,\ldots}$ be a derivation with the result M (that is $M = \bigcup_{n=0}^{\infty} X_n$). Since for every non-negative integer n, $X_n^{P,M} \subseteq M$, to prove the assertion it is enough to show that for every non-negative integer n, $X_n \subseteq X_n^{P,M}$. We proceed by induction on n. The claim is evident for $n = 0$, as $X_0 = \emptyset$. Let us assume that the claim holds for some non-negative integer n. By the definition of P-derivation, $X_{n+1} \subseteq hset(P(X_n)) \subseteq hset(P(X_n^{P,M}))$. Since $X_{n+1} \subseteq M$, $X_{n+1} \subseteq M \cap hset(P(X_n^{P,M})) = X_{n+1}^{P,M}$. That completes the inductive step and the proof of the necessity of the condition. The converse implication (sufficiency) is straightforward. □

Proposition 6 and Theorem 4 entail several properties of Horn mca-programs, their derivations and models. We gather them in the following corollary.

Corollary 4
Let P be a Horn mca-program. Then:

1. P has at least one derivable model.
2. P has a largest derivable model.
3. Every derivable model of P is a supported model of P.
4. For every model M of P there is a derivable model M' of P such that $M' \subseteq M$.
5. Every minimal model of P is derivable.

Proof
(1) Since P has a model, it has a P-derivation (Theorem 4). The result of this derivation is a model of P (Proposition 6). By the definition, this model is derivable.
(2) The set $hset(P)$ is a model of P. Let R be the result of the canonical P-derivation for $hset(P)$. Clearly, R is a derivable model of P. We will show that every derivable model of P is a subset of R. Let M be a derivable model of P. Then M is the result of a canonical derivation for M. Since $M \subseteq hset(P)$, it follows by an easy induction that for every non-negative integer n, $X_n^{P,M} \subseteq X_n^{P,hset(P)}$. Consequently, $M \subseteq R$.
(3) This assertion follows directly from Proposition 6.
(4) Let M be a model of P and let t be a canonical P-derivation for M. Then, $R_t \subseteq M$ and, by Proposition 6, R_t is a model of P. Moreover, since R_t is the result of a derivation, it is a derivable model of P.
(5) This assertion follows directly from (4). □

Proposition 7 allows us to determine the complexity of the problem to verify whether a finite set of atoms M is a derivable model of a Horn mca-program P. First, we note that checking whether M is a model of P is a polynomial-time task. If M is not a model of P, it is not a derivable model of P. Thus, from now on, we can assume that M is a model of P and that the canonical P-derivation $t^{P,M}$ is well defined. Next, we observe that given the set $X_n^{P,M}$ of atoms (we recall that we write $X_n^{P,M}$ for terms of the canonical P-derivation for M), the set $X_{n+1}^{P,M}$ can be computed in polynomial time. Indeed, it takes polynomially many steps to verify whether an mca-clause $r \in P$ belongs to $P(M)$ and, if so, to compute $hd(r) \cap M$ — the contribution of the clause r to $X_{n+1}^{P,M}$. Since M is finite, the canonical P-derivation $t^{P,M}$ stabilizes (reaches its result) in at most $|M|$ steps. By Proposition 7, M is a derivable model of P if and only if it is a model of P and $M = R_{t^{P,M}}$. Our argument shows one can verify these two conditions in polynomial time. Thus, we get the following result.

Corollary 5
Given a finite Horn mca-program P and a finite set of atoms M, the problem to verify whether M is a derivable model of P is solvable in polynomial time.

We finish this section by proving some monotonicity properties concerning the non-deterministic one-step provability operators corresponding to Horn mca-programs.

Proposition 8
Let P be a Horn mca-program and let $M_1 \subseteq M_2$ be two sets of atoms. Then:

1. For every $Y \in \mathrm{T}_P^{\mathrm{nd}}(M_2)$ there exists $X \in \mathrm{T}_P^{\mathrm{nd}}(M_1)$ such that $X \subseteq Y$
2. For every $X \in \mathrm{T}_P^{\mathrm{nd}}(M_1)$ there exists $Y \in \mathrm{T}_P^{\mathrm{nd}}(M_2)$ such that $X \subseteq Y$

Proof
(1) By the definition, $Y \subseteq hset(P(M_2))$ and $Y \models hd(P(M_2))$. Let $X = Y \cap hset(P(M_1))$. Clearly, $X \subseteq hset(P(M_1))$. Next, we note that since P is a Horn mca-program, $P(M_1) \subseteq P(M_2)$ and, consequently, $Y \models hd(P(M_1))$. Thus, $X \models hd(P(M_1))$ and $X \in \mathrm{T}_P^{\mathrm{nd}}(M_1)$.
(2) Let $Y = hset(P(M_2))$. By Proposition 2, $Y \in \mathrm{T}_P^{\mathrm{nd}}(M_2)$. Moreover, we have $X \subseteq hset(P(M_1))$ and $P(M_1) \subseteq P(M_2)$. Thus, $X \subseteq Y$. □

Proposition 9
Let $P_1 \subseteq P_2$ be two Horn mca-programs and let M be a set of atoms. Then:

1. For every $Y \in \mathrm{T}^{\mathrm{nd}}_{P_2}(M)$ there exists $X \in \mathrm{T}^{\mathrm{nd}}_{P_1}(M)$ such that $X \subseteq Y$
2. For every $X \in \mathrm{T}^{\mathrm{nd}}_{P_1}(M)$ there exists $Y \in \mathrm{T}^{\mathrm{nd}}_{P_2}(M)$ such that $X \subseteq Y$

Proof
(1) Let us set $X = Y \cap hset(P_1(M))$. Clearly, $X \subseteq hset(P_1(M))$. Moreover, $Y \models hd(P_2(M))$ and, consequently, $Y \models hd(P_1(M))$. It follows that $X \models hd(P_1(M))$, too.
(2) It is easy to check that $Y = hset(P_2(M))$ has all the required properties. □

4 Stable models of mca-programs

We will now use the results of the two previous sections to introduce and study the class of *stable* models of mca-programs.

Definition 6
Let P be an mca-program and let $M \subseteq At$. The *reduct* of P with respect to M, P^M in symbols, is a Horn mca-program obtained from P by (1) removing from P every clause containing in the body a literal $\mathbf{not}(A)$ such that $M \models A$, and (2) removing all literals of the form $\mathbf{not}(A)$ from all remaining clauses in P. A set of atoms M is a *stable* model of P if M is a derivable model of the reduct P^M.

Stable models of an mca-program P are indeed models of P. Thus, the use of the term "model" in their name is justified. In fact, a stronger property holds: stable models of mca-programs are supported.

Proposition 10
Let P be an mca-program. If $M \subseteq At$ is a stable model of P then M is a supported model of P.

Proof
First, it follows directly from the corresponding definitions that $\mathrm{T}^{\mathrm{nd}}_P(M) = \mathrm{T}^{\mathrm{nd}}_{P^M}(M)$. Next, since M is a derivable model of P^M, M is a supported model of P^M (Corollary 4(3)). Thus, $M \in \mathrm{T}^{\mathrm{nd}}_{P^M}(M)$ and, consequently, $M \in \mathrm{T}^{\mathrm{nd}}_P(M)$. It follows that M is a supported model of P. □

With the notion of a stable model in hand, we can strengthen Proposition 6.

Proposition 11
Let P be a Horn mca-program. A set of atoms $M \subseteq At$ is a derivable model of P if and only if M is a stable model of P.

Proof
The assertion is a direct consequence of the fact that for every Horn mca-program P and for every set of atoms M, $P = P^M$. □

We will now prove yet another result that generalizes a well-known property of stable models of normal logic programs.

Proposition 12
Let Q and R be two mca-logic programs, and let M be a stable model of Q. If M is a model of R then M is a stable model of $Q \cup R$.

Proof
Since M is a stable model of Q, M is a derivable model of Q^M. By Proposition 7, M is the result of the canonical Q^M-derivation with respect to M. Since M is a model of $Q \cup R$, M is a model of $(Q \cup R)^M = Q^M \cup R^M$. Therefore, the canonical $(Q^M \cup R^M)$-derivation with respect to M is well defined. Its result is clearly contained in M. On the other hand, it contains the result of the canonical Q^M-derivation with respect to M, which is M. Therefore, the result of the canonical $(Q^M \cup R^M)$-derivation with respect to M is M. Thus, M is a derivable model of $(Q \cup R)^M$ and a stable model of $Q \cup R$. □

We will now describe a procedural characterization of stable models of mca-programs, relying on a notion of a derivation related to but different from the one we discussed in Section 3 in the context of Horn programs. A difference is that now at each stage in a derivation we must make sure that once a clause is applied, it remains applicable at *any* stage of the process. That property is not *a priori* guaranteed, due to the presence of negation in the bodies of general mca-clauses. We observe that, like before, this property of mca-programs generalizes a property of normal logic programs discussed in (Marek, Nerode and Remmel 1999).

Let P be an mca-program. A *cautious P-derivation* is a sequence $\langle X_n \rangle_{n=0,1,\ldots}$ of sets of atoms such that $X_0 = \emptyset$ and there exists a sequence of programs $\langle P_n \rangle_{n=0,1,\ldots}$ such that for all $n \in N$, $P_n \subseteq P$, $P_n \subseteq P_{n+1}$ and

(a) $X_{n+1} \in \mathrm{T}^{\mathrm{nd}}_{P_n}(X_n)$
(b) For every rule $r \in P_n$, if $\mathbf{not}(L) \in bd(r)$ then $X_{n+1} \models \mathbf{not}(L)$.

Thus, in a cautious P-derivation, the choice of the next set in the sequence is bound by two conditions. First, we choose a set according to the operation of the nondeterministic one-step-provability operator, but in each step the program used is only growing. Second, the choice made at step n never invalidates the application of any rule used in step n or earlier. We use the term *cautious* to emphasize that latter restriction.

We now have the following result.

Theorem 7
Let P be an mca-program and let M be a set of atoms. Then M is a stable model of M if and only if M is a model of P and there is a cautious P-derivation $\langle X_n \rangle_{n=0,1,\ldots}$ such that $M = \bigcup_{n \in N} X_n$.

Proof
We assume first that M is a stable model of P. Then M is a model of P. Let us consider the reduct P^M of P with respect to M and let $\langle X_n \rangle_{n=0,1,\ldots}$ be the canonical P^M-derivation of M (since M is a model of P, M is a model of P^M).

By Proposition 7, $M = \bigcup_{n=0}^{\infty} X_n$. Thus, to complete the proof of this part of the assertion, it suffices to show that $\langle X_n \rangle_{n=0,1,\ldots}$ is a cautious P-derivation.

Let X be a set of atoms. We call an mca-clause $r \in P$ *X-allowed* if $X \models \mathbf{not}(L)$, for every negated literal $\mathbf{not}(L)$ in the body of r. We denote the set of all X-allowed mca-clauses in P by $P^-(X)$. We then define:

$$P_n = P^-(M) \cap P(X_n)\cdot$$

Let $r \in P_n$. Then $r \in P^-(M)$. Since $X_{n+1} \subseteq M$, X_{n+1} satisfies all negated literals in the body of r. Moreover, since $X_n \subseteq X_{n+1}$ and $r \in P(X_n)$, X_{n+1} satisfies all other literals in the body of r, as well. Thus, $r \in P(X_{n+1})$ and, consequently, $r \in P_{n+1}$. It follows that $P_n \subseteq P_{n+1}$.

By the definition of P_n, it follows that $\mathrm{T}^{\mathrm{nd}}_{P_n}(X_n) = \mathrm{T}^{\mathrm{nd}}_{P^M}(X_n)$. Since $\langle X_n \rangle_{n=0,1,\ldots}$ is a P^M-derivation, the condition (a) of the definition of a cautious P-derivation holds. Let $r \in P_n$. Then $r \in P^-(M)$. Since $X_{n+1} \subseteq M$, the condition (b) follows. Thus $\langle X_n \rangle_{n=0,1,\ldots}$ is a cautious P-derivation.

Conversely, let us assume that M is a model of P and that for some cautious P-derivation $\langle X_n \rangle_{n=0,1,\ldots}$, $M = \bigcup_{n=0}^{\infty} X_n$. Moreover, let $\langle P_n \rangle_{n=0,1,\ldots}$ be any "witnessing" sequence of subprograms of P, guaranteed to exist by the definition of a cautious P-derivation.

Since M is a model of P, M is a model of the reduct P^M. Let $\langle Y_n \rangle_{n=0,1,\ldots}$ be the canonical P^M-derivation for M. Clearly, $\bigcup_{n=0}^{\infty} Y_n \subseteq M$ and, for every $n \geq 0$, $Y_{n+1} = hset(P^-(M)(Y_n)) \cap M$, where $P^-(M)(Y_n)$ consists of all clauses in P that are M-allowed and Y_n-applicable.

We will show that for every n, $X_n \subseteq Y_n$. We proceed by the induction. The inclusion clearly holds for $n = 0$ (both sets are empty, then). Let us assume that $X_n \subseteq Y_n$. Since $P_n \subseteq P^-(M)$ and $X_{n+1} \in \mathrm{T}^{\mathrm{nd}}_{P_n}(X_n)$, $X_{n+1} \subseteq hset(P^-(M)(X_n))$. By the induction hypothesis, $X_n \subseteq Y_n$. Thus, $hset(P^-(M)(X_n)) \subseteq hset(P^-(M)(Y_n))$. Moreover, $X_{n+1} \subseteq M$. Consequently, $X_{n+1} \subseteq hset(P^-(M)(Y_n)) \cap M = Y_{n+1}$.

We now have $M = \bigcup_{n=0}^{\infty} X_n \subseteq \bigcup_{n=0}^{\infty} Y_n \subseteq M$. Thus, $\bigcup_{n=0}^{\infty} Y_n = M$ and M is a derivable model of P^M or, equivalently, a stable model of M. □

Theorem 7 states that if we apply clauses *carefully*, making sure that at no stage we satisfy an mc-atom appearing negated in clauses applied so far (including the one selected to apply at the present stage) and we ever compute a model in this way, then this model is a stable model of P. Conversely, every stable model can be obtained as a result of such a *careful* derivation.

5 Constraints in mca-programs

Let P be an mca-program. An mca-clause in P is a *constraint* for P if it is of the form

$$1\{a\} \leftarrow L_1, \ldots, L_m, \mathbf{not}(1\{a\}), \tag{2}$$

where L_i, $1 \leq i \leq m$, are mc-literals and a is an atom that does not appear in any of the literals L_i and in no other rule of P. We commonly write a constraint of the form (2) as

$$\leftarrow L_1, \ldots, L_m.$$

We have the following result concerning supported and stable models of programs with constraints.

Proposition 13
Let P be an mca-program and let P' be the set of mca-clauses in P that are constraints for P. A set of atoms M is a *supported* (*stable*) model of P if and only if M is a supported (stable) model of $P \setminus P'$ and a model of P'.

Proof
The assertion concerning supported models is an easy consequence of Proposition 4.

Let M be a stable model of P. Then M is a model of P and so, also of P'. We next observe that M contains no atom appearing in the head of a constraint for P. Indeed, if a is an atom appearing in the head of an mca-clause $r \in P'$ and $a \in M$, then r does not contribute to the reduct P^M (r is not M-allowed). Since no mca-clause in P other than r contains a in the head, no model derivable from P^M contains a, a contradiction. Since M is a derivable model of P^M and since no rule in the reduct P^M contributed by a constraint is applied in the corresponding P^M-derivation (otherwise, the atom in the head of that constraint would belong to M), M is a derivable model of $(P \setminus P')^M$ and, consequently, a stable model of $P \setminus P'$.

Conversely, if M is a stable model of $P \setminus P'$, then M is a derivable model of $(P \setminus P')^M$. Since M is a model of P', it follows that every $(P \setminus P')^M$-derivation that has M as its result is also a P^M-derivation. Thus, M is a derivable model of P^M and, consequently, a stable model of P. □

6 Mca-programs and normal logic programming

An mc-atom $1\{a\}$ is true in a model M if and only if a is true in M. Thus, intuitively, $1\{a\}$ and a are equivalent. That suggests a way to interpret normal clauses and programs as mca-clauses and mca-programs. Let r be the following normal clause:

$$r = \quad c \leftarrow a_1, \ldots, a_m, \mathbf{not}(b_1), \ldots, \mathbf{not}(b_n).$$

By $mca(r)$ we mean the mc-clause

$$1\{c\} \leftarrow 1\{a_1\}, \ldots, 1\{a_m\}, \mathbf{not}(1\{b_1\}), \ldots, \mathbf{not}(1\{b_n\}).$$

(If all a_i and all b_i are distinct, which we can assume without loss of generality, a simpler translation, $1\{c\} \leftarrow m\{a_1, \ldots, a_m\}, \mathbf{not}(1\{b_1, \ldots, b_n\})$, could be used.) Moreover, given a normal program P, we set $mca(P) = \{mc(r) \colon r \in P\}$.

This encoding interprets normal logic programs as mca-programs so that basic properties and concepts of normal logic programming can be viewed as special cases of properties and concepts in mca-programming. In the following theorem, we gather several results establishing appropriate correspondences.

Theorem 8
Let P be a normal logic program and let M be a set of atoms.

1. P is a Horn program if and only if $mca(P)$ is a Horn mca-program.
2. If P is a Horn program then the least model of P is the only derivable model of $mca(P)$.
3. $\{\mathrm{T}_P(M)\} = \mathrm{T}^{\mathrm{nd}}_{mca(P)}(M)$.
4. $mca(P^M) = mca(P)^M$.
5. M is a model (supported model, stable model) of P if and only if M is a model (supported model, stable model) of $mca(P)$.

Theorem 8 allows us to establish the complexity of deciding whether an mca-program P has a supported (stable) model.

Corollary 6
Given a finite mca-program P, the problems to decide whether P has supported (respectively, stable) models is NP-complete.

Proof
Hardness follows by Theorem 8 directly from the fact the the corresponding problems are NP-complete for the class of normal logic programs (Marek and Truszczyński 1991). To see that the problem of existence of supported models is in the class NP, we note that, given a set of atoms M, the conditions of Proposition 4 can be verified in polynomial time. To prove that the problem of existence of stable models is in the class NP, we note that, given a set of atoms M, computing the reduct P^M can be done in polynomial time and verifying whether M is a derivable model of P^M is also a polynomial-time task (Corollary 5). □

Finally, we define a class of mca-programs, which offers a most direct extension of normal logic programming. This class of programs, in a more general first-order setting, was thoroughly studied in (Denecker et al. 2001; Pelov et al. 2004).

Definition 9
An mca-clause r is *deterministic* if $hd(r) = 1\{a\}$, for some atom a. An mca-program is *deterministic* if every clause in P is deterministic.

The intuition behind the term is clear. If the head of an mca-clause is of the form $1\{a\}$, then there is only one possible effect of applying the clause: a has to be concluded. Thus, the nondeterminism that arises in the context of arbitrary mc-atoms disappears. Formally, we capture this property in the following result.

Proposition 14
Let P be a deterministic mca-program. Then, for every set of atoms M, $\mathrm{T}^{\mathrm{nd}}_P(M) = \{M'\}$, for some set of atoms M'.

Thus, for a deterministic mca-program P, the operator $\mathrm{T}^{\mathrm{nd}}_P$ is deterministic and, so, can be regarded as an operator with both the domain and codomain $\mathcal{P}(At)$. We will write $\mathrm{T}^{\mathrm{d}}_P$, to denote it. Models, supported models and stable models of a deterministic mca-program can be introduced in terms of the operator $\mathrm{T}^{\mathrm{d}}_P$ in exactly

the same way the corresponding concepts are defined in normal logic programming. In particular, the algebraic treatment of logic programming developed in (Fitting 2002; Przymusinski 1990; Denecker et al. 2000) applies literally to deterministic mca-programs and results in a natural and direct extension of normal logic programming (Denecker et al. 2001; Pelov et al. 2004). We will explicitly mention just one result here that will be of importance later in the paper.

Proposition 15
Let P be a deterministic Horn program. Then P has exactly one derivable model and this model is the least model of P.

7 Mca-programs and ca-programs

We will first briefly review the concept of programs with cardinality atoms (Niemelä et al. 1999) and the semantics of stable models of such programs, as introduced in (Niemelä et al. 1999). We will then relate this formalism to that of mca-programs.

A *cardinality atom* (c-atom, for short) is an expression of the form kXl, where $X \subseteq At$, and l and k are integers such that $0 \leq k \leq l \leq |X|$. We call X an *atom set* of a c-atom $A = kXl$ and, as before, we denote it by $aset(A)$[1].

We say that a set of atoms M satisfies a c-atom kXl if $k \leq |M \cap X| \leq l$ ($M \models kXl$, in symbols). It is clear that when $k = 0$ or $l = |X|$, the corresponding inequality is trivially true. Thus, we omit from the notation k, if equal to 0, and l, if equal to $|X|$.

A *cardinality-atom clause* (ca-clause, for short) is an expression r of the form

$$A \leftarrow B_1, \ldots, B_n,$$

where A and B_i, $1 \leq i \leq n$, are c-atoms. We call A the head of r and $\{B_1, \ldots, B_n\}$ the *body* of r. We denote them by $hd(r)$ and $bd(r)$, respectively. A *ca-program* is a collection of ca-clauses.

We say that a set $M \subseteq At$ *satisfies* a ca-clause r if M satisfies $hd(r)$ whenever it satisfies each c-atom in the body of r. We say that M satisfies a ca-program P if M satisfies each ca-clause in P. We write $M \models r$ and $M \models P$ in these cases, respectively.

We will now recall the concept of a stable model of a ca-program (Niemelä et al. 1999). Let P be an ca-program and let $M \subseteq At$. By the *NSS-reduct* of P with respect to M we mean the ca-program obtained by:

1. eliminating from P every clause r such that $M \not\models B$, for at least one c-atom $B \in bd(r)$.
2. replacing each remaining ca-clause $r = kXl \leftarrow k_1\, Y_1\, l_1, \ldots, k_n\, Y_n\, l_n$ with all clauses of the form $1\{a\} \leftarrow k_1\, Y_1, \ldots, k_n\, Y_n$, where $a \in X \cap M$.

[1] To be precise, (Niemelä et al. 1999) allows also for negated atoms to appear as elements of X. One can eliminate occurrences of negative literals by introducing new atoms. We give further details in the end of the section.

With some abuse of notation, we denote the resulting program by P^M (the type of the program determines which reduct we have in mind). It is clear that P^M is a deterministic Horn mca-program. Thus, it has a least model, $lm(P^M)$.

Definition 10
Let P be a ca-program. A set $M \subseteq At$ is a *stable model* of P if $M = lm(P^M)$ and $M \models P$.

We will now show that the formalisms of mca-programs and ca-programs with their corresponding stable-model semantics are equivalent. We start by describing an encoding of ca-clauses and ca-programs by mca-clauses and mca-programs. To simplify the description of the encoding and make it uniform, we assume that all bounds are present (we recall that whenever any of the bounds are missing from the notation, they can be introduced back). Let r be the following ca-clause:

$$kXl \leftarrow k_1X_1l_1, \ldots, k_mX_ml_m.$$

We represent this ca-clause by a pair of mca-clauses, $e^1_{mca}(r)$ and $e^2_{mca}(r)$ that we define as

$$kX \leftarrow k_1X_1, \ldots, k_mX_m, \mathbf{not}((l_1+1)X_1), \ldots, \mathbf{not}((l_m+1)X_m),$$

and

$$\leftarrow (l+1)X, k_1X_1, \ldots, k_mX_m, \mathbf{not}((l_1+1)X_1), \ldots, \mathbf{not}((l_m+1)X_m),$$

respectively. Given a ca-program P, we translate it into an mca-program

$$e_{mca}(P) = \bigcup_{r \in P} \{e^1_{mca}(r), e^2_{mca}(r)\}.$$

Theorem 11
Let P be a ca-program. A set of atoms M is a stable model of P, as defined for ca-programs, if and only if M is a stable model of $e_{mca}(P)$, as defined for mca-programs.

Proof
In the proof we write *NSS-stable* and *mca-stable* to emphasize the notion of stability we have in mind (even though the two notions can be distinguished from the context by the type of a program, to which they are applied). We will also use the notation:

$$P^1_{mca} = \bigcup\{e^1_{mca}(r) \colon r \in P\} \quad \text{and} \quad P^2_{mca} = \bigcup\{e^2_{mca}(r) \colon r \in P\}.$$

Let us assume first that M is an NSS-stable model of a ca-program P. We will show that M is an mca-stable model of the mca-program $e_{mca}(P)$, which in our terminology is equal to $P^1_{mca} \cup P^2_{mca}$.

Since M is an NSS-stable model of P, it is a model of P (Definition 10). Consequently, it follows that M is a model of all rules in P^2_{mca}. By Proposition 13, to complete this part of the proof it suffices to show that M is an mca-stable model

of the program P^1_{mca}. To this end, we note that the definitions of the respective reducts imply that a rule

$$1\{a\} \leftarrow k_1 X_1, \ldots k_m X_m$$

belongs to the NSS-reduct of P with respect to M if and only if for some $X \subseteq At$, $a \in X \cap M$ and there is a rule

$$kX \leftarrow k_1 X_1, \ldots k_m X_m$$

in the reduct of P^1_{mca} with respect to M. We will denote the two reducts by Q and Q', respectively. From this relationship it follows that the results of the canonical derivations from Q and Q' with respect to M coincide (we recall that both reducts are Horn mca-programs). Since M is the least model of Q, it is the result of the canonical derivation from Q with respect to M. Thus, M is also the result of the canonical derivation from Q' with respect to M. In other words,, M is a derivable model of Q' and, consequently, an mca-stable model of P^1_{mca}.

Conversely, let us assume that M is an mca-stable model of $P^1_{mca} \cup P^2_{mca}$. It follows that M is a model of $P^1_{mca} \cup P^2_{mca}$ and, consequently, a model of P. Next, we note that since M is an mca-stable model of $P^1_{mca} \cup P^2_{mca}$, it is an mca-stable model of P^1_{mca} (by Proposition 13). Thus, it is a derivable model of its reduct Q' and, therefore, it is also the result of the canonical derivation from Q' with respect to M. Our observation about the relationship between the reducts Q' of P^1_{mca} and Q of P (both with respect to M) applies now, as well. Consequently, M is the result of the canonical derivation from Q with respect to M. Thus, M is a derivable model of Q. Since Q is a deterministic Horn mca-program, it has only one derivable model — its least model. It follows that M is the least model of Q and, consequently, an NSS-stable model of P. □

This theorem shows that the formalism of mca-programs is at least as expressive as that of ca-programs. The converse is true as well: ca-programs are at least as expressive as mca-programs. Let r be the following mca-clause:

$$kX \leftarrow k_1 X_1, \ldots, k_m X_m, \mathbf{not}(l_1 Y_1), \ldots, \mathbf{not}(l_n X_n)\cdot$$

We define $e_{ca}(r)$ as follows. If there is i, $1 \leq i \leq n$, such that $l_i = 0$, we set $e_{ca}(r) = \quad kX \leftarrow kX$ (in fact any tautology would do). Otherwise, we set

$$e_{ca}(r) = \quad kX \leftarrow k_1 X_1, \ldots, k_m X_m, Y_1(l_1 - 1), \ldots, Y_n(l_n - 1)\cdot$$

Given an mca-program P, we define $e_{ca}(P) = \{e_{ca}(r) \colon r \in P\}$.

Theorem 12
Let P be an mca-program. A set of atoms M is a stable model of P, as defined for mca-programs, if and only if M is a stable model of $e_{ca}(P)$, as defined for ca-programs.

Proof
First, we observe that P and $e_{ca}(P)$ have the same models. Next, similarly as before, we have that the NSS-reduct Q of $e_{ca}(P)$ contains a rule of the form

$$1\{a\}X \leftarrow k_1X_1, \ldots k_mX_m, Y_1, \ldots, Y_n$$

if and only if there is $X \subseteq At$ such that $a \in X \cap M$ and the rule

$$kX \leftarrow k_1X_1, \ldots k_mX_m$$

belongs to the reduct Q' of P with respect to M. Since in the rules of the first type mc-atoms Y_i are always true, as before, the results of the canonical derivations from Q and Q' with respect to every model M of P (or, equivalently, of $e_{ca}(P)$) coincide (we recall that both reducts are Horn mca-programs). Using this observation one can complete the proof reasoning as in the previous proof. □

Theorems 11 and 12 establish the equivalence of ca-programs and mca-programs with respect to the stable model semantics. The same translations also preserve the concept of a model. Finally, Theorem 11 suggests a way to introduce the notion of a supported model for a ca-program: a set of atoms M is defined to be a *supported* model of a ca-program P if it is a supported model of the mca-program $e_{mca}(P)$. With this definition, the two translations e_{mca} and e_{ca} also preserve the concept of a supported model. In other words, the translations e_{mca} and e_{ca} uniformly preserve several semantic notions and allow us to view ca-programs and mca-programs as syntactic variations of each other.

We also note that this equivalence demonstrates that ca-programs with the semantics of stable models as defined in (Niemelä et al. 1999) can be viewed as a generalization of normal logic programming. It follows from Theorems 8 and 12 that the encoding of normal logic programs as ca-programs, defined as the composition of the translations *mca* and e_{ca}, preserves the semantics of models, supported models and stable models (an alternative proof of this fact, restricted to the case of stable models only was first given in (Niemelä et al. 1999) and served as a motivation for the class of ca-programs and its stable-model semantics). This result is important, as it is not at all evident that the NSS-reduct and Definition 10 generalize the semantics of stable models as defined in (Gelfond and Lifschitz 1988).

Given that the formalisms of ca-atoms and mca-atoms are equivalent, it is important to stress what differs them. The advantage of the formalism of ca-programs is that it does not require the negation operator in the language. The strength of the formalism of mca-programs lies in the fact that its syntax so closely resembles that of normal logic programs, and that the development of the theory of mca-programs so closely follows that of the normal logic programming.

As we noted at the beginning of the section, (Niemelä et al. 1999) allow also for negated atoms to appear as elements of X in a cardinality atom kXl. One can eliminate occurrences of negative literals by introducing new atoms in the following way. First, for each negated literal $\mathbf{not}(b)$ appearing in the set of a c-atom, we introduce a new propositional atom $\bar{b}$. Then, for each such atom, we include a ca-clause $\bar{b}_i \leftarrow \{b_i\}0$. Finally, we replace each cardinality atom

$$k\{a_1, \ldots, a_m, \mathbf{not}(b_1), \ldots, \mathbf{not}(b_n)\}l\cdot$$

with a c-atom

$$k\{a_1, \ldots, a_m, \bar{b}_1, \ldots, \bar{b}_n\}l.$$

This transformation has the property that stable models are preserved. However, it remains an open question whether negative literals in cardinality atoms can be eliminated without introducing new atoms in the program.

8 Normal form for mca-programs

In the case of normal logic programs researchers proved several normal form theorems. A general template for such results is: for every normal logic program P there is a normal logic program P_1, in some restricted syntactic form, such that P and P_1 have the same (possibly modulo some new atoms) intended models (typically, stable, supported or both).

Here are two examples of results of this type. We recall that a normal logic program P is *purely negative* if the clauses of P have no positive atoms. Further, a logic program P is a *2-program* if the bodies of clauses in P consist of at most 2 literals.

Proposition 16 (Dung and Kanchansut 1989)
For every normal logic program P there exists a purely negative program P_1 (over the same set of atoms) such that P and P_1 have the same stable models.

Proposition 17 (Blair 1989)
For every normal program P there is a 2-program P_1 (in general, over a larger set of atoms), such that there is a one-to-one projection from the set of stable models of P_1 to the set of stable models of P. The program P_1 can be computed from P in linear time.

We know of no generalization of the result by Dung to the classes of mca- and ca-programs. The result of Blair generalizes to each of these broader settings. It is a consequence of a normal form result proved by Marek and Remmel (Marek and Remmel 2005). An mca-program (ca-program) P is *body-normal* if the bodies of its clauses are *normal*, that is, consist of atoms and negated atoms (or, to be precise, of their representations in the respective formalisms: expressions $1\{a\}$ and $\mathbf{not}(1\{a\})$, in the case of mca-programs, and expressions $1\{a\}1$ and $0\{a\}0$ in the case of ca-programs).

Proposition 18 (*Marek and Remmel 2005*)
For every mca-program (and also for every ca-program) P there exists a body-normal program P_1, over the same set of atoms, such that P and P_1 have precisely the same stable models. There also exists a body-normal program P_2 (in general, over a bigger set of atoms) that can be constructed from P in polynomial time, so that there is a one-to-one projection from the set of stable models of P_2 to the set of stable models of P.

We observe that the method used by Blair in transform arbitrary normal logic programs to 2-programs applies in the case of body-normal mca- and ca-programs. That implies that we can limit to two the number of literals in the bodies of clauses of programs P_1 and P_2 asserted in Proposition 18.

We also note that there are other normal form results that may be of some importance in the context of algorithms for computing stable models. Here are several obvious transformation techniques. If the body of a clause contains the atoms kX_1 and lX_2 where $|X_1| = k$ and $|X_2| = l$, then these two atoms can be combined into a single atom $(k + l - m)(X_1 \cup X_2)$, where $m = |X_1 \cap X_2|$. A similar transformation rule exists for c-atoms expressing upper bounds (and, consequently, for negated atoms $\mathbf{not}(kX_1)$ and $\mathbf{not}(lX_2)$).

9 Mca-programs and disjunctive logic programs

The formalism of mca-programs also extends an approach to disjunctive logic programming, proposed in (Sakama and Inoue 1994). In that paper, the authors introduced and investigated a semantics of *possible models* for disjunctive logic programs. We will now show that disjunctive programming with the semantics of possible models is a special case of the logic mca-programs with the semantics of stable models.

Let r be a disjunctive logic program clause of the form:

$$c_1 \vee \ldots \vee c_k \leftarrow a_1, \ldots, a_m, \mathbf{not}(b_1), \ldots, \mathbf{not}(b_n),$$

where all a_i, b_i and c_i are atoms. We define an mca-clause

$$mca_d(r) = \quad 1\{c_1, \ldots, c_k\} \leftarrow 1\{a_1\}, \ldots, 1\{a_m\}, \mathbf{not}(1\{b_1\}), \ldots, \mathbf{not}(1\{b_n\}).$$

(If all a_i and b_i are distinct, the following translation could be used instead: $1\{c_1, \ldots, c_k\} \leftarrow m\{a_1, \ldots, a_m\}, \mathbf{not}(1\{b_1, \ldots, b_n\})$.) For a disjunctive logic program P, we define $mca_d(P) = \{mca_d(r) \colon r \in P\}$. We have the following theorem.

Theorem 13
Let P be a disjunctive logic program. A set of atoms M is a possible model of P if and only if M is a stable model of the mca-program $mca_d(P)$.

We also note that there are strong analogies between the approach we propose here and some of the techniques discussed in (Sakama and Inoue 1994). In particular, (Sakama and Inoue 1994) presents a computational procedure for disjunctive programs without negation that is equivalent to our notion of a P-derivation. We stress however, that the class of mca-programs is more general and that our approach, consistently exploiting properties of an operator $\mathrm{T}_P^{\mathrm{nd}}$, is better aligned with a standard development of normal logic programming.

10 Discussion

Results of our paper point to a central position of mca-programs among other logic programming formalisms. First, mca-programs form a natural generalization of normal logic programs, with most concepts and techniques closely patterned after their

counterparts in normal logic programming. Second, mca-programs with the stable-model semantics generalize disjunctive logic programming with the possible-model semantics of (Sakama and Inoue 1994). Third, mca-programs provide direct means to model cardinality constraints, a feature that has become broadly recognized as essential to computational knowledge representation formalisms. Moreover, it turns out that mca-programs are, in a certain sense that we made precise in the paper, equivalent, to logic programs with cardinality atoms proposed and studied in (Niemelä et al. 1999). Thus, mca-programs provide a natural link between normal logic programs and the formalism of (Niemelä et al. 1999), and help explain the nature of this relationship, hidden by the original definitions in (Niemelä et al. 1999).

Even more, there is straightforward extension of our theory to the case of programs built of *monotone-weight atoms*, that is, expressions of the form $a\{p_1 : w_1, \ldots, p_k : w_k\}$, where a, $w_1, \ldots w_k$ are non-negative reals and $p_1, \ldots, p_k$ are propositional atoms. Intuitively, such an atom is satisfied by an interpretation (set of atoms) M if the sum of weights assigned to atoms in $M \cap \{p_1, \ldots, p_k\}$ is at least a. The theory of programs with monotone-weight atoms, that closely parallels the theory presented here, offers a theoretical account to programs with weight constraints, as introduced and studied in (Niemelä et al. 1999; Simons et al. 2002).

In this paper, we outlined the rudiments of the theory of mca-programs. There are several questions that follow from our work and that deserve more attention. For instance, there is a question whether Fages lemma (Fages 1994) generalizes to mca-programs. If so, for some classes of programs, one could reduce stable-model computation to satisfiability checking for propositional theories with cardinality atoms (East and Truszczyński 2001; Liu and Truszczyński 2003). That, in turn, might lead to effective computational methods, alternative to direct algorithms such as *smodels* (Niemelä and Simons 1996) and similar in spirit to the approach of *cmodels* (Erdem and Lifschitz 2003; Babovich and Lifschitz 2002). A related question is that of generalizing to the class of mca-programs the concepts of completion (Clark 1978) and of a loop formula (Lin and Zhao 2002), as that might lead to methods to compute stable models of mca-programs (or, more generally, programs with monotone-weight atoms) by means of pseudo-boolean satisfiability solvers (Barth 1995; Walser 1997; Aloul et al. 2002; East and Truszczyński 2004). We believe these questions can be studied in a general context of programs built of *monotone constraint atoms* (Marek and Truszczyński 2004).

The emergence of a nondeterministic one-step provability operator is particularly intriguing. It suggests that, as in the case of normal logic programming (Fitting 2002; Przymusinski 1990), the theory of mca-programs can be developed by algebraic means. For that to happen, one would need techniques for handling nondeterministic operators on lattices, similar to those presented in the deterministic operators in (Denecker et al. 2000; Denecker et al. 2002). That approach might ultimately lead to a generalization of the well-founded semantics to the case of mca-programs.

Acknowledgments

The second author was supported by the Academy of Finland grant 53695. The other two authors were supported by the NSF grants IIS-0097278 and IIS-0325063.

References

Aloul, F., Ramani, A., Markov, I., and Sakallah, K. 2002. PBS: a backtrack-search pseudo-boolean solver and optimizer. In *Proceedings of the 5th International Symposium on Theory and Applications of Satisfiability*. 346 – 353.

Babovich, Y. and Lifschitz, V. 2002. *Cmodels package*. `http://www.cs.utexas.edu/users/tag/cmodels.html`.

Barth, P. 1995. A Davis-Putnam based elimination algorithm for linear pseudo-boolean optimization. Tech. rep., Max-Planck-Institut für Informatik. MPI-I-95-2-003.

Blair, H. 1989. Normal-form results for Logic Programs. Personal Communication

Clark, K. 1978. Negation as failure. In *Logic and data bases*, H. Gallaire and J. Minker, Eds. Plenum Press, New York-London, 293–322.

Denecker, M., Marek, V., and Truszczyński, M. 2000. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In *Logic-Based Artificial Intelligence*, J. Minker, Ed. Kluwer Academic Publishers, 127–144.

Denecker, M., Marek, V., and Truszczyński, M. 2002. Ultimate approximations in nonmonotonic knowledge representation systems. In *Principles of Knowledge Representation and Reasoning, Proceedings of the 8th International Conference (KR2002)*. Morgan Kaufmann Publishers, 177–188.

Denecker, M., Pelov, N., and Bruynooghe, M. 2001. Ultimate well-founded and stable semantics for logic programs with aggregates. In *Logic programming, Proceedings of the 2001 International Conference on Logic Programming*, P. Codognet, Ed. Vol. 2237. Springer, 212–226.

Dung, P.M. and Kanchanasut, K.. 1989. On the generalized predicate completion of non-Horn programs. In: Logic programming, Proceedings of the North American Conference, MIT Press, pages 587–603.

East, D. and Truszczyński, M. 2001. Propositional satisfiability in answer-set programming. In *Proceedings of Joint German/Austrian Conference on Artificial Intelligence (KI-2001)*. LNAI, vol. 2174. Springer, 138–153.

East, D. and Truszczyński, M. 2004. Predicate-calculus based logics for modeling and solving search problems. *ACM Transactions on Computational Logic*. To appear, available at `http://www.acm.org/tocl/accepted.html`.

Erdem, E. and Lifschitz, V. 2003. Tight logic programs. *Theory and Practice of Logic Programming 3,* 4-5, 499–518.

Fages, F. 1994. Consistency of Clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science 1*, 51–60.

Fitting, M. C. 2002. Fixpoint semantics for logic programming – a survey. *Theoretical Computer Science 278*, 25–51.

Gelfond, M. and Lifschitz, V. 1988. The stable semantics for logic programs. In *Proceedings of the 5th International Conference on Logic Programming*. MIT Press, 1070–1080.

Lin, F. and Zhao, Y. 2002. ASSAT: Computing answer sets of a logic program by SAT solvers. In *Proccedings of the 18th National Conference on Artificial Intelligence (AAAI-2002)*. AAAI Press, 112–117.

Liu, L. and Truszczyński, M. 2003. Local-search techniques in propositional logic extended with cardinality atoms. In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming, CP-2003*. LNCS, vol. 2833. Springer, 495–509.

Marek, V.W., Nerode and Remmel, J.B. 1999. Logic Programs, Well-Orderings and Forward Chaining, *Annals of Pure and Applied Logic 96*, 231–276.

Marek, V.W. and Remmel, J.B. 2005. Body-mormal programs with cardinality constraints. Forthcoming.

Marek, V. and Truszczyński, M. 2004. Logic programs with abstract constraint atoms. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-04)*. AAAI Press.

Marek, W. and Truszczyński, M. 1991. Autoepistemic logic. *Journal of the ACM 38,* 3, 588–619.

Niemelä, I. and Simons, P. 1996. Efficient implementation of the well-founded and stable model semantics. In *Proceedings of JICSLP-96*. MIT Press.

Niemelä, I. and Simons, P. 2000. Extending the smodels system with cardinality and weight constraints. In *Logic-Based Artificial Intelligence*, J. Minker, Ed. Kluwer Academic Publishers, 491–521.

Niemelä, I., Simons, P., and Soininen, T. 1999. Stable model semantics of weight constraint rules. In *Proceedings of LPNMR-1999*. Lecture Notes in Computer Science, vol. 1730. Springer, 317–331.

Pelov, N., Denecker, M., and Bruynooghe, M. 2004. Partial stable semantics for logic programs with aggregates. In *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning*, V. Lifschitz and I. Niemelä, Eds. Lecture Notes in Artificial Intelligence, vol. 2923. Springer, 207–219.

Przymusinski, T. 1990. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae 13,* 4, 445–464.

Sakama, C. and Inoue, K. 1994. An alternative approach to the semantics of disjunctive logic programs and deductive databases. *Journal of Automated Reasoning 13*, 145–172.

Simons, P., Niemelä, I., and Soininen, T. 2002. Extending and implementing the stable model semantics. *Artificial Intelligence 138*, 181–234.

Walser, J. 1997. Solving linear pseudo-boolean constraints with local search. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-97)*. AAAI Press, 269–274.