A Generalized Gelfond-Lifschitz Transformation for Logic Programs with Abstract Constraints

Yi-Dong Shen
State Key Laboratory of Computer Science
Institute of Software, the Chinese Academy of Sciences
Beijing 100080, China
E-mail: ydshen@ios.ac.cn

Jia-Huai You
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
E-mail: you@cs.ualberta.ca

Abstract

We present a generalized Gelfond-Lifschitz transformation in order to define stable models for a logic program with arbitrary abstract constraints on sets (c-atoms). The generalization is based on a formal semantics and a novel abstract representation of c-atoms, as opposed to the commonly used power set form representation. In many cases, the abstract representation of a c-atom results in a substantial reduction of size from its power set form representation. We show that any c-atom \( A = (A_d, A_c) \) in the body of a clause can be characterized using its satisfiable sets, so that given an interpretation \( I \) the c-atom can be handled simply by introducing a new atom \( \theta_A \) together with a new clause \( \theta_A \leftarrow A_1, \ldots, A_n \) for each satisfiable set \( \{A_1, \ldots, A_n\} \) of \( A \). We also prove that the latest fixpoint approach presented by Son et al. and our approach using the generalized Gelfond-Lifschitz transformation are semantically equivalent in the sense that they define the same set of stable models.

Introduction

Answer set programming (ASP) has been demonstrated to be an effective knowledge representation formalism for solving combinatorial search problems arising in many application areas such as planning, reasoning about actions, diagnosis, abduction, and so on (Baral 2003). In recent years, researchers have paid a particular attention to extensions of ASP with means to model aggregate constraints like COUNT\( \{X \mid p(X)\} = 1 \) in particular, and abstract constraints on sets like \( \{\{p(a), p(b)\}, \{\{p(a)\}, \{p(b)\}\} \) in general (Calimeri et al. 2005; Dell’Armi et al. 2003; Denecker, Pelov, & Bruynooghe 2001; Elkabani, Pontelli, & Son 2004; 2005; Faber, Leone, & Pfeifer 2004; Ferraris 2005; Liu & Truszczynski 2005; Marek & Truszczynski 2004; Pelov, Denecker, & Bruynooghe 2003; Pelov & Truszczynski 2004; Son & Pontelli 2007; Son, Pontelli, & Tu 2006). Due to the presence of such constraints, the original ASP semantics, i.e. the stable model semantics (Gelfond & Lifschitz 1988) for normal logic programs, is no longer applicable. It has then become an active research subject to develop an appropriate semantics for logic programs with abstract constraints.

Under the stable model semantics, an intended model of a normal logic program is a stable model (or answer set used exchangeably in this paper), which is declaratively defined using a simple Gelfond-Lifschitz transformation (Gelfond & Lifschitz 1988). It turns out, however, that directly applying this transformation to logic programs with abstract constraints may produce unintuitive results. Therefore, different approaches have been proposed to handle abstract constraints in the last few years, each giving a different definition of stable models for such programs. These approaches can be roughly classified into three types. Let \( P \) be a logic program with abstract constraints and \( I \) an interpretation. (1) The unfolding (or translation) approach (Pelov, Denecker, & Bruynooghe 2003; Son & Pontelli 2007) finds all solutions w.r.t. \( I \) of each abstract constraint in \( P \) and then uses the solutions to transform \( P \) to a normal logic program \( P' \). \( P' \) is defined to be a stable model of \( P \) if it is a stable model of \( P' \). (2) The fixpoint (or operator-based) approach (Marek & Truszczynski 2004; Pelov & Truszczynski 2004; Son, Pontelli, & Tu 2006) constructs a fixed point \( lfp(P) \) w.r.t. \( I \) by applying some specific immediate consequence operator over \( P \) and defines \( I \) as a stable model of \( P \) if \( I = lfp(P) \). (3) The minimal model approach (Faber, Leone, & Pfeifer 2004) defines \( I \) as a stable model of \( P \) if it is a minimal model of \( P \).

In this paper, we first establish a formal semantics for abstract constraint atoms (or c-atoms), as it is the basis for defining the semantics of logic programs with abstract constraints. We then address an interesting yet critical issue in the representation of c-atoms. In the current literature as mentioned above, a c-atom is expressed as a pair \( (D, C) \) where \( D \) is a finite set of ground atoms and \( C \) is a collection of sets of atoms in \( D \). We call this a power set form representation (w.r.t. \( D \)) of c-atoms, as \( C \) may enumerate the whole power set \( 2^D \) of \( D \) (e.g., such a case occurs for all monotone c-atoms with the property that for any \( S \subseteq D \), if \( S \in C \) then all of its supersets in \( 2^D \) are in \( C \)). For instance, we may have a c-atom \( A = \{A_d, A_c\} \) where \( A_d = \{a, b, c, d\} \) and \( A_c = \{\emptyset, \{b\}, \{c\}, \{a, c\}, \{b, c\}, \{a, b, c\}\} \). We may also have a c-atom \( B = \{B_d, B_c\} \) with \( B_d = \{b_1, \ldots, b_n\} \) for some large \( n \) and \( B_c \) containing all items of \( 2^{B_d} \) except \( \emptyset \). We observe that it is this power set form representation that makes the existing approaches rather inefficient in unfolding c-atoms. Therefore, the second contribution of our work is the introduction of a succinct abstract representation of c-atoms in which a c-atom is coded with a...
substantially smaller size. The abstract form of a c-atom \(\{A_d, A_c\}\) is a pair \((A_d, A_c^*)\) where each item of \(A_c^*\) is an abstract \(W\)-prefix fixed power set of the form \(W \cup V\), which covers all items \(W \cup S\) of \(A_c\) with \(S \in 2^V\). For example, the above example c-atom \(A\) has an abstract representation with \(A_c^* = \{\emptyset \cup \{b, c\}, \{c\} \cup \{a, b\}\}. Note that \(\emptyset \cup \{b, c\}\) covers the power set of \(\{b, c\}\) and \(\{c\} \cup \{a, b\}\) covers all sets \(\{c\} \cup S\) with \(S\) being in the power set of \(\{a, b\}\). Similarly, the abstract representation of the c-atom \(B\) is \(B_c^* = \{\{b_1\} \cup \{b_2, ..., b_n\}, ..., \{b_n\} \cup \{b_1, ..., b_{n-1}\}\}\). We see a huge reduction in size from the power set form representation to the abstract representation, e.g., 6 for \(A_c\) v.s. 2 for \(A_c^*\), and \((2^n - 1)\) for \(B_c\) v.s. \(n\) for \(B_c^*\).

Our third contribution is the introduction of a generalized Gelfond-Lifschitz transformation for logic programs with c-atoms. The key idea is to characterize c-atoms using their satisfiable sets, a notion defined directly over the proposed abstract representation of c-atoms. Informally, given an interpretation \(I\), \(W\) is a satisfiable set of a c-atom \((A_d, A_c^*)\) if \(A_c^*\) contains an abstract \(W\)-prefix fixed power set \(W \cup V\) covering \(I \cap A_d\). The generalized Gelfond-Lifschitz transformation is defined declaratively in the same way as the standard Gelfond-Lifschitz transformation, where each c-atom \(A\) in the body of a clause is handled simply by introducing a special atom \(\theta_a\) together with a new clause \(\theta_a \leftarrow A_1, ..., A_n\) for each satisfiable set \(\{A_1, ..., A_n\}\) of \(A\). We will prove that the latest fixpoint approach (Son, Pontelli, & Tu 2006) and our approach using the generalized Gelfond-Lifschitz transformation are semantically equivalent in the sense that they define the same set of stable models.

Preliminaries

We consider propositional (ground) logic programs and assume a fixed propositional language with a countable set \(\Sigma\) of propositional atoms (atoms for short). Any subset \(I\) of \(\Sigma\) is called an interpretation. A literal is an atom \(A\) (a positive literal) or its negation \(\neg A\) (a negative literal). For a set \(S = \{A_1, ..., A_n\}\) of atoms, we use \(not S\) to denote \(\{not A_1, ..., not A_n\}\) and \(|S|\) to denote the size of \(S\).

An abstract constraint atom (or c-atom) \(A\) is a pair \((D, C)\) where \(D\) is a finite set of atoms in \(\Sigma\) and \(C\) is a collection of sets of atoms in \(D\), i.e., \(C \subseteq 2^D\) for convenience, we use \(A_d\) and \(A_c\) to refer to the components \(D\) and \(C\) of \(A\), respectively. In practical situations, a c-atom \(A\) expresses a constraint on the set \(A_d\) of atoms with \(A_c\) being its admissible solutions. For instance, a c-atom \((\{p(a), p(b)\}, \{\{p(a)\}, \{p(b)\}\})\) expresses the aggregate constraint \(COUNT_X\{p(X)\} = 1\), where \(X\) takes on values in \(\{a, b\}\). Since aggregate constraints can be equivalently represented by abstract constraints (Marek & Truszczynski 2004), for convenience of presentation we use aggregate constraints and c-atoms exchangeably.

A logic program \(P\) is a finite set of clauses of the form

\[A \leftarrow A_1, ..., A_m, not B_1, ..., not B_n\]

where \(A\) and each \(A_i\) and \(B_j\) are either an atom or a c-atom (*"-" is omitted when \(m = n = 0\)). Let \(r\) be a clause. We use \(head(r)\) and \(body(r)\) to refer to its head and body, respectively. When \(P\) contains no c-atoms, it is called a normal logic program. \(P\) is a positive logic program if it is a normal logic program without negative literals.

An interpretation \(I\) satisfies (1) a positive literal \(A\) if \(A \in I\), (2) a negative literal \(\neg A\) if \(A \notin I\), (3) a c-atom \(A\) if \(A_d \cap I \in A_c\), and (4) the negation \(not A\) of a c-atom \(A\) if \(A_d \cap I \notin A_c\). \(I\) satisfies a set of literals if it satisfies each literal in the set. \(I\) satisfies the body of a clause \(r\) if it satisfies all items in \(body(r)\). \(I\) satisfies a clause \(r\) if it satisfies \(head(r)\) or it does not satisfy \(body(r)\). \(I\) is a model of a logic program \(P\) if it satisfies all clauses of \(P\). \(I\) is a minimal model of \(P\) if it is a model of \(P\) and there is no proper subset of \(I\) which is also a model of \(P\). For any expression \(F\), we use \(I \models F\) to denote that \(I\) satisfies \(F\).

By definition, for any c-atom \(A = (A_d, A_c)\), its negation \(not A\) is also a c-atom \((A_d, A_c^*)\) with \(A_c^*\) being the complement of \(A_c\), i.e., \(A_c^* = 2^D \setminus A_c\). So a logic program with negated c-atoms can be transformed to a logic program free of negated c-atoms by replacing all occurrences of negated c-atoms with their respective complement c-atoms. Due to this, in the sequel we only consider logic programs without negated c-atoms in clause bodies.

Given a normal logic program \(P\) and an interpretation \(I\), the standard Gelfond-Lifschitz transformation of \(P\) w.r.t. \(I\), written as \(P^I\), is obtained from \(P\) by performing two operations: (1) remove from \(P\) all clauses whose bodies contain a negative literal \(\neg A\) such that \(I \not\models \neg A\), and (2) remove from the remaining clauses all negative literals. Since \(P^I\) is a positive logic program, it has a unique minimal model \(M\). \(I\) is defined to be a stable model of \(P\) if \(I = M\).

A Formal Definition of the Semantics of C-Atoms

In the current literature, (the meaning of) a c-atom \(A\) is interpreted by means of propositional interpretations (truth assignments) (Marek & Truszczynski 2004); i.e., an interpretation \(I\) satisfies \(A\) if \(A_d \cap I \in A_c\), and \(I\) satisfies the negation \(not A\) if \(A_d \cap I \notin A_c\). We find that such an interpretation of c-atoms can be concisely formalized using a logic formula, thus leading to a formal definition of the semantics of c-atoms.

Definition 1 Let \(A\) be a c-atom with \(A_c = \{S_1, ..., S_m\}\). The semantics of the c-atom \(A\) is defined by

\[A \equiv C_1 \lor ... \lor C_m\]

where each \(C_i\) is a conjunction \(S_i \land not (A_d \setminus S_i)\).

It is easy to prove that \(I\) satisfies \(A\) if and only if \(C_1 \lor ... \lor C_m\) is true in \(I\) and that \(I\) satisfies \(not A\) if and only if \(not (C_1 \lor ... \lor C_m)\) is true in \(I\).

With this semantics, we can apply standard mathematic logic rules to simplify c-atoms in order to understand the characteristics of a c-atom more clearly. For example, when \(A = (\{a, b, c\}, \{a, b, \{a, b, c\}\})\), its semantics is \(A \equiv (a \land b \land not c) \lor (a \land b \land c)\), which can be logically simplified to \(A \equiv a \land b\).

An Abstract Representation of C-Atoms

In this section, we present a data structure for an abstract representation of c-atoms. We begin by introducing a notion
of prefixed power sets.

**Definition 2** Let \( I = \{a_1, ..., a_m\} \) and \( J = \{b_1, ..., b_n\} \) \((m, n \geq 0)\) be two sets of atoms.

1. The **I-prefix** of \( J \) is a collection \( \{I \cup J_{\text{sub}}, J_{\text{sub}} \subset 2^J\} \); i.e., each set in the collection consists of all \( a_i \)'s in \( I \) plus zero or more \( b_j \)'s in \( J \). We use \( I \cup J \) as an abstract form to compactly represent the \( I \)-prefix of \( J \) and for any set \( S \) of atoms, we say \( S \) is covered by \( I \cup J \) (or \( I \cup J \) covers \( S \)) if \( I \subseteq S \) and \( S \cap J \subseteq J \).

2. For any two abstract prefixed power sets \( I \cup J \) and \( I_1 \cup J_1 \), \( I \cup J \) is included in \( I_1 \cup J_1 \) if any set covered by \( I \cup J \) is covered by \( I_1 \cup J_1 \).

**Theorem 1** When \( I \cup J \) is included in \( I_1 \cup J_1 \), we have \( I_1 \subseteq I \) and \( I \cup J \subseteq I_1 \cup J_1 \). If \( I \cup J \) is included in \( I_1 \cup J_1 \) and \( I_1 \cup J_1 \) is included in \( I \cup J \), then \( I \cup J \) is included in \( I_2 \cup J_2 \).

**Definition 3** Let \( A \) be a c-atom and \( S \in \mathbb{A}_c \). The **collection of abstract S-prefix** power sets of \( A \) is \( \{S \cap S_1, ..., S \cap S_n\} \) such that for any \( S_1 \subseteq A \), \( S \cap S_1 \) is in the collection if and only if all sets covered by \( S \cap S_1 \) are in \( A \), and there is no atom \( a \in A \setminus (S \cap S_1) \) such that \( S \cap S_1 \cup \{a\} \) has the following property.

For instance, consider a c-atom \( A = \{a, b, c, d\} \) and \( A_c = \{\emptyset, \{b\}, \{c\}, \{a, c\}, \{b, c\}, \{c, d\}, \{a, b, c\}, \{b, c, d\}\} \). For \( \emptyset \in A_c \), the collection of abstract \( \emptyset \)-prefix power sets of \( A \) is \( \emptyset, \emptyset \cup \{b\}, \emptyset \cup \{c\}, \emptyset \cup \{a, c\}, \emptyset \cup \{b, c\} \). For \( \emptyset \notin A_c \), the collection is \( \emptyset \cup \{b\}, \emptyset \cup \{c\}, \emptyset \cup \{a, c\}, \emptyset \cup \{b, c\} \). Note that \( \emptyset \cup \{b\} \notin A_c \).

**Definition 4** The **abstract representation** of a c-atom \( A \) is a pair \((A_d, A_c^*)\) where \( A_d^* \) is the collection \( \bigcup_{S \in \mathbb{A}_c} C_S \), where \( C_S \) is the collection of abstract S-prefix power sets of \( A \), with all such abstract prefix power sets removed that are included in some other ones.

Consider the above example c-atom \( A \) again. Its abstract representation is \((A_d, A_c^*)\) where \( A_d^* = \emptyset \cup \{b, c\}, \emptyset \cup \{a, b\}, \emptyset \cup \{c\} \cup \{b, d\} \).

Note that the purpose of our development of an abstract representation is to substantially reduce the coding size of c-atoms by compactly compressing all power set items in \( A_c \). We say that \( A_c^* \) is power set free if for no non-empty subset \( V \) of \( A_d \), for some \( J_1, J_2 \subseteq A_d \) \( A_c^* \) contains either (i) all abstract prefixed power sets of the form \( J_1 \cup S \cup J_2 \) or (ii) all of the form \( J_1 \cup S \cup J_2 \), where \( S \in 2^J \), or (iii) all of the form \( J_1 \cup S_1 \cup S_2 \), where \( S_1 \cup S_2 \subset 2^J \).

**Theorem 2** Let \( A = (A_d, A_c) \) be a c-atom. (1) A has a unique abstract form \((A_d^*, A_c^*)\). (2) For any interpretation \( I \), \( I \models A \) if and only if \( A_c^* \) has an abstract prefixed power set \( W \) covering \( I \cap A_d \). (3) \( A_c^* \) is power set free.

The proof of (1) and (2) is routine. To prove (3), assume, on the contrary, that for some \( J_1, J_2 \subseteq A_d \) there is a non-empty subset \( V \) of \( A_d \) such that \( A_c^* \) satisfies one of the above three conditions, (i), (ii) or (iii). We show that each of the three cases introduces a contradiction. Assume that case (i) holds. Then all sets covered by \( J_1 \cap J_2 \cup V \) must be in \( A_c \).

We distinguish between two cases. (a) \( J_1 \cap J_2 \cup V \) is in \( A_c^* \). By Definition 4, for no \( S \in 2^V \), \( J_1 \cup S \cup J_2 \) would be in \( A_c^* \) (since it is included in \( J_1 \cap J_2 \cup V \)), a contradiction. (b) \( J_1 \cap J_2 \cup V \) is not in \( A_c^* \). By Definition 3, there must be a subset \( W \) of \( A_d \) with \( W \cap V \) such that \( J_1 \cap J_2 \cap W \) is in \( A_c^* \). Again, for no \( S \in 2^V \), \( J_1 \cup S \cup J_2 \) would be in \( A_c^* \) (since it is included in \( J_1 \cap J_2 \cup W \)), a contradiction. Now assume that either case (ii) or case (iii) holds. Since \( V \) is not empty, in either case \( A_c^* \) must contain two abstract prefixed power sets with one included in the other, a contradiction to the condition of Definition 4. Therefore, we conclude that \( A_c^* \) is power set free.

By Theorem 2, to check \( I \models A \) it suffices to search \( A_c^* \) instead of \( A_c \). For an abstract power set \( W \) covering \( I \cap A_d \), the time for this search is linear in the size of \( A_c^* \) which in many cases would be substantially smaller than \( A_c \). (In the most extreme case where \( A_c = 2^A \), \( A_c^* \) consists of only one item, \( \emptyset \cup A_d \).

**A Generalization of the Gelfond-Lifschitz Transformation**

Based on the formal semantics and the proposed abstract representation of c-atoms, in this section we introduce a novel generalization of the Gelfond-Lifschitz transformation for logic programs with abstract constraints. In the sequel, given an interpretation \( I \), for any c-atom \( A \) we use \( T_A \) to denote \( I \cap A_d \) and \( F_A \) to denote \( A_d \setminus T_A \).

**Definition 5** Let \( A \) be a c-atom and \( I \) an interpretation with \( I \models A \). \( S \subseteq T_A \) is a **satisfiable set** of \( A \) w.r.t. \( T_A \) if \( A_c^* \) contains an S-prefix power set \( S \cup S_1 \) covering \( T_A \).

Satisfiable sets have the following property.

**Theorem 3** Let \( A \) be a c-atom and \( I \) an interpretation. If \( S \) is a satisfiable set, then for every \( S' \) with \( S \subseteq S' \subseteq T_A \), we have \( S' \in A_c \).

Applying this theorem to the semantics of c-atoms (see Definition 1) leads to the following principal result.

**Theorem 4** Let \( A \) be a c-atom and \( I \) an interpretation. Assume that \( A \) has in total \( N \) satisfiable sets \( J_1, ..., J_N \). Then, given \( I \), \( A \) can be characterized by the set of satisfiable sets; i.e., when assuming not \( F_A \), we have \( A \equiv J_1 \cup ... \cup J_N \).

Theorem 4 lays a solid basis on which the standard Gelfond-Lifschitz transformation can be generalized to logic programs with c-atoms. In the following, we will use a special atom \( \bot \) and two special atoms, \( \theta_A \) and \( \beta_A \), for each c-atom \( A \). Unless otherwise stated, we assume that such special atoms will not occur in any given logic programs and interpretations.

**Definition 6** Given a logic program \( P \) and an interpretation \( I \), the **generalized Gelfond-Lifschitz transformation** of \( P \) w.r.t. \( I \) written as \( P^I \), is obtained from \( P \) by performing the following four operations:
1. Remove from $P$ all clauses whose bodies contain either a negative literal not $A$ such that $I \not \models A$ or a c-atom $A$ such that $I \not \models A$.

2. Remove from the remaining clauses all negative literals, and then

3. Replace each c-atom $A$ in the body of a clause with a special atom $\theta_A$ and introduce a new clause $\theta_A \leftarrow A_1, \ldots, A_m$ for each satisfiable set $\{A_1, \ldots, A_m\}$ of $A$ w.r.t. $T_A$.

4. Replace each c-atom $A$ in the head of a clause with $\bot$ if $I \not \models A$, or replace it with a special atom $\beta_A$ and introduce a new clause $B \leftarrow \beta_A$ for each $B \in T_A$ and a new clause $\bot \leftarrow B, \beta_A$ for each $B \in F_A$.

In the first operation, we remove all clauses whose bodies are not satisfied in $I$ because of the presence of a negative literal or a c-atom that is not satisfied in $I$. In the second operation, we remove all negative literals because they are satisfied in $I$. The last two operations transform c-atoms in the body and head of each clause, respectively. Each c-atom $A$ in the body of a clause is substituted by a special atom $\theta_A$, which is defined by the satisfiable sets of $A$ (based on Theorem 4). Note that each c-atom $A$ in the head of a clause represents a conclusion that every $B \in T_A$ is true and every $B \in F_A$ is false. Therefore, when $I \models A$, we substitute $A$ with a special atom $\beta_A$ and define $\beta_A$ using a clause $B \leftarrow \beta_A$ for each $B \in T_A$ and a clause $\bot \leftarrow B, \beta_A$ for each $B \in F_A$. $\bot$ is a special atom meaning false. When $I \not \models A$, we replace $A$ with $\bot$.

Apparently, the generalized Gelfond-Lifschitz transformation coincides with the standard Gelfond-Lifschitz transformation when $P$ is a normal logic program.

Since the generalized transformation $P^I$ is a positive logic program, we define the stable model semantics of a logic program with c-atoms in the same way as that of a normal logic program.

**Definition 7** For any logic program $P$, an interpretation $I$ is a stable model of $P$ if $I = M \setminus \{\theta_X, \beta_X\}$ where $M$ is the minimal model of the generalized Gelfond-Lifschitz transformation $P^I$.

Again, stable models of $P$ under the generalized Gelfond-Lifschitz transformation coincides with stable models under the standard Gelfond-Lifschitz transformation (Gelfond & Lifschitz 1988) when $P$ is a normal logic program. In the following, unless otherwise stated, by stable models we refer to stable models under the generalized Gelfond-Lifschitz transformation.

**Theorem 5** Any stable model $M$ of $P$ is a model of $P$.

A stable model may not be a minimal model for some logic programs. Assume that $P$ consists of one single clause $\{a\}, \{\emptyset \cup \{a\}\}$. The c-atom in the clause head expresses a constraint that is true if $a$ is true or $a$ is false. As a result, $P$ has two stable models, $I = \emptyset$ and $I_1 = \{a\}$. We see that $I_1$ is not minimal.

**Theorem 6** Let $P$ be a logic program with $n$ different c-atoms and $I$ an interpretation. Let $A$ be a c-atom such that $I \models A$.

1. The time complexity of computing all satisfiable sets of $A$ w.r.t. $T_A$ is linear in the size of $A_2^*$.

2. The time complexity of the generalized Gelfond-Lifschitz transformation is bounded by $O(|P| + n * (2M_A^* + M_A))$, where $M_A^*$ and $M_A$ are the maximum sizes of $A_2^*$ and $A_2$ of a c-atom in $P$, respectively.

**Theorem 7** The size of $P^I$ is bounded by $O(|P| + n * (M_A + M_A))$.

**Example 1** Consider the following logic program:

$$
P_1: \quad p(a) \leftarrow COUNT\{X[p(X)]\} > 0.
\quad p(b) \leftarrow not q.
\quad q \leftarrow not p(b).
\quad COUNT\{X[p(X)]\} = 1 \leftarrow p(b).
$$

The aggregate constraint $COUNT\{X[p(X)]\} > 0$ is a c-atom $A$ with $A_d = \{p(a), p(b)\}$ and $A_e = \{p(a), p(b)\}, \{p(a), p(b)\}$. Its abstract form is $(A_d, A^*_2)$ with $A^*_2 = \{p(a)\} \cup \{p(b)\}, \{p(b)\} \cup \{p(a)\}$. The constraint $COUNT\{X[p(X)]\} = 1$ in the head of the last clause is a c-atom $A_1 = \{\{p(a), p(b)\}\}, \{\{p(a), p(b)\}\}$, whose abstract form is $\{\{p(a), p(b)\}\}, \{\{p(a), p(b)\}\} \cup \emptyset, \{p(b)\} \cup \emptyset)$. We use three different interpretations to illustrate the generalized Gelfond-Lifschitz transformation.

1. Let $I_1 = \{p(a), q\}$. In the first operation, the second clause is removed. In the second operation, not $p(b)$ is removed from the third clause. In the third operation, we see $I_1 \models A$ with $T_A = I_1 \cap A_d = \{p(a)\}$. $T_A$ is covered by $\{p(a)\} \cup \{p(b)\}$, so $p(a)$ is the only satisfiable set of $A$. We replace $A$ (i.e., the aggregate constraint $COUNT\{X[p(X)]\} > 0$ in the first clause of $P_1$) with a special atom $\theta_A$ and introduce a new clause $T_A \leftarrow p(a)$.

Since $I_1 \models A$ with $T_A = I_1 \cap A_d = \{p(a)\}$, in the fourth operation, we replace $A$ in the head of the last clause with a special atom $\beta_A$ and introduce two new clauses $p(a) \leftarrow \beta_A$ and $\bot \leftarrow p(b), \beta_A$. Consequently, we obtain the following generalized Gelfond-Lifschitz transformation:

$$
P^I_{1}: \quad p(a) \leftarrow \theta_A.
\quad \theta_A \leftarrow p(a).
\quad q \leftarrow not p(b).
\quad \beta_A \leftarrow p(b).
\quad p(a) \leftarrow \beta_A.
\quad \bot \leftarrow p(b), \beta_A.
$$

Obviously, $I_1$ is not a stable model of $P_1$, as the minimal model of $P^I_{1}$ is $\{q\}$.

2. Let $I_2 = \{p(a), p(b)\}$. In the first operation, the third clause is removed. In the second operation, not $q$ is removed from the second clause. In the third operation, since $I_2 \models A$ with $T_A = I_2 \cap A_d = \{p(a), p(b)\}$, $A$ has two satisfiable sets w.r.t. $T_A$: $\{p(a)\} and \{p(b)\}$. The aggregate constraint in the first clause is then replaced by a special atom $\theta_A$ and $\theta_A$ is defined by two new clauses $\theta_A \leftarrow p(a)$ and $\theta_A \leftarrow p(b)$. Since $I_2 \not \models A_1$, in the fourth operation we replace the c-atom $A_1$ in the head of the last clause with $\bot$. Consequently, we obtain:

$$
P^I_{2}: \quad p(a) \leftarrow \theta_A.
\quad \theta_A \leftarrow p(a).
$$
3. Let $I$ w.r.t. $A$ two sets of atoms. The set such that Definition 9 ((Son, Pontelli, & Tu 2006)) of c-atoms with no negation.

head that positive basic logic program the following, by a satisfaction a fixpoint definition of stable models for logic programs with Most recently, Son et al. (Son, Pontelli, & Tu 2006) propose Theorem 8

dition and satisfiable sets.

Theorem 9 Let $P$ be a positive basic logic program and $I$ an interpretation. $I$ is a stable model under Son et al.'s fixpoint definition if and only if it is a stable model derived from the generalized Gelfond-Lifschitz transformation.

For a positive basic logic program, any stable model under Son et al.'s fixpoint definition proves to be minimal (Son, Pontelli, & Tu 2006). The following result is then immediate from Theorem 9.

Corollary 1 For a positive basic logic program $P$, any stable model of $P$ derived from the generalized Gelfond-Lifschitz transformation is a minimal model of $P$.

When the head $A$ of a clause $r$ is not an elementary c-atom, (Son, Pontelli, & Tu 2006) transform $r$ into the following set of clauses given an interpretation $I$:

$B ← body(r)$, for each $B ∈ T_A$

$⊥ ← B, body(r)$, for each $B ∈ F_A$

Note that in our generalized Gelfond-Lifschitz transformation, $r$ is transformed into the following set of clauses:

$β_A ← body(r)$,

$B ← β_A$, for each $B ∈ T_A$

$⊥ ← B, β_A$, for each $B ∈ F_A$

It is easy to see that the two transformations are semantically equivalent, although ours would be simpler when $body(r)$ consists of more than one item or when $A$ appears in the heads of more than one clause.

It is worth pointing out that our approach to handling c-atoms can easily be extended to disjunctive logic programs with c-atoms. Let $I$ be an interpretation and $r$ a disjunctive clause of the form

$A_1 ∨ ... ∨ A_n ← body(r)$

where each $A_i$ is a c-atom. Assume that there are totally $k > 0 A_i$s satisfied in $I$. We then introduce a special atom $β_A_i$, for each $A_i$ such that $I |= A_i$ and transform $r$ into the following clauses:

$β_A_i ∨ ... ∨ β_A_n ← body(r)$,

$B ← β_A_i$, for each $β_A_i$ and each $B ∈ T_{A_i}$

$⊥ ← B, β_A_i$, for each $β_A_i$ and each $B ∈ F_{A_i}$

When none of the $A_i$s is satisfied in $I$, we transform $r$ into the clause $⊥ ← body(r)$.

Related Work

The notion of c-atoms is introduced in (Marek & Truszczynski 2004) and further developed in (Liu & Truszczynski 2005; Marek, Niemela, & Truszczynski 2007; Son & Pontelli 2007; Son, Pontelli, & Tu 2006). In this paper, we establish a formal semantics for c-atoms. As far as we can determine, all existing approaches use a power set form $(A_d, A_c)$. 487
where \( A_c \subseteq 2^{A_t} \), to represent an arbitrary c-atom \( A \). It is quite infeasible, if not impossible, to practically store and handle c-atoms of this form. We address this critical issue by introducing a novel abstract structure \( (A_d, A^*_c) \), where \( A^*_c \) would be substantially smaller than \( A_c \), if it is power set free. We generalize the standard Gelfond-Lifschitz transformation to logic programs with c-atoms based on the formal semantics and this abstract representation.

Representative unfolding approaches to handling c-atoms include (Pelov, Denecker, & Bruynooghe 2003; Son & Pontelli 2007), where a notion of aggregate solutions (or solutions) is introduced. Informally, a solution of a c-atom \( A = (A_d, A_c) \) is a pair \( (S_1, S_2) \) of disjoint sets of atoms such that for every interpretation \( I \), if \( S_1 \subseteq I \) and \( S_2 \cap I = \emptyset \) then \( I \models A \). Let \( I \) be an interpretation and \( r \) a clause of the form \( B \leftarrow A_1, ..., A_m \), where each \( A_i \) is a c-atom. Assume that each \( A_i \) has \( n_i \) solutions w.r.t. \( I \). An unfolding approach will transform \( r \) into \( n_1 \times \ldots \times n_m \) new clauses of the form \( B \leftarrow A_1, ..., A_m \), where each \( A_i \) is built from a solution of \( A_i \) w.r.t. \( I \). Our work significantly differs from this. We show that each c-atom \( A_i \) in \( \text{body}(r) \) can be characterized by its satisfiable sets, so that \( A_i \) can be handled simply by introducing a special atom \( \theta_{A_i} \) together with a new clause \( \theta_{A_i} \leftarrow D_1, ..., D_n \) for each satisfiable set \( \{D_1, ..., D_n\} \) of \( A_i \) w.r.t. \( I \). That is, our approach transforms \( r \) into \( 1 + n_1 + \ldots + n_m \) new clauses where \( n_i \) is the number of satisfiable sets of \( A_i \) w.r.t. \( T_A \). In general, for each \( i \) we have \( n_i \gg n_i' \).

Representative fixpoint approaches include (Marek & Truszczynski 2004; Pelov & Truszczynski 2004; Marek, Niemela, & Truszczynski 2007; Son, Pontelli, & Tu 2006). (Son, Pontelli, & Tu 2006) can handle arbitrary c-atoms, while the others apply only to monotone c-atoms. In Theorem 9, we prove that Son et al.’s fixpoint definition and our definition of stable models using the generalized Gelfond-Lifschitz transformation are semantically equivalent.

(Faber, Leone, & Pfeifer 2004) propose a minimal model approach. To check if an interpretation \( I \) is a stable model of \( P \), they first remove all clauses in \( P \) that have a negative literal not \( A \) in their bodies such that \( A \in I \), and then check if \( I \) is a minimal model of the simplified program. They consider disjunctive logic programs whose clause heads are a disjunction of atoms. As we illustrated earlier, a logic program whose clause heads are arbitrary c-atoms may have non-minimal stable models under Son et al.’s fixpoint definition (or equivalently under our definition using the generalized Gelfond-Lifschitz transformation).

Conclusions
We have presented a formal semantics and a novel abstract representation of c-atoms and developed a generalized Gelfond-Lifschitz transformation for logic programs with arbitrary c-atoms. We transform a logic program with c-atoms into a positive logic program in the same way as the standard Gelfond-Lifschitz transformation for normal logic programs, where each c-atom in clause bodies is characterized by its satisfiable sets. We also proved that the latest fixpoint approach (Son, Pontelli, & Tu 2006) and our approach using the generalized Gelfond-Lifschitz transformation are semantically equivalent.

Acknowledgments
This work is supported in part by NSERC and NSFC grants 60673103, 60421001 and 60373052.

References


Elkabani, I.; Pontelli, E.; and Son, T. C. 2004. Smodels with clp and its applications: A simple and effective approach to aggregates in asp. In ICLP’04, 73–89.


Son, T. C.; Pontelli, E.; and Tu, P. H. 2006. Answer sets for logic programs with arbitrary abstract constraint atoms. In AAAI’06.