Technical Communications of the International Conference on Logic Programming, 2010 (Edinburgh), pp. 94–103
http://www.floc-conference.org/ICLP-home.html

SAMPLER PROGRAMS: THE STABLE MODEL SEMANTICS OF ABSTRACT CONSTRAINT PROGRAMS REVISITED

TOMI JANHUNEN

Aalto University School of Science and Technology
Department of Information and Computer Science
PO Box 15400, FI-00076 Aalto, Finland
E-mail address: Tomi.Janhunen@tkk.fi

Abstract. Abstract constraint atoms provide a general framework for the study of aggregates utilized in answer set programming. Such primitives suitably increase the expressive power of rules and enable more concise representation of various domains as answer set programs. However, it is non-trivial to generalize the stable model semantics for programs involving arbitrary abstract constraint atoms. For instance, a nondeterministic variant of the immediate consequence operator is needed, or the definition of stable models cannot be stated directly using primitives of logic programs. In this paper, we propose sampler programs as a relaxation of abstract constraint programs that better lend themselves to the program transformation involved in the definition of stable models. Consequently, the declarative nature of stable models can be restored for sampler programs and abstract constraint programs are also covered if decomposed into sampler programs. Moreover, we study the relationships of the classes of programs involved and provide a characterization in terms of abstract but essentially deterministic computations. This result indicates that all nondeterminism related with abstract constraint atoms can be resolved at the level of program reduct when sampler programs are used as the intermediate representation.

1. Introduction

The stable model semantics [Gel88] of logic programs, also known as the answer set semantics, constitutes the semantical cornerstone of *answer set programming* (ASP). Undoubtedly, the simple and intuitive definition of stable models [Lif08] has played a major role in the success of ASP during the past two decades. Applications that emerged in the meantime demonstrate that knowledge engineers have easily grasped the essentials of rules subject to stable models. Nevertheless, the practise of ASP has led to a rich body of extensions to the basic syntax of *normal logic programs* such as strong negation [Gel90], disjunctions [Gel91], and various kinds of *aggregates*, which have also appeared in other similar disciplines. Extensions in the last category typically enable concise expression of a particular combinatorial condition involving a set of atoms or objects. Examples of aggregates supported by contemporary ASP solvers include the *cardinality* and *weight constraints* [Sim99] and the *sum*, *count*, and *max* aggregates [Del03]. To get a concrete idea of their power,

1998 ACM Subject Classification: I.2.4, F.4.1.
Key words and phrases: stable models, abstract constraints, program reduction, translation, choice rules.
This research has been partially funded by the Academy of Finland under project #122399.

Technical Communications of the 26th International Conference on Logic Programming, Edinburgh, July, 2010
Editors: Manuel Hermenegildo, Torsten Schaub
LIPIcs - Leibniz International Proceedings in Informatics. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/LIPIcs.ICLP.2010.94

consider a sum aggregate $500 \leq \mathsf{sum}\{Y : \mathsf{capacity}(X,Y) : \mathsf{in}(X) : \mathsf{disk}(X)\}$ formalizing the sufficiency of disk space selected for a particular PC configuration. Such an aggregate requires no updates when the number of disks for configuring PCs is changed.

The study of aggregates has recently lifted to the level of *abstract constraint atoms* which nicely capture a variety of important aggregates. However, it is non-trivial to generalize stable models for arbitrary abstract constraint atoms as witnessed by the number of proposals in this respect. Only the interpretation of special cases, viz. *monotone* [Mar08] and *convex* abstract constraints [Liu06], is unanimous. As regards the general case, abstract *computations*, i.e., sequences of interpretations associated with a program, have been proposed as a semantical basis [Liu10]. A key justification is that *persistent* deterministic computations essentially capture stable models of *normal logic programs*. This interconnection suggests an alternative way of defining the semantics of abstract constraint programs, but computations bring along nondeterminism and other degrees of freedom that pre-empt a conclusive semantical definition. Besides, the declarative nature of stable models is jeopardized because the outcome of a computation potentially depends on the entire sequence.

Our hypothesis is that abstract constraint programs lack a natural counterpart of the Gelfond-Lifschitz reduct [Gel88] which plays a key role in the definition of stable models. For instance, in case of a normal logic program P, a stable model M is defined as the *least model* of P^M, i.e., the program P reduced with respect to M. Attempts to generalize this idea for abstract constraint programs become intricate because the reduced program cannot be directly represented as an abstract constraint program. For instance, the representation proposed in [She07] requires new atoms and it effectively produces a positive *normal program*. Alternatively, propositional (default) logic has been used to formalize the reduct [She09a].

In this paper, we address the aforementioned deficiency related to reducts by proposing a completely new class of programs, viz. *sampler programs*. They form a relaxation of abstract constraint programs so that a reasonable notion of a reduct can be established within the class of sampler programs. This class of programs is introduced as follows. First, in Section 2, we recall a much simpler class of *choice logic programs* [Soi99] designed for modelling *product configurations*. The syntax is based on a slight extension of normal rules—enabling a straightforward generalization of stable models. Nevertheless, it provides us with insights into how stable models can be lifted to the case of sampler programs as carried out in Section 3. The relationship of sampler programs and choice logic programs is then explored in terms of translations in Section 4. The translations presented in this case indicate that choice logic programs and sampler programs are equally expressive [Jan06].

In the second part of this paper, we apply the theory developed for sampler programs to abstract constraint programs, which are first recalled in Section 5. The idea is to decompose abstract constraint atoms into sets of samplers systematically. In this way, we are able to define stable models in the context of abstract constraint programs in a traditional way [Gel88] using the notions of a program reduct and the least model. This aspect restores the declarative nature of stable models and makes our approach original due to its simplicity. The semantics obtained in this way coincides with the one proposed in [She07, She09a, She09b]. Moreover, motivated by the computation-based approach [Liu10], we propose a notion of *canonical computations* for abstract constraint programs in Section 6. The novelty is that all nondeterminism can be handled globally via the definition of stable models, and resorting to nondeterministic variants of the immediate consequence operator or *conditional satisfaction* [Son07b] can be avoided altogether. A comparison with related work is carried out in Section 7. Finally, we present our conclusions in Section 8.

2. Choice Logic Programs (CLPs)

In this section, we introduce the class of *choice logic programs* [Soi99] that suit well for nondeterministic specifications. A choice logic program (CLP) P is a set of *choice rules*

$$a_1 \mid \ldots \mid a_l \leftarrow b_1, \ldots, b_m, {\sim}c_1, \ldots, {\sim}c_n \tag{2.1}$$

where $l \geq 0$, $m \geq 0$, $n \geq 0$, and $a_1, \ldots, a_l$, $b_1, \ldots, b_m$, and $c_1, \ldots, c_n$ are *propositional atoms*, or just *atoms* for short. A rule is *normal*, iff $l = 1$, and an *integrity constraint* (IC), iff $l = 0$. A *fact* is a normal rule with $l = 1$, $m = 0$, and $n = 0$, and thus written briefly as $a_1 \leftarrow$.

The *signature* of a CLP P, denoted by $\mathrm{At}(P)$, is the set of atoms appearing in its rules. An *interpretation* $I \subseteq \mathrm{At}(P)$ of P determines which atoms of $\mathrm{At}(P)$ are *true* ($a \in I$) and which *false* ($a \notin I$). A *positive literal* b is satisfied in an interpretation, denoted $I \models b$, iff $b \in I$. A *negative literal* ${\sim}c$ is satisfied in I, denoted $I \models {\sim}c$, iff $c \notin I$. A conjunction $l_1, \ldots, l_n$ of literals is satisfied in I, denoted $I \models l_1, \ldots, l_n$, iff $I \models l_1$, $\ldots$, and $I \models l_n$. A disjunction $a_1 \mid \ldots \mid a_l$ of atoms is satisfied in I, denoted $I \models a_1 \mid \ldots \mid a_l$, iff $I \models a_i$ holds for some $i \in \{1, \ldots, l\}$. A choice rule of the form (2.1) is satisfied in I iff $I \models b_1, \ldots, b_m$ and $I \models {\sim}c_1, \ldots, {\sim}c_n$ imply $I \models a_1 \mid \ldots \mid a_l$. An interpretation $M \subseteq \mathrm{At}(P)$ for a CLP P is called a *model* of P, denoted by $M \models P$, iff every choice rule (2.1) of P is satisfied by M.

Definition 2.1 (Reduct [Soi99])**.** The reduct P^M of a CLP P with respect to an interpretation $M \subseteq \mathrm{At}(P)$ contains a positive rule $a \leftarrow b_1, \ldots, b_m$ for each choice rule (2.1) such that (i) $a \in \{a_1, \ldots, a_l\}$, (ii) $M \models a$, and (iii) $M \models {\sim}c_1, \ldots, {\sim}c_n$.

The reduced program P^M is a *positive* normal program having rules of the form $a \leftarrow b_1, \ldots, b_m$ where $m \geq 0$. Such a program P has a unique $\subseteq$-minimal model, also known as the *least model* of P hereafter denoted by $\mathrm{LM}(P)$. Stable models are defined as follows.

Definition 2.2 (Stable Model [Soi99])**.** An interpretation $M \subseteq \mathrm{At}(P)$ is a stable model of a CLP P iff $M \models P$ and $M = \mathrm{LM}(P^M)$. The set of stable models of P is denoted by $\mathrm{SM}(P)$.

This definition coincides with [Gel88] when $l = 1$ for every rule (2.1). Moreover, any ICs contained in P do not contribute to P^M and their satisfaction is enforced by the condition $M \models P$ above. In fact, this condition implies $M \models P^M$, and thus also $\mathrm{LM}(P^M) \subseteq M$, but the converse does not hold in general. Consider, e.g., the CLP $P = \{\leftarrow b, {\sim}c\}$ and $M = \{b\}$.

Example 2.3. We note that $P = \{a \mid b \mid c \leftarrow {\sim}d\}$ has seven stable models $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$, $\{a, c\}$, $\{b, c\}$, $\{a, b, c\}$. To verify the last but one model, i.e., $M = \{b, c\}$, we observe that $M \models P$ and $P^M = \{b \leftarrow;\ \ c \leftarrow\}$[1] so that $\mathrm{LM}(P^M) = \{b, c\}$ coincides with M.

Let us stress that a *choice rule* $\{a_1, \ldots, a_l\} \leftarrow b_1, \ldots, b_m, {\sim}c_1, \ldots, {\sim}c_n$ in the style of SMODELS [Sim02] can be captured with $a_1 \mid \ldots \mid a_l \mid e \leftarrow b_1, \ldots, b_m, {\sim}c_1, \ldots, {\sim}c_n$ and $e \leftarrow$.

3. Sampler Programs (SPs)

Our next objective is to develop the theory of *sampling atoms*, or *samplers* for short, and to propose a completely new class of logic programs based on them.

Definition 3.1 (Sampler)**.** A *sampling atom*, or a *sampler* for short, π is a triple $\langle D, L, G \rangle$ where the *domain* $\pi_{\mathbf{D}} = D$ of π is a *finite* set of atoms and $L \subseteq G \subseteq D$. The sets L and G are the *least* and the *greatest satisfier* of π, also denoted by $\pi_{\mathbf{L}}$ and $\pi_{\mathbf{G}}$, respectively.

[1] For clarity, semicolons are used to separate rules in programs.

The basic intuition behind a sampler π is that it provides a compact representation for the set of literals $\{a \mid a \in \pi_{\mathbf{L}}\} \cup \{\sim a \mid a \in \pi_{\mathbf{D}} \setminus \pi_{\mathbf{G}}\}$. Therefore, we define that π is *satisfied* in an interpretation I, denoted by $I \models \pi$, iff $\pi_{\mathbf{L}} \subseteq I \cap \pi_{\mathbf{D}} \subseteq \pi_{\mathbf{G}}$. This definition justifies the name of the new primitive: the projection of I with respect to $\pi_{\mathbf{D}}$ can be viewed as a *sample* of the interpretation I. In order to satisfy π, the sample must be in the *range* determined by $\pi_{\mathbf{L}}$ and $\pi_{\mathbf{G}}$, i.e., a superset of $\pi_{\mathbf{L}}$ and a subset of $\pi_{\mathbf{G}}$. The set of *satisfiers* of a sampler π, denoted $\pi_{\mathbf{S}}$, is $\{S \subseteq \pi_{\mathbf{D}} \mid \pi_{\mathbf{L}} \subseteq S \subseteq \pi_{\mathbf{G}}\}$. A sampler π is called *exact*, if $\pi_{\mathbf{L}} = \pi_{\mathbf{G}}$, and then abbreviated as a pair $\langle D, S\rangle$ where $S = L = G$. Thus positive and negative literals based on an atom a are captured by *primitive* exact samplers of the forms $\langle\{a\}, \{a\}\rangle$ and $\langle\{a\}, \emptyset\rangle$.

We assign a *disjunctive* interpretation to any set of samplers $\Pi = \{\pi_1, \ldots, \pi_k\}$, i.e., $I \models \Pi$ iff $I \models \pi_j$ for some $1 \leq j \leq k$. A *sampler program* (SP) P is a set of *sampling rules* of the form $\Pi \leftarrow \Pi_1, \ldots, \Pi_n$ where Π and each Π_i with $1 \leq i \leq n$ is such a set. In this notation, a singleton $\{\pi\}$ can be abbreviated by π whereas primitive exact samplers $\langle\{a\}, \{a\}\rangle$ and $\langle\{a\}, \emptyset\rangle$ are abbreviated by a and $\sim a$, respectively. The set of *head samplers* that appear in some rule head Π of P is denoted by $\mathrm{HeadS}(P)$. Likewise, we define the *set* $\mathrm{BodySS}(P)$ of sampler sets Π that occur in the rule bodies of P. A sampling rule $\Pi \leftarrow \Pi_1, \ldots, \Pi_n$ is satisfied in an interpretation I iff $I \models \Pi_1, \ldots, I \models \Pi_n$ imply $I \models \Pi$. Intuitively speaking, the body conditions $\Pi_1, \ldots, \Pi_n$ correspond to sets of samples taken of I. If at least one sample in each set Π_i produces the expected outcome, the same must hold for the head Π.

Example 3.2. Consider an infinite SP P having rules $\{p_0, q_0\} \leftarrow$ and $\{p_{i+1}, q_{i+1}\} \leftarrow \{p_i, q_i\}$ for all $i \geq 0$. Here each atom a denotes a primitive exact sampler $\langle\{a\}, \{a\}\rangle$. The latter rules correspond to choice rules $p_{i+1} | q_{i+1} \leftarrow p_i$ and $p_{i+1} | q_{i+1} \leftarrow q_i$ for each $i \geq 0$. Thus the "alternating" interpretation $M = \{p_0, q_1, p_2, q_3, \ldots\}$ is a model of P among others.

Definition 3.3 (Reduct)**.** For an SP P and an interpretation $M \subseteq \mathrm{At}(P)$, the reduct of

(1) a sampler $\pi = \langle D, L, G\rangle$, denoted π^M, is the sampler $\langle G, L, G\rangle$, if $M \models \pi$,
(2) a set Π of samplers, denoted Π^M, is the sampler set $\{\pi^M \mid \pi \in \Pi \text{ and } M \models \pi\}$, and
(3) a sampler program P, denoted by P^M, contains for all $\Pi \leftarrow \Pi_1, \ldots, \Pi_n \in P$ such that $M \models \Pi_1, \ldots, M \models \Pi_n$, and for all $\pi \in \Pi$ such that $M \models \pi$, a reduced sampling rule $\langle S, S\rangle \leftarrow \Pi_1^M, \ldots, \Pi_n^M$ where the exact satisfier $S = M \cap \pi_{\mathbf{G}}$ belongs to $\pi_{\mathbf{S}}$.

The goal of the definition of π^M is to partially evaluate negative default literals in $L = \{\sim a \mid a \in D \setminus G\}$ with respect to $M \models \pi$. For the same reason, we also have $\pi_{\mathbf{L}} \subseteq S \subseteq \pi_{\mathbf{G}}$ for the satisfier S in the last item. Thus $M \models P$ implies $M \models P^M$. The rules of a reduced SP P^M are all *positive* in the following sense: their heads comprise of *single* sampling atoms π satisfying $\pi_{\mathbf{D}} = \pi_{\mathbf{L}} = \pi_{\mathbf{G}}$ and their bodies involve only sampling atoms π with $\pi_{\mathbf{D}} = \pi_{\mathbf{G}}$. Positive SPs share a number of properties with their counterparts amongst normal programs.

Proposition 3.4 (Properties of Positive SPs)**.** *Let P and Q be two positive SPs.*

(1) *If $M_1 \models P$ and $M_2 \models P$ are two models of P, then also $M_1 \cap M_2 \models P$.*
(2) *The program P has a unique $\subseteq$-minimal model, i.e., the* least model $\mathrm{LM}(P)$ *of P which coincides with $\bigcap\{M \subseteq \mathrm{At}(P) \mid M \models P\}$.*
(3) *The least model $\mathrm{LM}(P)$ is the least fixed point $\mathrm{lfp}(\mathbf{T}_P)$ of the immediate consequence operator $\mathbf{T}_P$ defined for any $I \subseteq \mathrm{At}(P)$ by $\mathbf{T}_P(I) =$*

$$\bigcup\{S \mid \pi \leftarrow \Pi_1, \ldots, \Pi_n \in P, \pi_{\mathbf{S}} = \{S\}, \text{ and } I \models \Pi_1, \ldots, I \models \Pi_n\}.$$

Example 3.5. Consider a positive SP P with one sampling rule $\langle\{a\}, \{a\}\rangle \leftarrow \langle\{a\}, \emptyset, \{a\}\rangle$. The interpretation $M_1 = \emptyset$ is not a model of P but $M_2 = \{a\}$ is the least one.

We conclude that SPs provide a reasonable generalization of normal programs and CLPs. Accordingly, the definition of stable models (Definition 2.2) is applicable to SPs as such.

Example 3.6. For the sampler program P and interpretation M from Example 3.2, the reduct P^M is the positive SP $\{p_0 \leftarrow;\ \ q_1 \leftarrow p_0;\ \ p_2 \leftarrow q_1;\ \ q_3 \leftarrow p_2;\ \ \ldots\}$. Thus M is stable as $M \models P$ and $\mathrm{LM}(P^M) = M$. In Example 3.5, $M_2 = \{a\}$ is uniquely stable as $P^{M_2} = P$.

4. Relationship of CLPs and SPs

Let us begin by explaining how choice programs can be viewed as a special case of sampler programs. In this respect, we can fully exploit the conciseness of samplers and abbreviations introduced so far. A choice rule r of the form (2.1) can be rewritten as

$$\mathrm{Tr}_{\mathrm{SP}}(r) \quad = \quad \{a_1, \ldots, a_l\} \leftarrow \langle\{b_1, \ldots, b_m\}, \{b_1, \ldots, b_m\}\rangle, \langle\{c_1, \ldots, c_n\}, \emptyset\rangle. \tag{4.1}$$

In particular, the head of (4.1) is a shorthand for $\{\langle\{a_1\}, \{a_1\}\rangle, \ldots, \langle\{a_l\}, \{a_l\}\rangle\}$ by the notational conventions introduced above—not to be confused with the head of an SMODELS choice rule. The correctness of the program level transformation $\mathrm{Tr}_{\mathrm{SP}}(P) = \bigcup_{r \in P} \mathrm{Tr}_{\mathrm{SP}}(r)$ is formulated below. We omit the proof of this and subsequent theorems for space reasons.

Theorem 4.1 (Correctness of $\mathrm{Tr}_{\mathrm{SP}}$)**.** *For any CLP P,* $\mathrm{SM}(P) = \mathrm{SM}(\mathrm{Tr}_{\mathrm{SP}}(P))$.

Transforming SPs into CLPs is of equal interest. Due to the disjunctive interpretation of sets of sampling atoms, a set of choice rules is required to represent a sampling rule $r = \Pi \leftarrow \Pi_1, \ldots, \Pi_n$ in general. The length of the resulting CLP, denoted by $\mathrm{Tr}_{\mathrm{CLP}}(r)$ in the sequel, can be kept polynomial with respect to $\|r\|$ using new atoms.

Definition 4.2. A sampling rule $\Pi \leftarrow \Pi_1, \ldots, \Pi_n$ with $\Pi = \{\pi_1, \ldots, \pi_l\}$ is translated into

(1) a normal rule $s_i \leftarrow \pi_{\mathbf{L}}, {\sim}(\pi_{\mathbf{D}} \setminus \pi_{\mathbf{G}})$ for each $\pi \in \Pi_i$;
(2) a choice rule $h_1 | \ldots | h_l \leftarrow s_1, \ldots, s_n$;
(3) for each $\pi_i \in \Pi$, an integrity constraint $\leftarrow (\pi_i)_{\mathbf{L}}, s_1, \ldots, s_n, {\sim}h_i, {\sim}((\pi_i)_{\mathbf{D}} \setminus (\pi_i)_{\mathbf{G}})$;
(4) a normal rule $a \leftarrow h_i$ for each $\pi_i \in \Pi$ and $a \in (\pi_i)_{\mathbf{L}}$;
(5) a choice rule $c_1 \mid \ldots \mid c_k \mid e \leftarrow h_i$ with $\{c_1, \ldots, c_k\} = (\pi_i)_{\mathbf{G}} \setminus (\pi_i)_{\mathbf{L}}$ for each $\pi_i \in \Pi$;
(6) and an integrity constraint $\leftarrow h_i, b$ for each $\pi_i \in \Pi$ and $b \in (\pi_i)_{\mathbf{D}} \setminus (\pi_i)_{\mathbf{G}}$.

In the above, $h_1, \ldots, h_l$ and $s_1, \ldots, s_n$ are new atoms corresponding to samplers $\pi_1, \ldots, \pi_l$ in the head Π and the sets $\Pi_1, \ldots, \Pi_n$ in the body, respectively. The atom e in (5) is new.

The rules of Item 1 evaluate sampler sets $\Pi_1, \ldots, \Pi_n$ in the body. The rule of Item 2 is a skeleton of the original sampling rule. The application of head samplers is enforced by the integrity constraints of Item 3 (cf. Definition 3.3). The rules in Items 4–6 enforce the satisfaction of a single head sampler $\pi_i \in \Pi$ once applied. The translation of an entire SP P is $\mathrm{Tr}_{\mathrm{CLP}}(P) = (\bigcup_{r \in P} \mathrm{Tr}_{\mathrm{CLP}}(r)) \cup \{e \leftarrow\}$. To formulate the correctness of $\mathrm{Tr}_{\mathrm{CLP}}$, we need to map any interpretation $M \subseteq \mathrm{At}(P)$ to an interpretation $\mathrm{Ext}_P(M) \subseteq \mathrm{At}(\mathrm{Tr}_{\mathrm{CLP}}(P))$ which includes (i) M as such, (ii) the atom s associated with $\Pi \in \mathrm{BodySS}(P)$ iff $M \models \Pi$, (iii) the atom h associated with $\pi \in \mathrm{HeadS}(P)$ iff $M \models \pi$ and $M \models \Pi_1, \ldots, M \models \Pi_n$, and (iv) e.

Theorem 4.3 (Faithfulness of $\mathrm{Tr}_{\mathrm{CLP}}$)**.** *Let P be any SP and $\mathrm{Tr}_{\mathrm{CLP}}(P)$ its translation into a CLP. (i) If $M \in \mathrm{SM}(P)$, then $N = \mathrm{Ext}_P(M) \in \mathrm{SM}(\mathrm{Tr}_{\mathrm{CLP}}(P))$. (ii) If $N \in \mathrm{SM}(\mathrm{Tr}_{\mathrm{CLP}}(P))$, then its projection $M = N \cap \mathrm{At}(P)$ belongs to $\mathrm{SM}(P)$ and $N = \mathrm{Ext}_P(M)$.*

Due to new atoms, any SP P and $\mathrm{Tr}_{\mathrm{CLP}}(P)$ are *visibly equivalent* [Jan06] but not *strongly equivalent* [Lif01]. To conclude, SPs may provide more compact representations than CLPs.

5. Abstract Constraint Programs (ACPs)

The objective of this section is to show how SPs can be exploited to define the semantics of *abstract constraint programs* in the general case [Bla08]. Our strategy is to extend the stable model semantics by decomposing *abstract constraint atoms* into sets of samplers.

Definition 5.1. An abstract constraint atom π, or an *ac-atom* for short, has the form $\langle D, \{S_1, \ldots, S_k\}\rangle$ where the domain $\pi_{\mathbf{D}} = D$ is a finite set propositional atoms and each set $S_j \subseteq D$ where $1 \leq j \leq k$ is a satisfier in the set $\pi_{\mathbf{S}} = \{S_1, \ldots, S_k\}$ of satisfiers.

The idea is that an interpretation M satisfies an abstract constraint atom π iff the projection $M \cap \pi_{\mathbf{D}} \in \pi_{\mathbf{S}}$. An abstract constraint program (ACP) consists of rules of the form $\pi \leftarrow \pi_1, \ldots, \pi_n$ where π and each π_i is an abstract constraint atom. Certain subclasses have been identified: An ac-atom is *monotone* [Mar08] iff $S_1 \in \pi_{\mathbf{S}}$ and $S_1 \subseteq S_2 \subseteq \pi_{\mathbf{D}}$ imply $S_2 \in \pi_{\mathbf{S}}$. Furthermore, an ac-atom is *convex* [Liu06] iff $S_1 \in \pi_{\mathbf{S}}$, $S_1 \subseteq S_2 \subseteq S_3$, and $S_3 \in \pi_{\mathbf{S}}$ imply $S_2 \in \pi_{\mathbf{S}}$. The rules of *monotone* ACPs and *convex* ACPs solely consist of monotone and convex ac-atoms, respectively. It is clear that monotone ACPs specialize convex ones.

We are now ready to address the semantics of ACPs from the perspective of SPs. Consider two samplers π and π' such that $\pi_{\mathbf{D}} = (\pi')_{\mathbf{D}}$. We say that π *extends* π' iff $(\pi')_{\mathbf{S}} \subseteq \pi_{\mathbf{S}}$, i.e., $\pi_{\mathbf{L}} \subseteq (\pi')_{\mathbf{L}}$ and $(\pi')_{\mathbf{G}} \subseteq \pi_{\mathbf{G}}$. Intuitively speaking, the range of π is greater than or equal to that of π', denoted $\pi' \leq \pi$. Samplers which are $\leq$-maximal provide a basis for the decomposition of ac-atoms and they also guarantee the uniqueness of decompositions.

Definition 5.2 (Decomposition)**.** An ac-atom $\pi = \langle D, \{S_1, \ldots, S_k\}\rangle$ is decomposed into a set of samplers $\mathrm{DS}(\pi) = \{\pi_1, \ldots, \pi_m\}$ such that $(\pi_j)_{\mathbf{D}} = D$, $(\pi_j)_{\mathbf{L}} \in \pi_{\mathbf{S}}$, and $(\pi_j)_{\mathbf{G}} \in \pi_{\mathbf{S}}$ for each $1 \leq j \leq m$, $\bigcup_{i=1}^{m} (\pi_j)_{\mathbf{S}} = \pi_{\mathbf{S}}$, and each $\pi_j \in \mathrm{DS}(\pi)$ is $\leq$-maximal within $\mathrm{DS}(\pi)$.

We observe that $1 \leq m \leq k$ holds for the cardinality m of $\mathrm{DS}(\pi)$. If $m = 1$, then $k = 2^{|G \setminus L|}$, which shows that $\mathrm{DS}(\pi)$ can provide exponentially more succinct representation of π. If $m = k$, then each S_i, $1 \leq i \leq k$, corresponds to an exact sampler $\langle D, S_i\rangle$ of its own. The decomposition of ac-atoms preserves satisfaction under classical semantics, i.e., $I \models \pi$ iff $I \models \mathrm{DS}(\pi)$ holds for any ac-atom π and any interpretation I of π. If an ac-atom π is monotone, then $\mathrm{DS}(\pi)$ includes a sampler $\pi' = \langle D, S_i, D\rangle$ for the domain $D = \pi_{\mathbf{D}}$ and each $\subseteq$-minimal satisfier $S_i \in \pi_{\mathbf{S}}$. Thus we obtain $\mathrm{DS}(\langle\{a\}, \{\emptyset, \{a\}\}\rangle) = \{\langle\{a\}, \emptyset, \{a\}\rangle\}$. On the other hand, if π is convex, then $\mathrm{DS}(\pi)$ contains a sampler $\pi' = \langle D, S_i, S_j\rangle$ for each pair of a $\subseteq$-minimal satisfier $S_i \in \pi_{\mathbf{S}}$ and a $\subseteq$-maximal satisfier $S_j \in \pi_{\mathbf{S}}$ such that $S_i \subseteq S_j$. Note that for monotone ac-atoms π, the domain $D = \pi_{\mathbf{D}}$ is the unique $\subseteq$-maximal element in $\pi_{\mathbf{S}}$.

As regards a rule $\pi_0 \leftarrow \pi_1, \ldots, \pi_n$ involving ac-atoms, it can be modularly decomposed into a sampling rule $\mathrm{DS}(\pi_0) \leftarrow \mathrm{DS}(\pi_1), \ldots, \mathrm{DS}(\pi_n)$. The respective decomposition of an entire ACP P is denoted by $\mathrm{DS}(P)$. Stable models generalize for ACPs via Definition 2.2.

Definition 5.3 (Stable Models of ACPs)**.** Given an ACP P, an interpretation $M \subseteq \mathrm{At}(P)$ of P is a stable model of P iff $M \models P$ and $M = \mathrm{LM}(\mathrm{DS}(P)^M)$.

The reduct $\mathrm{DS}(P)^M$ is a positive SP for which the least model is well-defined according to Proposition 3.4. In addition, the condition $M \models P$ is equivalent to $M \models \mathrm{DS}(P)$ as classical models are preserved by decomposition. Hence we have $\mathrm{SM}(P) = \mathrm{SM}(\mathrm{DS}(P))$ for any ACP P in general. It is also possible to combine Definitions 2.1 and 3.3 in order to generalize the Gelfond-Lifschitz reduct for ACPs. For an entire ACP P, we can define P^M as $\mathrm{DS}(P)^M$ so that Definition 2.2 becomes directly applicable to ACPs. For an individual rule $\pi_0 \leftarrow \pi_1, \ldots, \pi_n \in P$ such that $M \models \pi_1, \ldots, M \models \pi_n$ and $M \models \pi_0$ under the assumption

that $M \models P$, the reduct $\mathrm{DS}(P)^M$ contains a reduced rule $\langle S, S\rangle \leftarrow \mathrm{DS}(\pi_1)^M, \ldots, \mathrm{DS}(\pi_n)^M$ with $S = M \cap \pi_{\mathbf{G}}$ for every $\leq$-maximal sampler $\pi \in \mathrm{DS}(\pi_0)$ such that $M \models \pi$. When the reduction takes place, an ac-atom π_i is mapped into $\mathrm{DS}(\pi_i)^M$ which is not generally representable as an ac-atom due to fixed domains. A convex ACP is illustrated below.

Example 5.4. Consider an ACP P with the following rules:

$$\langle\{a\},\{\emptyset,\{a\}\}\rangle \leftarrow;\ \langle\{b\},\{\emptyset,\{b\}\}\rangle \leftarrow;\ \langle\{c\},\{\emptyset,\{c\}\}\rangle \leftarrow;$$
$$\langle\emptyset,\emptyset\rangle \leftarrow \langle\{a,b,c\},\{\{a\},\{b\},\{c\},\{a,b\},\{a,c\},\{b,c\}\}\rangle.$$

The first monotone rule expresses the free choice of a and it decomposes into $\langle\{a\},\emptyset,\{a\}\rangle \leftarrow$. The rules for b and c are analogous. The last rule captures a *cardinality constraint* [Sim99] $\leftarrow 1\{a,b,c\}2$ with a convex ac-atom. If decomposed, 6 samplers $\langle\{a,b,c\},L,G\rangle$ where $L \subseteq G$, $L \in \{\{a\},\{b\},\{c\}\}$ and $G \in \{\{a,b\},\{a,c\},\{b,c\}\}$ result. The models of P are $M_1 = \emptyset$ and $M_2 = \{a,b,c\}$. For M_1, we obtain only $\langle\emptyset,\emptyset\rangle \leftarrow$ to P^{M_1} so that $M_1 \in \mathrm{SM}(P)$. The reduct P^{M_2} contains rules $\langle\{a\},\{a\}\rangle \leftarrow$, $\langle\{b\},\{b\}\rangle \leftarrow$, and $\langle\{c\},\{c\}\rangle \leftarrow$. Thus $M_2 \in \mathrm{SM}(P)$.

6. Characterization Based on Computations

The stable models of ACPs have been characterized in terms of *abstract computations*, e.g., in the monotone case [Mar08, Liu06]. In what follows, we review the definition of computations for arbitrary ACPs [Liu10] but using ordinals as indices. Given an interpretation $I \subseteq \mathrm{At}(P)$ of an ACP P, the set $P(I)$ of *supporting rules* of P is $\{\pi \leftarrow \pi_1, \ldots, \pi_n \in P \mid I \models \pi_1, \ldots, I \models \pi_n\}$. Moreover, the set $\mathrm{HAt}(P)$ of *head atoms* of P is $\bigcup\{\pi_{\mathbf{D}} \mid \pi \leftarrow \pi_1, \ldots, \pi_n \in P\}$. Computations associated with P are *sequences of interpretations* $\langle I_\alpha\rangle$ indexed by ordinals α. Their properties are formalized using a *nondeterministic* immediate consequence operator $\mathbf{T}_P^{\mathrm{nd}}$ that assigns to any interpretation $I \subseteq \mathrm{At}(P)$ a *set* of interpretations $J \subseteq \mathrm{HAt}(P(I))$ such that $J \models \mathrm{Heads}(P(I))$ where $\mathrm{Heads}(P(I))$ is the set of *heads* of the rules in $P(I)$. *Persistent computations* $\langle I_\alpha\rangle$ meet the following criteria [Liu10]:

- (**R**) *Revision:* For every ordinal α, the interpretation $I_{\alpha+1}$ is grounded in I_α and P, i.e., $I_{\alpha+1} \in \mathbf{T}_Q^{\mathrm{nd}}(I_\alpha)$ for some program $Q \subseteq P(I_\alpha)$.
- (**P**) *Persistence of beliefs:* The sequence $\langle I_\alpha\rangle$ starts from $I_0 = \emptyset$ and it is monotonically increasing, i.e., $I_\alpha \subseteq I_{\alpha+1}$ for all ordinals α, and $I_\beta = \bigcup_{\alpha<\beta} I_\alpha$ for *limit* ordinals β.
- (**C**) *Convergence:* The limit I_∞ that defines the result of the computation $\langle I_\alpha\rangle$ is a *supported model* of P, i.e., it satisfies the fixed-point condition $I_\infty \in \mathbf{T}_P^{\mathrm{nd}}(I_\infty)$.
- (**Pr**) *Persistence of reasons:* There is a sequence $\langle P_\alpha\rangle$ of programs such that for all ordinals α, the program $P_\alpha \subseteq P(I_\alpha)$, $P_\alpha \subseteq P_{\alpha+1}$, and $I_{\alpha+1} \in \mathbf{T}_{P_\alpha}^{\mathrm{nd}}(I_\alpha)$.

Item (**P**) and Knaster-Tarski lemma guarantee that the limit $I_\infty = \bigcup_\alpha I_\alpha$ exists. It is defined as a stable model of P in [Liu10]. Definition 5.3 leads to another class of computations.

Definition 6.1 (Canonical Computations for ACPs)**.** Given an ACP P and an interpretation $M \subseteq \mathrm{At}(P)$, the canonical M-computation for P is a sequence $\langle I_\alpha\rangle$ such that (i) $I_0 = \emptyset$, (ii) $I_{\alpha+1} = \mathbf{T}_{\mathrm{DS}(P)^M}(I_\alpha)$ for each ordinal α, and (iii) $I_\beta = \bigcup_{\alpha<\beta} I_\alpha$ for limit ordinals β.

The operator $\mathbf{T}_{\mathrm{DS}(P)^M}$ is monotone and compact since the rules of $\mathrm{DS}(P)^M$ have the form $\langle S, S\rangle \leftarrow \mathrm{DS}(\pi_1)^M, \ldots, \mathrm{DS}(\pi_n)^M$ and the samplers involved have finite domains. Thus, given a canonical M-computation $\langle I_\alpha\rangle$ for an ACP P and $M \subseteq \mathrm{At}(P)$, we know that (i) $\langle I_\alpha\rangle$ is monotonically increasing, (ii) the limit $I_\omega = \mathrm{lfp}(\mathbf{T}_{\mathrm{DS}(P)^M})$, and (iii) $I_\omega = \mathbf{T}_{\mathrm{DS}(P)^M}(I_\omega)$.

Corollary 6.2 (Characterization). *For an ACP P, an interpretation $M \subseteq \mathrm{At}(P)$ is a stable model of P iff $M \models P$ and $M = I_\omega$ for the result I_ω of the canonical M-computation $\langle I_\alpha \rangle$.*

Theorem 6.3 (Properties of Canonical Computations). *Let P be an ACP and $M \subseteq \mathrm{At}(P)$ a model of P such that $I_\omega = M$ for the limit I_ω of the canonical M-computation $\langle I_\alpha \rangle$. Then $\langle I_\alpha \rangle$ satisfies (**R**), (**P**), (**C**), and (**Pr**) when P_α and Q in (**Pr**) and (**R**), respectively, are substituted by $P_\alpha(M) = \{\pi \leftarrow \pi_1, \ldots, \pi_n \in P(M) \mid I_\alpha \models \mathrm{DS}(\pi_1)^M, \ldots, I_\alpha \models \mathrm{DS}(\pi_n)^M\}$.*

The characterization above is limited to the "successful cases", i.e., when the result turns out to be a stable model. The properties of canonical M-computations are not semantically important when $M \not\models P$ or $I_\omega \neq M$. In both cases, the interpretation M is disqualified as a stable model. It is nevertheless clear that (**P**) holds even for failing computations. Finally, we note that the semantics based on Definition 5.3 can be stricter than the one based on abstract computations. As shown in [She09a], there is a persistent computation and a stable model $M = \{p(-1), p(1), p(2)\}$ for an ACP with $p(1) \leftarrow$; $p(-1) \leftarrow p(2)$; and $p(2) \leftarrow \pi$ where the ac-atom π corresponds to a sum aggregate $\mathrm{Sum}(\{X \mid p(X)\}) \geq 1$ based on the domain $\pi_{\mathbf{D}} = D = \{p(-1), p(1), p(2)\}$. In our approach, the reduct P^M consists of $p(1) \leftarrow$, $p(-1) \leftarrow p(2)$, and $p(2) \leftarrow \langle D, \{p(2)\}, D\rangle$ which indicate the instability of M.

7. Comparison with Previous Approaches

This research was initially motivated by the notions of computations proposed for ACPs. In view of the results presented in [Liu10], we have lifted the notion of M-computations, originally proposed for normal programs, to the case of ACPs. One of our key design decisions was to push all nondeterminism involved in abstract computations to the notion of a program reduct—much in the spirit of choice logic programs [Soi99] covered by Definition 2.1. Corollary 6.2 indicates that each stable model M of an ACP P is generated by a unique computation satisfying the criteria of [Liu10] by Theorem 6.3. As noted above, those criteria lead to a weaker notion of stability if directly generalized for ACPs. Stability notions based on additional criteria [Liu10] depart from the traditional fixed-point definition [Gel88].

The alternative interpretation of ac-atoms as sampler sets led us to an approach which is closely related to the one presented in [She07]. In this work, the counterpart of a sampler $\pi = \langle D, L, G\rangle$ is an *abstract L-prefixed power set* of the form $L \uplus (G \setminus L)$, i.e., the set of sets $\{L \cup K \mid K \subseteq G \setminus L\}$ which coincides with $\pi_{\mathbf{S}}$. These structures are merely used as a compact representation of ac-atoms rather than new primitives for logic programs. Moreover, the generalization of the Gelfond-Lifschitz reduct for an ACP P takes place at a lower level of abstraction: the Shen-You reduct P_M [She07] is formulated as a positive normal logic program and new atoms become a necessity. The way in which ac-atoms are decomposed as sets of samplers (cf. Definition 5.2) pave the way for a tight interconnection.

Theorem 7.1. *For an ACP P and an interpretation $M \subseteq \mathrm{At}(P)$, $M \in \mathrm{SM}(P)$ iff there is a unique minimal model N of the Shen-You reduct P_M such that $M = N \cap \mathrm{At}(P)$.*

This result covers also rules of the form $\pi \leftarrow \pi_1, \ldots, \pi_n$ which have arbitrary ac-atoms in their heads. Further interconnections can be reported from [She09a] for ACPs confining to a limited syntax, i.e., rules of the form $a \leftarrow \pi_1, \ldots, \pi_n$. Given this restriction, stable models of [Den01, Son07b] coincide with models obtained as minimal models of P_M. By Theorem 7.1 the same observation can be made for stable models conforming to Definition 5.3. The iterative construction of stable models in [Son07b] is analogous to canonical M-computations

introduced by us. A difference is that using SPs, a fixed reduct $P^M = \mathrm{DS}(P)^M$ of an ACP P can be formalized and there is no need to parameterize the construction of M otherwise. In this respect, the approaches in [Son07b, Son07a] resort to *conditional satisfaction*. There are further consequences of the relationships pointed above. First of all, the semantics of monotone ACPs is captured in the standard way [Mar08] in our approach. The case of convex ACPs [Liu06] is also covered as illustrated by Example 5.4. We also observe that cardinality and weight rules of the SMODELS system [Sim99] essentially lead to convex constraints. Finally, the notions based on minimal models [Del03, Fab04, Fer05] are prone to self-supporting stable models as shown in [She09a]. We avoid such models by Theorem 7.1.

As the last comparison, we address the program transformation $trm(\cdot)$ from [Pel03]. The idea is to map sets of classical literals $F^D_{\langle L,G\rangle} = \{b \mid b \in L\} \cup \{\neg c \mid c \in D \setminus G\}$ which satisfy an ac-atom $\pi = \langle D, \{S_1, \ldots, S_k\}\rangle$ and are $\subseteq$-minimal in this respect. Given a rule $a \leftarrow \pi$, each set $F^D_{\langle L,G\rangle}$ gives rise to one normal rule $a \leftarrow L, {\sim}(D \setminus G)$ in the translation. This is analogous to translating $\leq$-maximal samplers $\pi' = \langle D, L, G\rangle \in \mathrm{DS}(\pi)$ using $\mathrm{Tr}_{\mathrm{CLP}}(\cdot)$ from Definition 4.2. However, since new atoms are not introduced, the translation $trm(\cdot)$ may create an exponential number of normal rules already for rules of the form $a \leftarrow \pi_1, \ldots, \pi_n$ with several ac-atoms in the rule body. A further aspect is that rules $\pi \leftarrow \pi_1, \ldots, \pi_n$ involving proper ac-atoms π in their heads are not covered at all.

8. Conclusions

In this paper, we propose samplers as new building blocks of logic programs. The respective class of sampler programs (SPs) is designed using a conceptually simpler class of choice logic programs (CLPs) as a starting point. Based on the intuitions provided by CLPs, the stable model semantics (Definition 2.2) extends for SPs in a natural way. As witnessed by Definition 3.3, the notion of a program reduct [Gel88, Soi99] carries over for SPs so that, in particular, the resulting program is a (positive) SP. The availability of *polynomial*, *faithful*, and *modular* (PFM) translation functions $\mathrm{Tr}_{\mathrm{SP}}$ and $\mathrm{Tr}_{\mathrm{CLP}}$ suggests that CLPs and SPs have the same expressive power in the sense of [Jan06] but SPs may provide more concise representation due to samplers and the disjunctive interpretation of sampler sets.

The second main theme of this paper is the application of SPs in order to define the semantics of abstract constraint programs (ACPs). Notably, this class of programs lacks a natural counterpart of Gelfond-Lifschitz reduction [Gel88] which would yield ACPs as its outcome. In pursuit of a simple semantical definition, we propose an approach in which ACPs are interpreted as SPs using the decomposition of ac-atoms as sampler sets as basis. The decomposition method $\mathrm{DS}(\cdot)$ is highly modular since each ac-atom π can be locally decomposed into $\mathrm{DS}(\pi)$ independently of other ac-atoms in the program. The combination of Definitions 2.2, 3.3, and 5.2 enables the definition of stable models for ACPs as given in Definition 5.3: $M \in \mathrm{SM}(P)$ iff $M \models P$ and $M = \mathrm{LM}(\mathrm{DS}(P)^M)$. The new logic programming primitives proposed in this paper, samplers, play a key role in this streamlined definition. The definition is stated without a reference to other logics such as propositional or default logics in contrast with [She07, She09a]. Nevertheless, the semantics of ACPs originally proposed in [She07] is supported by our results (Theorem 7.1). In view of computations, we established in Corollary 6.2 that each $M \in \mathrm{SM}(P)$ has a unique *deterministic* computation, i.e., the canonical M-computation, associated with it. By these observations, we conclude that SPs form an interesting class of logic programs between CLPs and ACPs.

Acknowledgement

The author would like to thank Martin Gebser for his comments on a draft of this paper.

References

[Bla08] H. Blair, V. Marek, and J. Remmel. Set based logic programming. *Annals of Mathematics and Artificial Intelligence*, 52(1):81–105, 2008.

[Del03] T. Dell'Armi, W. Faber, G. Ielpa, N. Leone, and G. Pfeifer. Aggregate functions in disjunctive logic programming: Semantics, complexity, and implementation in dlv. In *Proc. IJCAI-03*, pp. 847–852. Morgan Kaufmann, 2003.

[Den01] M. Denecker, N. Pelov, and M. Bruynooghe. Ultimate well-founded and stable semantics for logic programs with aggregates. In *Proc. ICLP'01*, *LNCS*, vol. 2237, pp. 212–226. Springer, 2001.

[Fab04] W. Faber, N. Leone, and G. Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *Proc. JELIA'04*, *LNCS*, vol. 3229, pp. 200–212. Springer, 2004.

[Fer05] P. Ferraris. Answer sets for propositional theories. In *Proc. LPNMR'05*, *LNCS*, vol. 3662, pp. 119–131. Springer, 2005.

[Gel88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. ICLP'88*, pp. 1070–1080. 1988.

[Gel90] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *Proc. ICLP'90*, pp. 579–597. The MIT Press, 1990.

[Gel91] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–385, 1991.

[Jan06] T. Janhunen. Some (in)translatability results for normal logic programs and propositional theories. *Journal of Applied Non-Classical Logics*, 16(1–2):35–86, 2006.

[Lif01] V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.

[Lif08] Vladimir Lifschitz. Twelve definitions of a stable model. In *Proc. ICLP'08*, *LNCS*, vol. 5366, pp. 37–51. Springer, 2008.

[Liu06] L. Liu and M. Truszczynski. Properties and applications of programs with monotone and convex constraints. *Journal of Artificial Intelliegence Research*, 27:299–334, 2006.

[Liu10] L. Liu, E. Pontelli, T. C. Son, and M. Truszczynski. Logic programs with abstract constraint atoms: The role of computations. *Artificial Intelligence*, 174:295–315, 2010.

[Mar08] V. Marek, I. Niemelä, and M. Truszczynski. Logic programs with monotone abstract constraint atoms. *Theory and Practice of Logic Programming*, 8(2):167–199, 2008.

[Pel03] N. Pelov, M. Denecker, and M. Bruynooghe. Translation of aggregate programs to normal logic programs. In *Proc. ASP'03*, *CEUR Workshop Proceedings*, vol. 78. 2003.

[She07] Y.-D. Shen and J.-H. You. A generalized Gelfond-Lifschitz transformation for logic programs with abstract constraints. In *Proc. AAAI'07*, pp. 483–488. AAAI Press, 2007.

[She09a] Y.-D. Shen and J.-H. You. A default approach to semantics of logic programs with constraint atoms. In *Proc. LPNMR'09*, *LNCS*, vol. 5753, pp. 277–289. Springer, 2009.

[She09b] Y.-D. Shen, J.-H. You, and L.-Y. Yuan. Characterizations of stable model semantics for logic programs with arbitrary constraint atoms. *Theory and Practice of Logic Programming*, 9(4):529–564, 2009.

[Sim99] P. Simons. Extending the stable model semantics with more expressive rules. In *Proc. LPNMR'99*, *LNCS*, vol. 1730, pp. 305–316. Springer, 1999.

[Sim02] P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1–2):181–234, 2002.

[Soi99] T. Soininen and I. Niemelä. Developing a declarative rule language for applications in product configuration. In *Proc. PADL'99*, *LNCS*, vol. 1551, pp. 305–319. Springer, 1999.

[Son07a] T. C. Son and E. Pontelli. A constructive semantic characterization of aggregates in answer set programming. *Theory and Practice of Logic Programming*, 7(3):355–375, 2007.

[Son07b] T. C. Son, E. Pontelli, and P. H. Tu. Answer sets for logic programs with arbitrary abstract constraint atoms. *Journal of Artificial Intelligence Research*, 29:353–389, 2007.