TPLP **20** (6): 895–910, 2020.
doi:10.1017/S1471068420000393

ASP($\mathscr{AC}$): Answer Set Programming with Algebraic Constraints

THOMAS EITER and RAFAEL KIESEL

Technical University Vienna, Vienna, Austria
(*e-mail:* {thomas.eiter,rafael.kiesel}@tuwien.ac.at)

submitted 10 August 2020; revised 11 August 2020; accepted 12 August 2020

Abstract

Weighted Logic is a powerful tool for the specification of calculations over semirings that depend on qualitative information. Using a novel combination of Weighted Logic and Here-and-There (HT) Logic, in which this dependence is based on intuitionistic grounds, we introduce Answer Set Programming with Algebraic Constraints (ASP($\mathscr{AC}$)), where rules may contain constraints that compare semiring values to weighted formula evaluations. Such constraints provide streamlined access to a manifold of constructs available in ASP, like aggregates, choice constraints, and arithmetic operators. They extend some of them and provide a generic framework for defining programs with algebraic computation, which can be fruitfully used e.g. for provenance semantics of datalog programs. While undecidable in general, expressive fragments of ASP($\mathscr{AC}$) can be exploited for effective problem solving in a rich framework.

KEYWORDS: Weighted Logic, Here-and-There Logic, Answer Set Programming, Constraints

1 Introduction

Answer Set Programming (ASP) is a well-known non-monotonic declarative programming paradigm. Due to the need for more expressiveness and succinct descriptions, it has been extended with many different constructs, ranging from nested expressions (Lifschitz et al. 1999) to weight constraints with conditionals (Niemelä et al. 1999) and aggregates (Ferraris 2011). A more recent trend combines ASP with Constraint Processing (CP) employing both solvers for ASP and Satisfaction Modulo Theories (SMT) (Lierler 2014; Janhunen 2018). Many of these approaches keep nonmonotonicity on the ASP side, but also its use on the CP side was explored (Aziz et al. 2013). Cabalar et al. (2020a; 2020b) recently introduced a general non-monotonic integration of CP into ASP providing aggregates and conditionals for the specification of numeric values that depend on the satisfaction of formulas. The conditionals can be flexibly evaluated under the *vicious circle* (vc) or the *definedness* principles (df). Their approach treats constraints as black boxes, leaving their syntax open, and incorporates many previously introduced constructs.

An important feature of constraint extensions is the possibility to express (in)equations involving computations on an algebraic structure, whose solutions are accessible by the ASP rules. Basic such structures are *semirings* $\mathscr{R} = (R, \oplus, \otimes, e_{\oplus}, e_{\otimes})$, where $\oplus$ and $\otimes$ are addition and multiplication with neutral elements $e_{\oplus}$ and $e_{\otimes}$, respectively. They have been considered for constraint semantics (Bistarelli et al. 1997) and were used to provide rule languages with parameterised calculation in a uniform syntax but flexible semantics (Kimmig et al. 2011; Eiter and Kiesel 2020).

Notably, *Weighted Logic* (Droste and Gastin 2007) links semirings with predicate logic, where *weighted formulas* are interpreted as algebraic expressions over a semiring $\mathscr{R}$. Similar to the conditionals of Cabalar et al. (2020b), the value of a weighted formula α depends on the truth of the atoms in α. E.g. the weighted formula $1+\texttt{deadline}*(2+\texttt{pagelimit}*3)$ over the natural numbers $\mathbb{N}$ may represent how many cups of coffee a researcher drinks: if there is a `deadline` but no `pagelimit`, the result is $1+1\cdot(2+0\cdot 3)=3$.

In recent work (Eiter and Kiesel 2020), we exploited Weighted Logic in quantitative stream reasoning to assign weights to answer streams and aggregate over them. Here, we introduce a non-monotonic version of it in terms of *First-Order Weighted Here-and-There Logic (FO-WHT)*. The resulting logic complements Cabalar et al.'s work on abstract constraints in several respects:

- it offers an elegant way of specifying calculations, aggregates and non-monotonic conditionals natively and without the need for auxiliary definitions;
- the semantics provides a natural alternative to the vicious circle and definedness principles which arguably combines their strengths;
- the parameterisation with semirings allows for terms in a uniform syntax that are not bound to the reals but can be over any semiring (which may be fixed at runtime).

As customary for ASP we restrict ourselves to a fragment of FO-WHT Logic and introduce $\mathscr{AC}$-programs that allow for *algebraic constraints*, i.e. constraints on the values of weighted formulas, in both heads and bodies of rules. $\mathscr{AC}$-programs incorporate and extend many previous ASP constructs and thus provide a rich framework for declarative problem solving in a succinct form. The main contributions of this paper are briefly summarised as follows:

- We introduce First-Order Weighed HT Logic and $\mathscr{AC}$-programs that include constraints over weighted formulas (Section 3). By using a variant of HT Logic with non-disjoint sorts, such that variables can range over subsets shared by a domain and semirings, we enable the usage of constraints over different semirings within the same program.
- We consider different constructs in extensions of ASP like aggregates, choice constraints and conditionals, and we demonstrate how to model them in our framework. Further, we present the novel *minimised constraints* that allow for subset minimal guessing (Section 4).
- To demonstrate the power of ASP($\mathscr{AC}$) , we illustrate how provenance semantics for positive datalog programs can be elegantly encoded (Section 5).
- We consider different language aspects leading, firstly, to a broad class of *safe* $\mathscr{AC}$-programs, which we show to be domain independent; and, secondly, to a characterisation of strong equivalence for $\mathscr{AC}$-programs by equivalence in FO-WHT Logic (Section 6).
- We obtain that in the propositional (ground) case, the complexity of disjunctive logic programs is retained, i.e. model checking (MC) and strong equivalence are co-NP-complete while answer set existence (SAT) is Σ_2^p-complete, if the used semirings satisfy a practically mild encoding condition. For safe non-ground programs, MC is feasible in EXPTIME at most; SAT and SE are undecidable in general, but expressive decidable fragments are available (Section 7).

Proof details and more details can be found in the supplementary material corresponding to this paper at the TPLP archives.

2 Preliminaries

We start by introducing classical programs and their semantics. We use a variant of first-order HT semantics (Pearce and Valverde 2008) as this facilitates the generalisation of the semantics later on and is useful for work on strong equivalence. The variant is that we assign variables non-disjoint sorts, which lets us quantify over subsets of the domain. This slightly differs from other approaches in logic programming that use sorts, like that of Balai et al. (2013), where the arguments of predicates are sorted.

We consider sorted first-order formulas over a *signature* $\sigma = \langle \mathscr{D}, \mathscr{P}, \mathscr{X}, \mathscr{S}, r \rangle$, where $\mathscr{D}$ is a set of domain elements, $\mathscr{P}$ a set of predicates, $\mathscr{X}$ a set of sorted variables, $\mathscr{S}$ a set of sorts and $r : \mathscr{S} \to 2^{\mathscr{D}}$ a range function assigning each sort a subset of the domain. When $x \in \mathscr{X}$, we write $s(x)$ for the sort of x. Given a signature σ, we define the syntax of σ-*formulas* by

$$\phi ::= \bot \mid p(\vec{x}) \mid \phi \to \phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \exists y \phi \mid \forall y \phi, \tag{1}$$

where $p \in \mathscr{P}$, $\vec{x} = x_1, \ldots, x_n$, with $x_i \in \mathscr{D}$ or $x_i \in \mathscr{X}$ and $y \in \mathscr{X}$; $p(\vec{x})$ is called a σ-*atom*. We define $\neg\phi = \phi \to \bot$. A σ-*sentence* is a σ-formula without free variables.

Definition 1 (HT Semantics). *Let $\sigma = \langle \mathscr{D}, \mathscr{P}, \mathscr{X}, \mathscr{S}, r \rangle$ be a signature and $\mathscr{I}^H, \mathscr{I}^T$ be σ-*interpretations*, i.e. sets of σ-atoms without free variables over the predicates in $\mathscr{P}$ and elements in $\mathscr{D}$, s.t. $\mathscr{I}^H \subseteq \mathscr{I}^T$. Then $\mathscr{I} = (\mathscr{I}^H, \mathscr{I}^T)$ is a* σ-HT-interpretation *and $\mathscr{I}_w = (\mathscr{I}^H, \mathscr{I}^T, w)$, for $w \in \{H, T\}$, is a* pointed σ-HT-interpretation.

Satisfaction of a σ-sentence ϕ w.r.t. a pointed σ-HT-interpretation $\mathscr{I}_w = (\mathscr{I}^H, \mathscr{I}^T, w)$ is defined as follows, where we have the reflexive order $\geq$ on $\{H, T\}$, with $T \geq H$:

$$\begin{array}{lcl}
\mathscr{I}_w \not\models_\sigma \bot & & \\
\mathscr{I}_w \models_\sigma p(\vec{x}) & \iff & p(\vec{x}) \in \mathscr{I}^w \\
\mathscr{I}_w \models_\sigma \phi \to \psi & \iff & \mathscr{I}_{w'} \not\models_\sigma \phi \textit{ or } \mathscr{I}_{w'} \models_\sigma \psi \textit{ for all } w' \geq w \\
\mathscr{I}_w \models_\sigma \phi \vee \psi & \iff & \mathscr{I}_w \models_\sigma \phi \textit{ or } \mathscr{I}_w \models_\sigma \psi \\
\mathscr{I}_w \models_\sigma \phi \wedge \psi & \iff & \mathscr{I}_w \models_\sigma \phi \textit{ and } \mathscr{I}_w \models_\sigma \psi \\
\mathscr{I}_w \models_\sigma \exists x \phi(x) & \iff & \mathscr{I}_w \models_\sigma \phi(\xi), \textit{ for some } \xi \in r(s(x)) \\
\mathscr{I}_w \models_\sigma \forall x \phi(x) & \iff & \mathscr{I}_w \models_\sigma \phi(\xi), \textit{ for all } \xi \in r(s(x))
\end{array}$$

When T is a set of σ-sentences, then $\mathscr{I}_w \models_\sigma T$ if $\forall \phi \in T : \mathscr{I}_w \models_\sigma \phi$.

The semantics of classical rules and programs is introduced as an instantiation of the above semantics for restricted signatures. Let $\sigma = \langle \mathscr{D}, \mathscr{P}, \mathscr{X}, \mathscr{S}, r \rangle$ be a *classical signature*, i.e. $\mathscr{S} = \{\top\}, r(\top) = \mathscr{D}$. Then a *rule* is of the form

$$r = H(r) \leftarrow B(r) = \phi \leftarrow \psi_1, \ldots, \psi_n, \neg\theta_1, \ldots, \neg\theta_m,$$

where ϕ, ψ_i, θ_j are σ-atoms, with free variables $x_1, \ldots, x_k \in \mathscr{X}$. Its semantics is that of the σ-formula $\forall x_1, \ldots, x_k\, B_\wedge(r) \to \phi$ where $B_\wedge(r)$ is $\psi_1 \wedge \cdots \wedge \psi_n \wedge \neg\theta_1 \wedge \cdots \wedge \neg\theta_m$. Similarly, a *program* Π is a set of rules.

Definition 2 (Equilibrium Model). *Given a signature σ, a σ-interpretation $\mathscr{I}$ is an* equilibrium model *of a (set of) σ-sentence(s) ϕ if $(\mathscr{I}, \mathscr{I}, H) \models_\sigma \phi$ and for all $\mathscr{I}' \subsetneq \mathscr{I} : (\mathscr{I}', \mathscr{I}, H) \not\models_\sigma \phi$.*

To introduce weighted formulas, we first recall semirings.

Definition 3 (Semiring). *A* semiring *$\mathscr{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ is a set R equipped with two binary operations $\oplus$ and $\otimes$, which are called* addition *and* multiplication, *such that*

- *$(R, \oplus)$ is a commutative monoid with identity element $e_\oplus$,*
- *$(R, \otimes)$ is a monoid with identity element $e_\otimes$,*
- *multiplication left and right distributes over addition,*
- *and multiplication by $e_\oplus$ annihilates R, i.e. $\forall r \in R : r \otimes e_\oplus = e_\oplus = e_\oplus \otimes r$.*

Example 1 (Semirings)**.** *Some well known semirings are*

- $\mathbb{S} = (\mathbb{S}, +, \cdot, 0, 1)$, *for* $\mathbb{S} \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$, *the semiring over the numbers in* $\mathbb{S}$*. It is typically used for arithmetic.*
- $\mathbb{N}_\infty = (\mathbb{N} \cup \{\infty\}, +, \cdot, 0, 1)$*, where* $\infty + n = \infty$ *and* $\infty \cdot m = \infty$*, for* $m \neq 0$*. It is typically used in the context of provenance.*
- $\mathscr{R}_{\texttt{max}} = (\mathbb{Q} \cup \{-\infty, \infty\}, \max, +, -\infty, 0)$*, the max tropical semiring. It is typically used in the context of provenance and optimisation.*
- $\mathbb{B} = (\{0,1\}, \vee, \wedge, 0, 1)$*, the Boolean semiring. It is typically used for classical Boolean constraints.*
- $2^A = (2^A, \cup, \cap, \emptyset, A)$*, the powerset semiring over the set A. It is typically used in the context of measure theory but also for set arithmetic and succinct specifications of multiple requirements.*

The connective $\oplus$ (resp. $\otimes$) in a semiring $\mathscr{R}$ is *invertible*, if for every $r \in R$ (resp. $r \in R \setminus \{e_\oplus\}$) some $-r \in R$ (resp. $r^{-1} \in R$) exists s.t. $r \oplus -r = e_\oplus$ (resp. $r \otimes r^{-1} = e_\otimes$); inverse elements are unique allowing us to use $-(\cdot), (\cdot)^{-1}$ as unary connectives. An *ordered semiring* is pair $(\mathscr{R}, >)$, where $\mathscr{R}$ is a semiring and $>$ is a strict total order on R.

For space reasons, we must omit introducing Weighted Logic (Droste and Gastin 2007) explicitly and confine to compare our logic to the one by Droste and Gastin below.

3 ASP($\mathscr{AC}$)

We start by introducing First-Order Weighted HT Logic. Intuitively it generalises First-Order HT Logic by replacing disjunctive connectives ($\vee, \exists$) by additive ones ($+, \Sigma$), conjunctive ones ($\wedge, \forall$) by multiplicative ones ($*, \Pi$), and accordingly, the neutral elements $\bot, \top$ by zero ($e_\oplus$) and one ($e_\otimes$).

Definition 4 (Syntax)**.** *For a signature* $\sigma = \langle \mathscr{D}, \mathscr{P}, \mathscr{X}, \mathscr{S}, r \rangle$*, the* weighted σ-formulas *over the semiring* $\mathscr{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ *are of the form*

$$\alpha ::= k \mid x \mid \phi \mid \alpha \rightarrow_{\mathscr{R}} \alpha \mid \alpha + \alpha \mid \alpha * \alpha \mid -\alpha \mid \alpha^{-1} \mid \Sigma y \alpha \mid \Pi y \alpha,$$

where $k \in R$*,* $x, y \in \mathscr{X}$ *s.t.* $r(s(x)) \subseteq R$ *(i.e., x takes only values from R) and* ϕ *is a* σ*-formula. The use of* $-$ *and* $^{-1}$ *require that* $\oplus$ *and* $\otimes$ *are invertible, the use of* Πy *requires that* $\otimes$ *is commutative. We define* $\neg_{\mathscr{R}} \alpha = \alpha \rightarrow_{\mathscr{R}} e_\oplus$*. A* weighted σ-sentence *is a variable-free weighted* σ*-formula.*

Example 2. *Let* $\sigma = \langle \mathbb{Q}, \{p\}, \{X\}, \{S\}, \{S \mapsto \mathbb{Q}\} \rangle$ *and* $s(X) = S$*; thus, X ranges over the rational numbers. Then* $\Sigma X p(X) * X$ *is a weighted* σ*-sentence over the semirings* $\mathscr{R}_{\max}, \mathbb{Q}$ *but not over* $\mathbb{N}$*.*

Definition 5 (Semantics)**.** *Let* $\sigma = \langle \mathscr{D}, \mathscr{P}, \mathscr{X}, \mathscr{S}, r \rangle$ *be a signature. The semantics of a weighted* σ*-sentence over semiring* $\mathscr{R}$ *w.r.t.* $\mathscr{I}_w = (\mathscr{I}^H, \mathscr{I}^T, w)$ *is inductively defined in Figure 1.*

For the undefined value $e_\oplus^{-1}$ *we use* $e_\oplus$*; here,* $\text{supp}_\odot(\alpha(x), \mathscr{I}_w)$ *is the* support *of* $\alpha(x)$ *w.r.t.* $\mathscr{I}_w$ *and* $\odot \in \{\oplus, \otimes\}$*, defined as*

$$\text{supp}_\odot(\alpha(x), \mathscr{I}_w) = \{\xi \in r(s(x)) \mid [\![\alpha(\xi)]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w) \neq e_\odot\},$$

i.e., the elements ξ *in the range of x with a non-neutral value* $[\![\alpha(\xi)]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w)$ *w.r.t.* $\odot$*.*

Weighted HT Logic is a generalisation of HT Logic in the following sense:

Proposition 6 (Generalisation)**.** *Let* ϕ *be a* σ*-sentence and* $\mathscr{I}_w$ *be a pointed* σ*-HT-interpretation. Then, for the weighted* σ*-sentence* α *over the Boolean semiring* $\mathbb{B}$*, obtained from* ϕ *by replacing* $\bot, \vee, \wedge, \rightarrow, \exists, \forall$ *by* $0, +, *, \rightarrow_{\mathbb{B}}, \Sigma, \Pi$*, respectively, we have* $[\![\alpha]\!]^\sigma_{\mathbb{B}}(\mathscr{I}_w) = 1$ *iff* $\mathscr{I}_w \models_\sigma \phi$*.*

$$[\![k]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w) = k, \text{ for } k \in R \qquad [\![\phi]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w) = \begin{cases} e_\otimes, & \text{if } \mathscr{I}_w \models_\sigma \phi, \\ e_\oplus, & \text{otherwise.} \end{cases}, \text{ for } \sigma\text{-formulas } \phi$$

$$[\![-\alpha]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w) = -([\![\alpha]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w)) \qquad [\![\alpha+\beta]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w) = [\![\alpha]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w) \oplus [\![\beta]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w)$$

$$[\![\alpha^{-1}]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w) = ([\![\alpha]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w))^{-1} \qquad [\![\alpha*\beta]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w) = [\![\alpha]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w) \otimes [\![\beta]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w)$$

$$[\![\alpha \rightarrow_{\mathscr{R}} \beta]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w) = \begin{cases} e_\otimes, & \text{if } [\![\alpha]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_{w'}) = e_\oplus \text{ or } [\![\beta]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_{w'}) \neq e_\oplus \text{ for all } w' \geq w, \\ e_\oplus, & \textit{otherwise}. \end{cases}$$

$$[\![\Sigma x \alpha(x)]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w) = \begin{cases} \bigoplus_{\xi \in \mathrm{supp}_\oplus(\alpha(x),\mathscr{I}_w)} [\![\alpha(\xi)]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w), & \text{if } \mathrm{supp}_\oplus(\alpha(x),\mathscr{I}_w) \text{ is finite,} \\ \text{undefined,} & \text{otherwise.} \end{cases}$$

$$[\![\Pi x \alpha(x)]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w) = \begin{cases} \bigotimes_{\xi \in \mathrm{supp}_\otimes(\alpha(x),\mathscr{I}_w)} [\![\alpha(\xi)]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_w), & \text{if } \mathrm{supp}_\otimes(\alpha(x),\mathscr{I}_w) \text{ is finite,} \\ e_\oplus, & \text{if } r(s(x)) \setminus \mathrm{supp}_\oplus(\alpha(x),\mathscr{I}_w) \neq \emptyset, \\ \text{undefined,} & \text{otherwise.} \end{cases}$$

Fig. 1. Semantics of weighted σ-sentences.

The proof of the equivalence of $\rightarrow$ and $\rightarrow_{\mathscr{R}}$ works for arbitrary semirings $\mathscr{R}$: for σ-formulas ϕ,ψ the weighted formulas $\phi \rightarrow \psi$ and $\phi \rightarrow_{\mathscr{R}} \psi$ are equivalent. Thus, we can drop $\mathscr{R}$ from $\rightarrow_{\mathscr{R}}$.

Apart from being an HT Logic, the main difference between ours and the Weighted Logic introduced by Droste and Gastin (2007) is that we allow for the additional connectives $-$, $^{-1}$, and $\rightarrow$ and that ours is first-order over infinite domains instead of second-order over finite words.

Defining a reasonable semantics for the case of infinite support seems challenging in general. For example, $\mathbb{Q}$ is not closed under taking the limit of converging sequences and even in $\mathbb{R}$ not every infinite sum of numbers converges. For ω-continuous semirings such as $\mathbb{N}_\infty$, where both conditions above are satisfied, a definition would be possible (but omitted here).

Intuitively, weighted formulas specify calculations over semirings depending on the truth of formulas. The quantifier Σ allows us to aggregate the values of weighted formulas for all variable assignments using $\oplus$ as the aggregate function.

Example 3 (cont.)**.** *The semantics of $\Sigma X\ p(X)*X$ over $\mathscr{R}_{max}$ is the maximum value x s.t. $p(x)$ holds. As $[\![p(x)*x]\!]^\sigma_{\mathscr{R}_{\max}}(\mathscr{I}_w) \neq e_\oplus = -\infty$ iff $p(x) \in \mathscr{I}^w$, we see that for finite $\mathscr{I}^w$*

$$\begin{aligned}[\![\Sigma X\ p(X)*X]\!]^\sigma_{\mathscr{R}_{\max}}(\mathscr{I}_w) &= \max\{[\![p(x)*x]\!]^\sigma_{\mathscr{R}_{\max}}(\mathscr{I}_w) \mid p(x) \in \mathscr{I}^w, x \in \mathbb{Q}\} \\ &= \max\{0+x \mid p(x) \in \mathscr{I}^w, x \in \mathbb{Q}\} \ = \ \max\{x \mid p(x) \in \mathscr{I}^w\}.\end{aligned}$$

The semantics of weighted formulas is multi-valued in general. In order to return to the Boolean semantics for programs, we define algebraic constraints, which are (in)equations between a semiring value and a weighted formula.

Definition 7 (Algebraic Constraints)**.** *Let $\sigma = \langle \mathscr{D}, \mathscr{P}, \mathscr{X}, \mathscr{S}, r\rangle$ be a signature. An* algebraic constraint *is an expression $k \sim_{\mathscr{R}} \alpha$ or $x \sim_{\mathscr{R}} \alpha$, where $(\mathscr{R}, >)$ is an ordered semiring, α is a* weighted σ-formula *over $\mathscr{R}$, $k \in R$, $x \in \mathscr{X}, r(s(x)) \subseteq R$ and $\sim \in \{>,\geq,=,\leq,<,\not>,\not\geq,\neq,\not\leq,\not<\}$.*

A sentence $k \sim_{\mathscr{R}} \alpha$ is satisfied w.r.t. $\mathscr{I}_w$, if

$$\mathscr{I}_w \models_\sigma k \sim_{\mathscr{R}} \alpha \iff k \sim [\![\alpha]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_{w'}) \textit{ for all } w' \geq w.$$

The syntax of σ-formulas in Section 2 is extended to include algebraic constraints in (1) as a further case. The definitions of satisfaction (Defn. 1) and equilibrium model (Defn. 2) are amended in the obvious way. However, as the semantics of weighted formulas is undefined for infinite supports, there are two variants of interpreting the condition $\mathscr{I}' \subsetneq \mathscr{I} : (\mathscr{I}', \mathscr{I}, H) \not\models_\sigma$

ϕ in the definition of equilibrium models. If we adopt that $\not\models_\sigma$ holds when the semantics is undefined, we end up with *weak* equilibrium models, otherwise with *strong* equilibrium models.

To verify that the semantics of algebraic constraints is in line with the intuition of HT logic, we show that the persistence property is maintained for sentences that include algebraic constraints.

Proposition 8 (Persistence). *For any σ-sentence ϕ and σ-HT-interpretation $(\mathscr{I}^H, \mathscr{I}^T)$, it holds that $\mathscr{I}_H \models_\sigma \phi$ implies $\mathscr{I}_T \models_\sigma \phi$.*

Having established that the semantics behaves as desired, we formally define programs that can contain algebraic constraints in terms of a fragment of the logic over *semiring signatures*.

Definition 9 (Semiring Signature). *A signature $\sigma = \langle \mathscr{D}, \mathscr{P}, \mathscr{X}, \mathscr{S}, r\rangle$ is a* semiring signature *for semirings $\mathscr{R}_1, \ldots, \mathscr{R}_n$, where $\mathscr{R}_i = \langle R_i, \oplus_i, \otimes_i, e_{\oplus_i}, e_{\otimes_i}\rangle$, $i = 1, \ldots, n$, if (i) $\mathscr{S}$ is $2^{\{1,\ldots,n\}}$, (ii) $\mathscr{D}$ contains R_i, for all $i = 1, \ldots, n$, and (iii) $r : \mathscr{S} \to 2^{\mathscr{D}}$ maps $\{i_1, \ldots, i_m\}$ to $\bigcap_{j=1}^m R_{i_j}$.*

Intuitively, if a variable x has sort $\{i_1, \ldots, i_m\}$, then we only want to quantify over those domain-values that are in every semiring $\mathscr{R}_{i_1}$ to $\mathscr{R}_{i_m}$. Imagine for example that a variable x is used as a placeholder for a semiring value in two algebraic constraints, one over $\mathbb{N}$ and one over $\mathbb{Q}$. Then it only makes sense to quantify over domain-values that are contained in $\mathbb{N}$.

Definition 10 ($\mathscr{AC}$-Rules, $\mathscr{AC}$-Programs). *Let $\sigma = \langle \mathscr{D}, \mathscr{P}, \mathscr{X}, \mathscr{S}, r\rangle$ be a semiring signature for $\mathscr{R}_1, \ldots, \mathscr{R}_n$. Then an $\mathscr{AC}$-program is a set of $\mathscr{AC}$-rules of the form*

$$r = H(r) \leftarrow B(r) = \phi \leftarrow \psi_1, \ldots, \psi_n, \neg\theta_1, \ldots, \neg\theta_m, \qquad (2)$$

where each ϕ, ψ_i and θ_j is either a σ-atom or an algebraic constraint over $\mathscr{R}_i$ for some $i = 1, \ldots, n$, in which no quantifiers or nested constraints occur. Furthermore, we require for each variable x occurring in r that $i \in s(x)$ iff x occurs in place of a value from the semiring $\mathscr{R}_i$.

Example 4 (Rules). *The following are examples of $\mathscr{AC}$-rules:*

$$\texttt{loc\_sum}(Y) \leftarrow Y =_{\mathbb{Q}} \texttt{ind}(I) * \texttt{loc\_weight}(I, W) * W \qquad (3)$$

$$\texttt{glob\_sum}(Y) \leftarrow \texttt{glob\_weight}(W), Y =_{\mathbb{Q}} \texttt{ind}(I) * W \qquad (4)$$

Note that in $\mathscr{AC}$-rules quantifiers occur neither in weighted nor in unweighted formulas. Variables are quantified implicitly, depending on their scope defined as follows.

Definition 11 (Local & Global). *A variable x that occurs in an $\mathscr{AC}$-rule r is* local, *if it occurs in r only in weighted formulas, and* global *otherwise. A rule or program is* locally *(resp.* globally*)* ground, *if it has no local (resp. global) variables.*

Example 5 (cont.). *In the previous example Y and I are respectively global and local in both rules, whereas W is local in rule* (3) *and global in rule* (4).

We then quantify global variables universally and local variables "existentially" (i.e. using Σ).

Definition 12 (Program and Rule Semantics). *Let r be an $\mathscr{AC}$-rule of the form* (2) *that contains global variables $x_1, \ldots, x_k$. Its semantics is that of the σ-formula*

$$\forall x_1, \ldots, x_k\ (B_\wedge(r) \to \phi)^\Sigma, \qquad \text{where } B_\wedge(r) = \psi_1 \wedge \cdots \wedge \psi_n \wedge \neg\theta_1 \wedge \cdots \wedge \neg\theta_m$$

and $(\cdot)^\Sigma$ replaces every weighted formula α with local variables $y_1, \ldots, y_l$ by $\Sigma y_1, \ldots, y_l\ \alpha$.

Construct	ASP($\mathscr{AC}$)	Others
Nested Expressions	$1 =_{\mathbb{B}} \alpha \leftarrow 1 =_{\mathbb{B}} \beta$	$\alpha \leftarrow \beta$
Aggregates	$T \sim_{\mathbb{Q}} (p(X) + q(X)) * X$	$T \sim \mathtt{sum}\{X : p(X), X : q(X)\}$
Choice	$k \leq^{c}_{\mathscr{R}} \neg\neg q(X,W) * (q(X,W) \to p(X)) * W$	$k \leq \{p(X) : q(X,W) = W\}$
Minimised Choice	$k \leq_{\mathscr{R}} \neg\neg q(X,W) * (q(X,W) \to p(X)) * W$	n/a
Value Guess	$k \leq_{\mathscr{R}} \mathtt{val}(X) * X$	$k \leq \mathtt{val}$ (CP+ASP)
Arithmetics	$X =_{\mathbb{Q}} Y * Z^{\mathbf{-1}}$, $s \geq_{2^A} X + Y$	$X = Y \div Z$, $s \supseteq X \cup Y$

Table 1. *Constructs expressible in ASP($\mathscr{AC}$) and how they are expressed in other formalisms.*

Example 6 (cont.). *Consequently, the $\mathscr{AC}$-rules from above correspond to the formulas*

$$\forall Y\ (Y =_{\mathbb{Q}} \Sigma I \Sigma W\ \mathtt{ind}(I) * \mathtt{loc\_weight}(I,W) * W) \to \mathtt{loc\_sum}(Y)$$
$$\forall Y \forall W\ \mathtt{glob\_weight}(W) \wedge (Y =_{\mathbb{Q}} \Sigma I\ \mathtt{ind}(I) * W) \to \mathtt{glob\_sum}(Y).$$

We see that rule (3) *calculates the sum over all indices* $\{i \mid \mathtt{ind}(i)\}$ *weighted locally with* w *when* $\mathtt{loc\_weight}(i,w)$ *holds. Rule* (4) *calculates the sum over all indices* $\{i \mid \mathtt{ind}(i)\}$ *where all of them are weighted with the same weight* w *when* $\mathtt{glob\_weight}(w)$ *holds.*

Note that we strongly restricted the weighted formulas that are allowed in $\mathscr{AC}$-programs. The quantifier Π and nested algebraic constraints are unavailable and Σ quantifiers can only occur as a prefix. Removing these restrictions would lead to a much higher complexity. Already constraint evaluation would be PSPACE-hard for any non-trivial semiring. In addition, our choice allows us to keep the syntax of $\mathscr{AC}$-programs closer to the one of other programs with constraints.

In the sequel, we drop $\mathscr{AC}$ from $\mathscr{AC}$-rules and $\mathscr{AC}$-programs if no ambiguity arises.

4 Constructs in ASP($\mathscr{AC}$) and in other formalisms

We consider several constructs that we can express in ASP($\mathscr{AC}$) and relate them to constructs known from previous extensions of ASP; a summary is given in Table 1.

Nested Expressions The logic programs with arbitrary propositional formulas defined in (Lifschitz et al. 1999) are modelled simply using constraints over the Boolean semiring $\mathbb{B}$. As a special case, this shows the expressibility of disjunctive logic programs using $1 =_{\mathbb{B}} a_1 + \ldots + a_n \leftarrow B(r)$.

Conditionals Cabalar et al. (2020b) defined two semantics for conditionals $s = (s'|s'' : \phi)$, where s', s'' are terms and ϕ is a (quantifier-free) formula. They are named *vicious circle (vc)* and *definedness (df)*, respectively. Given an interpretation $(\mathscr{I}^H, \mathscr{I}^T)$,

$$vc_{\mathscr{I}_w}(s) = \begin{cases} s', & \text{if } \mathscr{I}_w \models_\sigma \phi, \\ s'', & \text{if } \mathscr{I}_w \models_\sigma \neg\phi, \\ \text{undefined}, & \text{otherwise.} \end{cases} \qquad df_{\mathscr{I}_w}(s) = \begin{cases} s', & \text{if } \mathscr{I}_w \models_\sigma \phi, \\ s'', & \text{otherwise.} \end{cases}$$

Syntax and semantics of weighted formulas could be readily extended to include these constructs. We present instead an alternative evaluation of conditionals as formulas $s' * \phi + s'' * \neg\phi$. Then

$$[\![s' * \phi + s'' * \neg\phi]\!]^{\sigma}_{\mathscr{R}}(\mathscr{I}_w) = \begin{cases} s', & \text{if } \mathscr{I}_w \models_\sigma \phi, \\ s'', & \text{if } \mathscr{I}_w \models_\sigma \neg\phi, \\ e_\oplus, & \text{otherwise.} \end{cases} \qquad (5)$$

That is, when neither ϕ nor $\neg\phi$ is satisfied, we end up with the neutral element $e_\oplus$. Consider the following rules r_1 and r_2:

$$r_1 = p \leftarrow \top = (\top \mid \bot : p) \vee \top \qquad\qquad r_2 = p \leftarrow \top = (\top \mid \top : p).$$

According to *vc* resp. *df*, they are equivalent: under *vc*, both have no stable model while under *df* both have the stable model $\{p\}$. We may expect that r_1 has the stable model $\{p\}$ as the formula $s \vee \top$ is equivalent to $\top$ regardless of the value of s. Therefore, the value of p should not influence the truth of the body of r_1. On the other hand, the value of the conditional in r_2 influences the truth of the body of r_2 and it depends on p. Therefore, if $\{p\}$ were a stable model of r_2, we would arguably derive p using the truth value of p. Accordingly, we may expect that r_2 does not have a stable model. These expectations align with the semantics for r_1 and r_2 from (5) above. This evaluation combines the ideas behind the *vc* and the *df* principle: the value of a conditional is always defined, but the vicious circle of deriving p by the truth value of p is avoided.

Apart from that, we may express *vc* and *df* in our formalism: adding the constraint $1 =_{\mathbb{B}} \phi + \neg\phi$ to a rule using a conditional on ϕ as in (5) corresponds to *vc* semantics. The constraint enforces that when the rule fires, ϕ either must be false already at world T, or when ϕ is true at T then it must also be forced to hold at world H by the rest of the program. If this is not the case, then $\mathscr{I}_T \models 1 =_{\mathbb{B}} \phi + \neg\phi$ but $\mathscr{I}_H \models 0 =_{\mathbb{B}} \phi + \neg\phi$. Thus, in this case, any rule containing this constraint in the body is trivially satisfied. Furthermore, provided the addition in $\mathscr{R}$ is invertible, we can use $\phi * s' + (e_\otimes + -\phi) * s''$ to capture *df* (cf. Appendix in the TPLP archives for more details and discussion). Summarising, the possibility to express *vc* and *df* as well as to define other semantics of conditionals exemplifies the power of FO-WHT Logic and $\mathscr{AC}$-programs.

Constraints in the head for guessing In many ASP extensions, constraints in rule heads and rule bodies behave differently; in heads, they are used as *choice constraints*. Consider for example the rule $10\{p(X) : q(X)\}$. in lparse syntax. Any interpretation s.t. $p(x)$ holds for ten or more values x can be stable. In order to express this constraint in our semantics, we need to take care of two aspects. The first one is that the above constraint only supports $p(x)$ for x s.t. $q(x)$ was already derived in another way. If we simply use the rule $10 \leq_{\mathbb{N}} p(X) * q(X) \leftarrow$ it can also derive $q(x)$ instead of using it as a precondition. We can however use instead the formula $\neg\neg q(X) * (q(X) \rightarrow p(X))$ to achieve the desired effect. More generally, we use the pattern

$$\alpha(X,W) = \neg\neg q(X,W) * (q(X,W) \rightarrow p(X)) * W. \tag{6}$$

Abstractly, it ensures that $p(x)$ can only be asserted for x s.t. we already know that $q(x,w)$ holds, so we cannot "invent" new constants. This can be seen as follows. Assume $\mathscr{I}$ contains $q(x,w), p(x)$. Then $[\![\alpha(X)]\!]_{\mathbb{N}}(\mathscr{I},\mathscr{I},T)$ is equal to $[\![\alpha(X)]\!]_{\mathbb{N}}(\mathscr{I} \setminus \{q(x,w),p(x)\},\mathscr{I},H)$ but unequal to $[\![\alpha(X)]\!]_{\mathbb{N}}(\mathscr{I} \setminus \{p(x)\},\mathscr{I},H)$. The variable W assigns the addition of $p(x)$ a weight w.

Secondly, given the rule $10 \leq_{\mathbb{N}} \neg\neg q(X) * (q(X) \rightarrow p(X)) \leftarrow$ only interpretations that assert $p(x)$ for exactly ten elements x can be stable. While such *minimised constraints* are useful in a different context, we also need to be able to specify choice constraints in our language. This can be achieved naturally without extending the semantics of our language, by introducing a syntactic shorthand $k \sim^c_{\mathscr{R}} \alpha$ for *algebraic choice constraints* in rule-heads. We define that

$$r = k \sim^c_{\mathscr{R}} \alpha \leftarrow B(r) \quad \text{stands for} \quad k \sim_{\mathscr{R}} \alpha \leftarrow B(r) \;\text{ and }\; X =_{\mathscr{R}} \alpha \leftarrow X =_{\mathscr{R}} \alpha^{\neg\neg}, B(r), \tag{7}$$

where $\alpha^{\neg\neg}$ is obtained from α by adding $\neg\neg$ in front of each atom $p(\vec{x})$. These algebraic choice constraints behave as expected of choice constraints. (The next proposition considers only globally ground rules in order to decrease the amount of syntactic noise.)

Proposition 13 (Choice Semantics)**.** *For any σ-HT-interpretation $(\mathscr{I}^H, \mathscr{I}^T)$ and any rule $r = k \sim^c_{\mathscr{R}} \alpha \leftarrow B(r)$, it holds that $\mathscr{I}_H \models_\sigma r$ iff (i) $\mathscr{I}_H \models_\sigma k \sim_{\mathscr{R}} \alpha \leftarrow B(r)$ and (ii) $\mathscr{I}_H \models_\sigma (B_\wedge(r))^\Sigma$ implies $[\![(\alpha)^\Sigma]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_T) = [\![(\alpha)^\Sigma]\!]^\sigma_{\mathscr{R}}(\mathscr{I}_H)$.*

Proof (sketch). Any satisfying interpretation has to satisfy both $k \sim_{\mathscr{R}} \alpha \leftarrow B(r)$ and $X =_{\mathscr{R}} \alpha \leftarrow X =_{\mathscr{R}} \alpha^{\neg\neg}, B(r)$. The first of the two says that the minimised constraint has to be satisfied when $B(r)$ is satisfied and corresponds to (i). The second says that when $B(r)$ is satisfied and $\alpha^{\neg\neg}$ has value X then also α needs to have value X. Since the value of $\alpha^{\neg\neg}$ is the value of α under $\mathscr{I}_T$ the second rule corresponds to (ii). $\square$

Choice constraints are already well-known from previous ASP extensions, so we do not explain them in more detail. The usefulness of the novel minimised choice constraints is demonstrated in the following example.

Example 7 (Integer Subset Sum)**.** *Consider the following variation of the Subset Sum Problem: Given a set $S \subseteq \mathbb{Z}$ and two bounds $l, u \in \mathbb{Z}$, determine a $\subseteq$-minimal solution $S' \subseteq S$ such that $l \leq \sum_{x \in S'} x \leq u$. When $\mathtt{s}(x)$ holds for $x \in S$, we can use the $\mathscr{AC}$-rules*

$$l \leq_{\mathbb{Z}} \neg\neg\mathtt{s}(X) * (\mathtt{s}(X) \rightarrow \mathtt{in}(X)) * X \leftarrow \quad \textit{and} \quad u \geq_{\mathbb{Z}} \neg\neg\mathtt{s}(X) * (\mathtt{s}(X) \rightarrow \mathtt{in}(X)) * X \leftarrow .$$

For every equilibrium model $\mathscr{I}$ the set $S' = \{x \mid \mathtt{in}(x) \in \mathscr{I}\}$ is a $\subseteq$-minimal solution and for every $\subseteq$-minimal solution there exists an equilibrium model. When using choice constraints, i.e. replacing $\sim_{\mathbb{Z}}$ by $\sim^c_{\mathbb{Z}}$, the program still obtains solutions, but not only $\subseteq$-minimal ones.

Aggregates As can be seen in Example 3, we can model aggregates whose aggregation function is the addition of some semiring. This restriction is mild in practice: The aggregates `min`, `max`, `sum`, `count` are expressible using a single algebraic constraint. `times` and `avg` are expressible using multiple algebraic constraints (e.g. `avg` is `sum` divided by `count`).

Value Guessing and Arithmetic Operators Value guessing and arithmetic operators are especially used in combinations of ASP and CP (Lierler 2014). We can guess a value from a semiring, perform arithmetic operations over semirings and evaluate (in)equations on the results. Again, we are mildly restricted as only semiring operations are available.

5 Provenance

Green at al. (2007) introduced a semiring-based semantics that is capable of expressing bag semantics, why-provenance and more. For positive logic programs, their semantics over a semiring $(R, \oplus, \otimes, e_\oplus, e_\otimes)$ is as follows: the label of a query result $q(\vec{x})$ is the sum (using $\oplus$) of the labels of derivation trees for $q(\vec{x})$, where the label of a derivation tree is the product (using $\otimes$) of the labels of the leaf nodes (i.e. extensional atoms). As the number of derivation trees may be countably infinite, Green et al. used ω-continuous semirings such as $\mathbb{N}_\infty$ that allow to have countable sums.

Example 8 (Bag Semantics)**.** *For ease of exposition, consider the propositional program*

$$r_1\colon b \leftarrow e_1, e_2 \qquad r_2\colon b \leftarrow e_1 \qquad r_3\colon c \leftarrow e_2, b \qquad r_4\colon c \leftarrow c, c$$

over $\mathbb{N}_\infty$ (i.e. with bag semantics) and the extensional database (edb) $\{(e_1, 2), (e_2, 0)\}$. The label *of b under bag semantics is $2+0\cdot 2=2$. Here 2 corresponds to the derivation from $r_2,(e_1,2)$ and $0\cdot 2$ to the derivation from $r_1,(e_1,2),(e_2,0)$. The label of c is 0 as it can only be derived using e_2.*

We can model the semiring semantics in our formalism, by allowing operations over countable supports $\mathrm{supp}_{\odot}(\alpha(x),\mathscr{I}_w)$ for ω-continuous semirings. Over $\mathbb{N}_\infty$ they always have the value ∞.

Example 9 (cont.). *The following $\mathscr{AC}$-program calculates the provenance semantics over $\mathbb{N}_\infty$ for the above positive logic program, depending on the edb:*

$$1 =_{\mathbb{B}} p(b,1,2,X)*d(b,2) \leftarrow p(e_1,1,X_1),p(e_2,1,X_2),X =_{\mathbb{N}_\infty} X_1+X_2 \tag{8}$$

$$1 =_{\mathbb{B}} p(b,2,1,X)*d(b,1) \leftarrow p(e_1,1,X) \tag{9}$$

$$1 =_{\mathbb{B}} p(c,3,V,X)*d(c,V) \leftarrow p(e_2,1,X_1),p(b,V_1,X_2),V =_{\mathbb{N}_\infty} V_1+1,X =_{\mathbb{N}_\infty} X_1+X_2 \tag{10}$$

$$1 =_{\mathbb{B}} p(c,4,V,X)*d(c,V) \leftarrow p(c,V_1,X_1),p(c,V_2,X_2),V =_{\mathbb{N}_\infty} V_1+1,X =_{\mathbb{N}_\infty} X_1+X_2 \tag{11}$$

$$1 =_{\mathbb{B}} p(A,V,X) \leftarrow d(A,V),X =_{\mathbb{N}_\infty} p(A,I,V,X^*)*X^* \tag{12}$$

$$1 =_{\mathbb{B}} f(A,X) \leftarrow d(A,V),X =_{\mathbb{N}_\infty} p(A,V^*,X^*)*X^* \tag{13}$$

Here $p(A,V,X)$ represents that X is the sum of all labels of derivation trees for A having exactly V many leaf nodes. We obtain this value first for all derivation trees that apply rule r_i last, in $p(A,i,V,X)$, and sum them up in rule (12). Similarly the final provenance value is obtained as the sum over the provenance values for each number of leaf nodes V^* in rule (13); $d(A,V)$ says that there is a derivation tree of A using V leaf nodes and ensures safety (see next section).

We can apply this strategy in general: Even for a non-ground positive logic program we can give an $\mathscr{AC}$-program that computes the provenance semantics. This can be achieved in a similar fashion as in the example above. Details can be found in the appendix in the TPLP archives. Exploring extensions of Green et al.'s semantics for the provenance of negated formulas remains for future work.

6 Language Aspects

Domain Independence and Safety We need to restrict ourselves to programs that are well behaved, i.e. independent of the domain they are evaluated over.

Example 10. *Consider the weighted formula $\alpha = \Sigma x\, \neg q(x)$, which counts the elements d in the domain s.t. $q(d)$ does not hold. It is easy to see that if we consider the semantics using the same interpretation but over different domains (or rather signatures) it can vary.*

We are interested in formulas that do not exhibit this kind of behaviour, formalised as:

Definition 14 (Domain Independence). *A sentence ϕ (resp. weighted sentence α over semiring $\mathscr{R}$) is* domain independent, *if for every two semiring signatures $\sigma_i = \langle \mathscr{D}_i,\mathscr{P},\mathscr{X},\mathscr{S},r_i\rangle (i=1,2)$ s.t. ϕ is a σ_i-formula (resp. α is a weighted σ_i-formula) for $i=1,2$ and every $\mathscr{I}_w = (\mathscr{I}^H,\mathscr{I}^T,w)$ that is a pointed σ_i-HT-interpretation for $i=1,2$ it holds that*

$$\mathscr{I}_w \models_{\sigma_1} \phi \text{ iff } \mathscr{I}_w \models_{\sigma_2} \phi \quad (\text{resp. } [\![\alpha]\!]^{\sigma_1}_{\mathscr{R}}(\mathscr{I}_w) = [\![\alpha]\!]^{\sigma_2}_{\mathscr{R}}(\mathscr{I}_w)).$$

We restrict ourselves to a fragment of weighted formulas. Intuitively, we need to ensure that every variable X in $\alpha(\vec{X})$ is bound by a positive occurrence of a predicate $p(X)$.

Definition 15 (Syntactic Domain Independence)**.** *A weighted formula $\alpha(\vec{X})$ over a semiring $\mathscr{R}$ is* syntactically domain independent *w.r.t. $\vec{X}$, if it is constructible following*

$$\phi(\vec{X}) ::= \bot \mid p(\vec{X}) \mid \neg\neg\phi(\vec{X}) \mid \phi(\vec{X}) \vee \phi(\vec{X}) \mid \phi(\vec{Y}) \wedge \phi(\vec{Z}) \mid \phi(\vec{X}) \wedge \psi(\vec{X}'),$$
$$\alpha(\vec{X}) ::= k \mid \phi(\vec{X}) \mid \neg\neg\alpha(\vec{X}) \mid \alpha(\vec{X}) + \alpha(\vec{X}) \mid \alpha(\vec{Y}) * \alpha(\vec{Z}) \mid \alpha(\vec{X}) * \beta(\vec{X}') \mid -\alpha(\vec{X}) \mid \alpha^{-1}(\vec{X}),$$

where $k \in R$, $p(\vec{X})$ is an atom, $\psi(\vec{X}')$ ($\beta(\vec{X}')$) is any (weighted) formula, $\vec{X}' \subseteq \vec{X}$ and $\vec{Y} \cup \vec{Z} = \vec{X}$.

Example 11 (cont.)**.** *While $\neg q(Y)$ from Example 10 is not syntactically domain independent w.r.t. Y, the formula $p(Y) * \neg q(Y)$, which counts the number d s.t. $p(d)$ holds but not $q(d)$, is. It can be constructed using $\alpha(\vec{X}) * \beta(\vec{X}')$.*

Our syntactic criterion guarantees domain independence.

Theorem 16 (Formula Domain Independence)**.** *If a formula $\alpha(\vec{X})$ over semiring $\mathscr{R}$ is syntactically domain independent w.r.t. $\vec{X}$, then $\alpha^{\Sigma} = \Sigma\vec{X}\ \alpha(\vec{X})$ is domain independent.*

Proof (sketch). Invariance of $\mathrm{supp}_{\oplus}(\alpha(x), \mathscr{I}_w)$ w.r.t. σ_i (or rather $\mathscr{D}_i$) is shown by structural induction. We show the invariance for one interesting case, namely $\alpha = \alpha_1(x) * \alpha_2$. Note that:

$$\{\xi \in r_1(s(x)) \mid [\![\alpha_1(\xi) * \alpha_2]\!]^{\sigma_1}_{\mathscr{R}}(\mathscr{I}_w) \neq e_{\oplus}\} \subseteq \{\xi \in r_1(s(x)) \mid [\![\alpha_1(\xi)]\!]^{\sigma_1}_{\mathscr{R}}(\mathscr{I}_w) \neq e_{\oplus}\}$$

Therefore, we obtain

$$\begin{aligned}&\{\xi \in r_1(s(x)) \mid [\![\alpha_1(\xi) * \alpha_2]\!]^{\sigma_1}_{\mathscr{R}}(\mathscr{I}_w) \neq e_{\oplus}\}\\ =&\{\xi \in \{\xi \in r_1(s(x)) \mid [\![\alpha_1(\xi)]\!]^{\sigma_1}_{\mathscr{R}}(\mathscr{I}_w) \neq e_{\oplus}\} \mid [\![\alpha_1(\xi) * \alpha_2]\!]^{\sigma_1}_{\mathscr{R}}(\mathscr{I}_w) \neq e_{\oplus}\}\end{aligned}$$

Next, we use the induction hypothesis on $\alpha_1(x)$ to obtain

$$\begin{aligned}=&\{\xi \in \{\xi \in r_2(s(x)) \mid [\![\alpha_1(\xi)]\!]^{\sigma_2}_{\mathscr{R}}(\mathscr{I}_w) \neq e_{\oplus}\} \mid [\![\alpha_1(\xi) * \alpha_2]\!]^{\sigma_2}_{\mathscr{R}}(\mathscr{I}_w) \neq e_{\oplus}\}\\ =&\{\xi \in r_2(s(x)) \mid [\![\alpha_1(\xi) * \alpha_2]\!]^{\sigma_2}_{\mathscr{R}}(\mathscr{I}_w) \neq e_{\oplus}\}.\end{aligned}$$

As the semantics of variable-free formulas is domain independent the claim follows. □

Safety of programs is defined as follows.

Definition 17 (Safety)**.** *A program Π is safe, if each rule $r \in \Pi$ of form* (2) *is safe, i.e. fulfills that*

(i) *every weighted formula in r is syntactically domain independent w.r.t. its local variables;*
(ii) *for every global variable X there exists some β_i s.t. (1) β_i is an atom and X occurs in it, or (2) β_i is $X =_{\mathscr{R}} \beta_i'$ and X does not occur in any weighted formula in the body of r.*

The restriction in (ii) that X does not reoccur is necessary to prohibit $p(X) \leftarrow X =_{\mathscr{R}} Y, Y =_{\mathscr{R}} X$. It could however be replaced by a more sophisticated acyclicity condition.

Example 12 (Safety)**.** *The rules* (3) *and* (4) *are safe. Without the predicate d the program in Example 9 would not be safe.*

Theorem 18 (Program Domain Independence)**.** *Safe programs are domain independent.*

Proof (sketch). Let $\sigma_i = \langle \mathscr{D}_i, \mathscr{P}, \mathscr{X}, \mathscr{S}, r_i \rangle, i = 1,2$ be semiring signatures s.t. Π is a σ_i-formula for $i = 1,2$ and let $\mathscr{I}_w = (\mathscr{I}^H, \mathscr{I}^T, w)$ be a pointed σ_i-HT-interpretation for $i = 1,2$.

Let $r \in \Pi$. If r does not contain global variables, the claim is evident. Otherwise assume $r = \forall x_1, \dots, x_n\ \alpha(x_1, \dots, x_n)$. When $\xi_j \in r_1(s(x_j)) \cap r_2(s(x_j))$ $(j = 1, \dots, n)$, the semantics of $\alpha(\xi_1, \dots, \xi_n)$ does not depend on σ_i. Suppose that $\xi_j \in r_1(s(x_j)) \setminus r_2(s(x_j))$. Then x_j cannot occur in place of a semiring value as for semiring signatures, we have $r_1(s(x_j)) = \bigcap_{j=1}^{m} R_{k_j} = r_2(s(x_j))$. Therefore x_j has to satisfy item (ii.1) of safety, implying that some atom β_k in the body of r is not satisfied by $\mathscr{I}_w$ and hence $\mathscr{I}_w \models_{\sigma_i} \alpha(\xi_1, \dots, \xi_n)$ for $i = 1, 2$. □

Not every domain independent program is safe. E.g. $p(X) \leftarrow \top =_{\mathbb{B}} q(X)$ is not safe but is equivalent to the safe rule $p(X) \leftarrow q(X)$ since X is a global variable and we can only derive $p(x)$ when $[\![q(x)]\!]_{\mathbb{B}} = 1$, i.e. when $q(x)$ holds. Domain independence is undecidable but safety is sufficient, allows for complex rules like (3), (4) and those in Example 9, and is easily checked.

In the rest of the paper, we restrict ourselves to domain independent programs and can therefore remove the annotation σ from $\models_\sigma$ and $[\![\cdot]\!]^{\sigma}_{\mathscr{R}}$ and use $\models$ and $[\![\cdot]\!]_{\mathscr{R}}$ instead. Accordingly, we do not need to specify the signature for $\mathscr{AC}$-programs Π anymore, as any semiring signature σ s.t. Π is an $\mathscr{AC}$-program over σ suffices.

Program Equivalence An additional benefit of HT-semantics is that we are able to characterise strong program equivalence as equivalence in the logic of HT.

Definition 19 (Strong Equivalence)**.** *Programs Π_1 and Π_2 are strongly equivalent, denoted by $\Pi_1 \equiv_s \Pi_2$, if for every program Π' the equilibrium models of $\Pi_1 \cup \Pi'$ and $\Pi_2 \cup \Pi'$ coincide.*

Similar results have already been proven for classical programs with (Pearce and Valverde 2008) or without variables (Lifschitz et al. 2001) and many more. As with classical programs:

Theorem 20. *For any Π_1, Π_2 programs, $\Pi_1 \equiv_s \Pi_2$ iff Π_1 has the same HT-models, i.e. satisfying pointed HT-interpretations, as Π_2.*

Proof (sketch). The direction $\Leftarrow$ is clear. For $\Rightarrow$ we can generalise the proof of Lifschitz et al. (2001), by constructing Π', which asserts a subset of the interpretation $\mathscr{I}^T$ that is ensured to be stable ($\mathscr{I}^H$), and a subset that if partly present is ensured to be fully present ($\mathscr{I}^T \setminus \mathscr{I}^H$).

Let Π_1 and Π_2 have different HT-models. W.l.o.g. there must be at least one HT-interpretation $(\mathscr{I}^H, \mathscr{I}^T)$ that is an HT-model of Π_1 but not of Π_2. Like Lifschitz et al. (2001) we simply define

$$\Pi' = \{p(\vec{x}) \leftarrow \mid p(\vec{x}) \in \mathscr{I}^H\} \cup \{p(\vec{x}) \leftarrow q(\vec{y}) \mid p(\vec{x}), q(\vec{y}) \in \mathscr{I}^T \setminus \mathscr{I}^H\}$$

Then $\mathscr{I}^T$ is an equilibrium model of $\Pi_2 \cup \Pi'$, but not of $\Pi_1 \cup \Pi'$ and therefore Π_1 and Π_2 are not strongly equivalent. □

Note that since $\mathscr{I}^H$ may be infinite, this may result in programs of infinite size. This can be circumvented if auxiliary predicates are allowed in Π' (see the Appendix in the TPLP archives).

7 Computational Complexity

We consider the computational complexity of the following problems:

- Model Checking (MC): Given a safe program Π and an interpretation $\mathscr{I}$ of Π, is $\mathscr{I}$ is an equilibrium model of Π?
- Satisfiability (SAT): Given a safe program Π, does Π have an equilibrium model?
- Strong Equivalence (SE): Given safe programs Π_1, Π_2, are Π_1 and Π_2 strongly equivalent?

The main factor that complicates these problems is that we may have to evaluate weighted formulas over an arbitrary semiring. If we want to prevent an increase in complexity, then we need to encode the elements of the semiring in some way which allows for efficient calculations and comparison. To this end, we use the previously introduced (Eiter and Kiesel 2020) condition of Efficient Encodability.

Definition 21 (Efficiently Encodable Semiring). *Let $\mathscr{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ be a semiring with strict order $>$ and $e : R \to \mathbb{N}$ a polynomially computable, injective function. We use $\|r\| = \log_2(e(r))$ for the length of $e(r)$'s representation. We say $\mathscr{R}$ is* efficiently encoded *by e if*

- *some $c \in \mathbb{N}$ exists such that for $r_1, r_2 \in R'$ and $\odot \in \{\oplus, \otimes\}$*
$$\|r_1 \odot r_2\| \leq \|r_1\| + \|r_2\| + c \quad \text{and} \quad \max(\|-r_1\|, \|r_1^{-1}\|) \leq \|r_1\| + c; \tag{14}$$
- *we can compute $e(r_1 \odot r_2), e(-r_1), e(r_1^{-1})$ in polynomial time given $e(r_1), e(r_2)$ resp. $e(r_1)$;*
- *$r_1 > r_2$ is decidable in time polynomial in $\|r_1\| + \|r_2\|$.*

This restriction is mild in practice; for example $\mathbb{B}$, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathscr{R}_{\max}$, 2^A are efficiently encodable.

Theorem 22 (Ground Complexity). *For variable-free programs over efficiently encoded semirings (i) MC and (propositional) SE are co-NP-complete, and (ii) SAT is Σ_2^p-complete.*

Proof (sketch). The hardness parts are inherited from disjunctive logic programs (Dantsin et al. 2001), cf. Section 4, resp. HT-Logic (Lifschitz et al. 2001). The membership parts result by guess and check algorithms: for similar bounds as in ordinary ASP, we just need that $\mathscr{I}_H \models k \sim_{\mathscr{R}} \alpha$ is polynomially decidable given $(\mathscr{I}^H, \mathscr{I}^T)$ and $k \sim_{\mathscr{R}} \alpha$; as $\mathscr{R}$ is efficiently encoded, this holds. □

The non-ground complexity is significantly higher.

Theorem 23 (Non-ground Complexity). *For safe programs over efficiently encoded semirings (i) MC is in EXPTIME, both co-NP^{PP}-hard and NP^{PP}-hard, and (ii) SAT and SE are undecidable.*

Proof (sketch). (i) Given an interpretation $\mathscr{I}$ as a set of ground atoms, we check $(\mathscr{I}, \mathscr{I}, H) \models r'$ and $(\mathscr{I}', \mathscr{I}, H) \not\models r'$ for each $\mathscr{I}' \subsetneq \mathscr{I}$ and each ground instance r' of a rule $r \in \Pi$ in exponential time. The evaluation of algebraic constraints $k \sim_{\mathscr{R}} \alpha$ is feasible in exponential time, since if α is of the form $\Sigma y_1, \ldots, y_n \alpha'(y_1, \ldots, y_n)$ where α' is quantifier-free, by safety of the program each y_i must occur in some atom $p(\vec{x})$. That is, to evaluate α, we only need to consider values $\xi(y_i)$ for y_i, $i = 1, \ldots,_n$ that occur in the interpretation $\mathscr{I}$. There are exponentially many such ξ; for each of them, the value of $\alpha'(\xi(y_1), \ldots, \xi(y_n))$ can be computed in polynomial time given that $\mathscr{R}$ is efficiently encoded, yielding a value r_ξ such that $e(r_\xi)$ occupies polynomially many bits. The aggregation $\Sigma_\xi\, r_\xi$ over all ξ is then feasible in exponential time by the assertion that $\|r_1 \oplus r_2\| \leq \|r_1\| + \|r_2\| + c$ and that $e(r_1 \oplus r_2)$ is computable in polynomial time given $e(r_1), e(r_2)$.

The (co-)NP^{PP}-hardness is by a reduction from (co-)E-MAJSAT (Littman et al. 1998), which asks whether for a Boolean formula $\phi(x_1, \ldots, x_n)$ a partial assignment to $x_1, \ldots, x_k$ exists s.t. more than $m = 2^{n-k-1}$ of the assignments to $x_{k+1}, \ldots, x_n$ satisfy $\phi(\vec{x})$. Then the program

$$v(0) \leftarrow; \qquad v(1) \leftarrow; \qquad f \leftarrow v(X_1), \ldots, v(X_k), m <_{\mathbb{N}} v(X_{k+1}) * \ldots * v(X_n) * \phi(\vec{X})$$

has an equilibrium model $\{f, v(0), v(1)\}$ if the answer for E-MAJSAT is yes and an equilibrium model $\{v(0), v(1)\}$ if the answer is no.

(ii) The undecidable Mortal Matrix Problem (MMP) asks whether any product of matrices in $X = \{X_1, \ldots, X_n\} \subset \mathbb{Z}^{d \times d}$ evaluates to the zero matrix 0_d (Cassaigne et al. 2014). The semiring $(\mathbb{Z}^{d \times d}, +, \cdot, 0_d, 1_d)$ is efficiently encodable, and the program Π

$$p(X_1) \leftarrow; \quad \ldots \quad p(X_n) \leftarrow; \qquad \bot \leftarrow \neg p(0_d); \qquad p(Y) \leftarrow p(Z_1), p(Z_2), Y =_{\mathbb{Z}^{d \times d}} Z_1 * Z_2$$

has an equilibrium model iff X is a yes-instance of MMP, as $p(0_d)$ needs to be supported. For undecidability, let Π be the program from above and $\Pi' = \Pi \setminus \{\bot \leftarrow \neg p(0)_d)\}$. As Π' has no negation, its HT-models are the interpretations $(\mathscr{I}', \mathscr{I})$ where both $\mathscr{I}'$ and $\mathscr{I}$ are closed under the rules of Π', sets S such that $p(X_1), \dots, p(X_n) \in S$ and whenever $p(Y), p(Z) \in S$ then also $p(Y * Z) \in S$. Similarly, the HT-models of Π are the interpretations $(\mathscr{I}', \mathscr{I})$ where $\mathscr{I}'$ and $\mathscr{I}$ are closed under the rules of Π' and in addition $p(0_d) \in \mathscr{I}'$.

Therefore, $\Pi \equiv_s \Pi'$ iff $p(0_d) \in L$, where L is the least set closed under the rules of Π', which holds iff the answer for the mortal matrix problem on X is yes. □

As NP$^{\text{PP}}$ contains the polynomial hierarchy (PH), this places MC between PH and EXPTIME; stronger assumptions on the encoding $e(r)$ allow for PSPACE. In particular, for programs over the canonical semiring $\mathbb{N}$, MC is co-NP^{C}-complete for $C =$ NP$^{\text{PP}}$ (while SAT and SE are undecidable). Naïve evaluation of $k \sim_{\mathscr{R}} \alpha$ is infeasible in polynomial space, as $\|[\![\alpha]\!]_{\mathscr{R}}(\mathscr{I}_H)\|$ can be exponential in the number of variables in α. We can retain decidability for SAT and SE by limiting value invention, i.e. constraints $X =_{\mathscr{R}} \alpha(\vec{Y})$, and value guessing. For the latter, we adapt domain restrictedness due to Niemelä et al. (1999).

Definition 24 (Domain Restrictedness)**.** *An algebraic constraint is* domain restricted *in variables $\vec{X}$, if it is of the form*

$$k \sim_{\mathscr{R}} \neg\neg\alpha(\vec{X}) * (\alpha(\vec{X}) \to \beta(\vec{X})) * \gamma(\vec{X}),$$

where $\alpha(\vec{X}), \beta(\vec{X})$ are syntactically domain independent and all atoms in $\gamma(\vec{X})$ are locally ground.

Intuitively, only constants "known" by predicates in α can be "transferred" to predicates in β, and γ assigns a weight to each substitution. The pattern is explained less generally in Section 4. Let us call a semiring *computable* if $\oplus, \otimes, -, ^{-1}, >$ are computable. Then we obtain:

Theorem 25. *For safe programs without value invention where all algebraic constraints in rule heads are domain restricted and all semirings are computable, both SAT and SE are decidable.*

Proof (sketch). For this class of programs we can show that they are finitely groundable, i.e. groundable over a finite domain without changing the answer sets, by using only the constants that occur in a program as the domain. Then the ground programs are variable free and since the semirings are computable both SAT and SE are decidable. □

However, prohibiting value invention entirely is unnecessarily strong. Weaker restrictions like aggregate stratification (Faber et al. 2011) or argument restrictedness (Lierler and Lifschitz 2009) can be adapted to ASP($\mathscr{AC}$); the resulting programs are finitely ground and decidable.

8 Related Work & Conclusion

A number of related works has already been mentioned above; we concentrate here on highlighting the differences of our approach to others.

• Semiring-based Constraint Logic Programming (Bistarelli et al. 1997), aProbLog (Kimmig et al. 2011) and our previous work (2020) also use semiring semantics. However, Bistarelli et al. aimed at semantics for CLP with multi-valued interpretations over lattice-like semirings and the other works aimed at semantics for weighted model counting and model selection over semirings.

• Hybrid ASP by Cabalar et al. (2020b; 2020a). They defined an extension of HT Logic that includes general constraints and multi-valued interpretations for handling mixtures of ASP and CP. The approach integrates conditionals and aggregates; however, it relies on extra definitions to introduce their semantics while our semantics can capture the different constructs natively. The syntax of constraints (apart from over the reals) is left open, while we provide a uniform syntax over any semiring. Moreover, we study domain independence and safety, characterise strong equivalence, and provide complexity results.

• Nested Formulas with Aggregates due to Ferraris and Lifschitz (2010) have semantics similar to that of HT. They allow for arbitrary aggregate functions but only over the integers, whereas we allow for arbitrary values using semiring operations. While defined, the usage of non-ground constraints in rule heads was not considered. We can transfer our results and show that both choice and minimised constraints can be encoded and used safely in the formalism.

• Gelfond-Zhang Aggregates (2014) are semantically different from ours but presumably encodable in ASP($\mathscr{AC}$). Their semantics introduced the vicious circle principle to ASP. Regarding expressiveness, aggregates are not allowed in rule-heads but aggregation functions are arbitrary.

• Arbitrary Constraint Atoms due to Son et al. (2007) as well as (monotone) Abstract Constraint Atoms due to Marek et al. (2006) define semantics for constraints abstractly by allowing them to be specified as a set of alternative sets of atoms that need to be satisfied. Naturally, this gives a semantics to arbitrary constraints, however syntactic shorthands are desirable to avoid an exponential blowup of the representation of the constraints. Marek et al. focus on monotone constraints, showing that their behaviour can be characterised by fixed-points of a non-deterministic operator.

• Weight Constraints with Conditionals by Niemelä et al. (1999) introduced the well known shorthands $k \leq \{p(X) : q(X,W) = W\}$. Our constraints generalise them to arbitrary semirings.

• Formalisms for Intensional Functions in ASP as those of Bartholomew and Lee (2019) as well as Cabalar (2011) define a semantics that allows the definition of functions using ASP. A priori, this differs from our goal aiming at an expressive predicate-based formalism. Nevertheless, Weighted Formulas could be used to specify the values of functions. Semantically we are closer to Cabalar et al.'s approach, where function values can be undefined and the stability condition is more similar.

Summary & Outlook We have seen that algebraic constraints unify many previously proposed constructs for more succinct answer set programs, with low practical restrictions and no increase in the ground complexity. Among other novelties, we can specify whether constraints in rule-heads are minimised or guessed, can explicitly represent values from different sets and give an interesting alternative semantics for conditionals. Overall, the introduced framework opens up new possibilities for expressing programs succinctly and it gives rise to interesting questions.

We currently consider only a fragment of the weighted formulas. It would be interesting to see in the future, if other new and useful constructs can be expressed with a different fragment. Besides this, we want to use the general applicability of HT and Weighted Logic and extend ASP($\mathscr{AC}$) to other domains, like temporal reasoning. Furthermore, an in-depth study of suitable conditions for finite groundability and the non-ground complexity in this context are indispensable for our ongoing work on an implementation.

Acknowledgement

Thanks to the reviewers for their constructive comments. This work has been supported by FWF project W1255-N23 and by FFG project 861263.

References

AZIZ, R. A., CHU, G., AND STUCKEY, P. J. 2013. Stable model semantics for founded bounds. *Theory Pract. Log. Program. 13,* 4-5, 517–532.

BALAI, E., GELFOND, M., AND ZHANG, Y. 2013. Sparc-sorted asp with consistency restoring rules. *arXiv preprint arXiv:1301.1386*.

BARTHOLOMEW, M. AND LEE, J. 2019. First-order stable model semantics with intensional functions. *Artificial Intelligence 273*, 56–93.

BISTARELLI, S., MONTANARI, U., AND ROSSI, F. 1997. Semiring-based constraint logic programming. In *Proc. IJCAI'97*. 352–357.

CABALAR, P. 2011. Functional answer set programming. *Theory and Practice of Logic Programming 11,* 2-3, 203–233.

CABALAR, P., FANDINNO, J., SCHAUB, T., AND WANKO, P. 2020a. An ASP semantics for constraints involving conditional aggregates. *arXiv preprint arXiv:2002.06911*.

CABALAR, P., FANDINNO, J., SCHAUB, T., AND WANKO, P. 2020b. A uniform treatment of aggregates and constraints in hybrid ASP. *arXiv preprint arXiv:2003.04176*.

CASSAIGNE, J., HALAVA, V., HARJU, T., AND NICOLAS, F. 2014. Tighter undecidability bounds for matrix mortality, zero-in-the-corner problems, and more. *CoRR abs/1404.0644*.

DANTSIN, E., EITER, T., GOTTLOB, G., AND VORONKOV, A. 2001. Complexity and expressive power of logic programming. *ACM Comput. Surv. 33,* 3, 374–425.

DROSTE, M. AND GASTIN, P. 2007. Weighted automata and weighted logics. *Theor.Comp.Sci. 380,* 1, 69.

EITER, T. AND KIESEL, R. 2020. Weighted lars for quantitative stream reasoning. In *Proc. ECAI'20*.

FABER, W., PFEIFER, G., AND LEONE, N. 2011. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence 175,* 1, 278–298.

FERRARIS, P. 2011. Logic programs with propositional connectives and aggregates. *ACM TOCL 12,* 4, 25.

FERRARIS, P. AND LIFSCHITZ, V. 2010. On the stable model semantics of first-order formulas with aggregates. In *Proc. International Workshop on Nonmonotonic Reasoning (NMR'10)*.

GELFOND, M. AND ZHANG, Y. 2014. Vicious circle principle and logic programs with aggregates. *Theory and Practice of Logic Programming 14,* 4-5, 587–601.

GREEN, T. J., KARVOUNARAKIS, G., AND TANNEN, V. 2007. Provenance semirings. In *Proc. ACM PODS'07*. ACM, 31–40.

JANHUNEN, T. 2018. Answer set programming related with other solving paradigms. *KI 32,* 2-3, 125–131.

KIMMIG, A., VAN DEN BROECK, G., AND DE RAEDT, L. 2011. An algebraic prolog for reasoning about possible worlds. In *Proc. AAAI'11*.

LIERLER, Y. 2014. Relating constraint answer set programming languages and algorithms. *Artificial Intelligence 207*, 1–22.

LIERLER, Y. AND LIFSCHITZ, V. 2009. One more decidable class of finitely ground programs. In *Proc. ICLP'09*. Springer, 489–493.

LIFSCHITZ, V., PEARCE, D., AND VALVERDE, A. 2001. Strongly equivalent logic programs. *ACM TOCL 2,* 4, 526–541.

LIFSCHITZ, V., TANG, L. R., AND TURNER, H. 1999. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence 25,* 3-4, 369–389.

LITTMAN, M. L., GOLDSMITH, J., AND MUNDHENK, M. 1998. The computational complexity of probabilistic planning. *J. Artificial Intelligence Research 9*, 1–36.

MAREK, V. W., NIEMELA, I., ET AL. 2006. Logic programs with monotone abstract constraint atoms. *arXiv preprint cs/0608103*.

NIEMELÄ, I., SIMONS, P., AND SOININEN, T. 1999. Stable model semantics of weight constraint rules. In *Proc. LPNMR'99*. Springer, 317–331.

PEARCE, D. AND VALVERDE, A. 2008. Quantified equilibrium logic and foundations for answer set programs. In *Proc. ICLP'08*, M. Garcia de la Banda and E. Pontelli, Eds. Springer, 546–560.

SON, T. C., PONTELLI, E., AND TU, P. H. 2007. Answer sets for logic programs with arbitrary abstract constraint atoms. *Journal of Artificial Intelligence Research 29*, 353–389.