algorithms

MDPI

Article

On the Semantics of Hybrid ASP Systems Based on `Clingo`

Pedro Cabalar [1], Jorge Fandinno [2], Torsten Schaub [3,*] and Philipp Wanko [3,*]

[1] Department of Computer Science, University of Corunna, 15008 A Corunna, Spain
[2] College of Information Science & Technology, University of Nebraska at Omaha, Omaha, NE 68182, USA
[3] Department of Computer Science, University of Potsdam, 14469 Potsdam, Germany
* Correspondence: torsten@cs.uni-potsdam.de (T.S.); wanko@cs.uni-potsdam.de (P.W.)

Abstract: Over the last decades, the development of Answer Set Programming (ASP) has brought about an expressive modeling language powered by highly performant systems. At the same time, it gets more and more difficult to provide semantic underpinnings capturing the resulting constructs and inferences. This is even more severe when it comes to hybrid ASP languages and systems that are often needed to handle real-world applications. We address this challenge and introduce the concept of abstract and structured theories that allow us to formally elaborate upon their integration with ASP. We then use this concept to make the semantic characterization of `clingo`'s theory-reasoning framework precise. This provides us with a formal framework in which we can elaborate upon the formal properties of existing hybridizations of `clingo`, such as `clingcon`, `clingo`[DL], and `clingo`[LP].

Keywords: answer set programming; answer set programming modulo theories; hybrid reasoning; semantic foundations

Citation: Cabalar, P.; Fandinno, J.; Schaub, T.; Wanko, P. On the Semantics of Hybrid ASP Systems Based on `Clingo`. *Algorithms* **2023**, *16*, 185. https://doi.org/10.3390/a16040185

Academic Editor: Angelo Montanari

Received: 11 February 2023
Revised: 17 March 2023
Accepted: 22 March 2023
Published: 28 March 2023

1. Introduction

Answer Set Programming (ASP) is about mapping a logic program onto its so-called stable models. Over the decades, stable models have been characterized in various ways [1], somehow underpinning their appropriateness as semantics for logic programs. However, over the same period, the syntax of logic programs has continuously been enriched, equipping ASP with a highly attractive modeling language. This development also brought about much more intricate semantics, as nicely reflected by the long distance traveled from the mathematical apparatus used in the original definition [2] to the one for capturing the input language of the ASP system `clingo` [3].

With the growing range of ASP applications in academia and industry [4], we also witness the emergence of more and more hybrid ASP systems [5,6], similar to the rise of SMT solving from plain SAT solving [7]. From a general perspective, the resulting paradigm of ASP *modulo theories* (AMT) can be seen as a refinement of ASP, in which an external theory certifies some of a program's stable models. A common approach, stemming from lazy theory solving [7], is to view a theory as an oracle that determines the truth of a designated set of (theory) atoms in the program. Meanwhile, this idea has been adapted to the setting of ASP in various ways, most prominently in Constraint ASP [5], extending ASP with linear equations over integers in various ways [8–14]. Beyond this, ASP systems such as `clingo` or `dlvhex` [15] leave the choice of the specific theory largely open. For instance, `clingo` merely requires fixing the interaction between theories and their theory atoms. As attractive as this generality may be from an implementation point of view, it complicates the development of generic semantics that are meaningful to existing systems. Not to mention that in ASP, the integration of theories takes place in a non-monotonic context.

As a matter of fact, most formal accounts of existing hybrid systems are implementation driven and lack a clear elaboration of the underlying semantic principles. We address this by separating the semantic foundations of hybrid languages from the corresponding systems' design decisions. As a result, we present a framework in which we can characterize such decisions semantically. More precisely, we provide an extension of logic programs

to incorporate very general abstract theories for which no predefined syntax is assumed. We extend the semantics of stable models for these programs, dealing with two types of theory atoms called *defined* and *external*. We further classify some types of abstract theories, which we call *structured*, where we may distinguish variables inside atoms, and the semantics can be defined in terms of denotations (via functions assigning values to those variables). Finally, we explain how several existing hybrid ASP systems, such as `clingcon`, `clingo`[DL], and `clingo`[LP], can be accommodated as instances of this general framework.

2. Background

We start considering an alphabet consisting of two disjoint sets, namely, a set $\mathcal{A}$ of *propositional* (*regular*) *atoms* and a set $\mathcal{T}$ of *theory atoms*, whose truth is governed by some external theory. Several theory atoms may represent the same theory entity. We use letters `a`, `s`, and b for atoms in $\mathcal{A}$, $\mathcal{T}$, and $\mathcal{A} \cup \mathcal{T}$, respectively. In `clingo`, these theory atoms are expressions preceded by '&', but their internal syntax is not predetermined: it can be defined by the user to build new extensions. As an example, the system `clingcon` extends the input language of `clingo` with linear equations, represented as theory atoms of the form

$$\texttt{\&sum}\{k_1 * x_1; \ldots; k_n * x_n\} \prec k_0 \tag{1}$$

where for each i such that $0 \leq i \leq n$, each x_i is an integer variable and each $k_i \in \mathbb{Z}$ is an integer constant, whereas $\prec$ is a comparison symbol, such as `<=`, `=`, `!=`, `<`, `>`, `>=`. As we said above, we may have two different theory atoms, such as $\texttt{\&sum}\{x\}\texttt{>0}$ and $\texttt{\&sum}\{\texttt{x}\}\texttt{>=1}$, actually representing the same condition (as a linear equation).

A *literal* is any atom $b \in \mathcal{A} \cup \mathcal{T}$ or its default negation $\neg b$. A $\mathcal{T}$-logic program over $\langle \mathcal{A}, \mathcal{T} \rangle$ is a set of rules of the form

$$b_0 \leftarrow b_1, \ldots, b_n, \neg b_{n+1}, \ldots, \neg b_m \tag{2}$$

where $b_i \in \mathcal{A} \cup \mathcal{T}$ for $1 \leq i \leq m$ and $b_0 \in \mathcal{A} \cup \mathcal{T} \cup \{\bot\}$ with $\bot \notin \mathcal{A} \cup \mathcal{T}$ denoting the falsum constant. We let notations $h(r) \stackrel{\text{def}}{=} b_0$, $B(r)^+ \stackrel{\text{def}}{=} \{b_1, \ldots, b_n\}$, and $B(r)^- \stackrel{\text{def}}{=} \{b_{n+1}, \ldots, b_m\}$ stand for the *head*, the *positive*, and the *negative body atoms* of a rule r of the form of (2). The set of *body atoms* of r is just $B(r) \stackrel{\text{def}}{=} B(r)^+ \cup B(r)^-$. Finally, the sets of *body* and *head atoms* of a program P are respectively defined as $B(P) \stackrel{\text{def}}{=} \bigcup_{r \in P} B(r)$ and $H(P) \stackrel{\text{def}}{=} \{h(r) \mid r \in P\} \setminus \{\bot\}$, as expected. An example of a system using $\mathcal{T}$-logic programs is the aforementioned `clingcon`, whose theory atoms correspond to linear integer equations. We refer to logic programs augmented by such theory atoms as `clingcon`-programs.

Next, we describe the semantics of $\mathcal{T}$-logic programs in `clingo` [16], used for several extensions such as the addition of difference constraints in `clingo`[DL], of linear equations over integers in `clingcon`, and over reals in `clingo`[LP]. In all these systems, we can see the role of the corresponding external theory as a kind of "*certification authority*" that sanctions regular stable models whose theory atoms are in accord with their underlying constraints. To this end, we envisage a two-step process: (1) generate regular stable models and (2) select the ones passing the theory certification. In step (1), we ignore the distinction between atoms in $\mathcal{A}$ and $\mathcal{T}$ and plainly apply the stable model semantics [2]. More precisely, a set $X \subseteq \mathcal{A} \cup \mathcal{T}$ of atoms is a *model* of a $\mathcal{T}$-logic program P over $\langle \mathcal{A}, \mathcal{T} \rangle$, if, for every rule $r \in P$ such that $B(r)^+ \subseteq X$ and $B(r)^- \cap X = \emptyset$, we have $h(r) \in X$. In the case of an integrity constraint with $h(r) = \bot$, note that $\bot$ does not belong to $X \subseteq \mathcal{A} \cup \mathcal{T}$. The stable models of P are defined via the *reduct* of P relative to a set X of atoms, viz.

$$P^X = \{h(r) \leftarrow B(r)^+ \mid r \in P,\ B(r)^- \cap X = \emptyset\}.$$

Then, as usual, a set, X, of atoms is a *stable model* of P if X is the least model of P^X. In step (2), an external theory $\mathfrak{T}$ is used to eliminate any stable model X whose theory atoms $X \cap \mathcal{T}$ are not *in accord* with $\mathfrak{T}$. Suppose step (1) yields a stable model of a `clingcon`-program

containing both atoms $\texttt{\&sum}\{x; 2*y\}\texttt{>9}$ and $\texttt{\&sum}\{2*x; 4*y\}\texttt{<=6}$. This stable model would not pass theory certification since no values for integer variables x and y can satisfy both corresponding linear equations, viz. $x + 2y > 9$ and $2x + 4y \leq 6$ (note that the latter is equivalent to $x + 2y \leq 3$).

In fact, the treatment of each theory atom $\texttt{s}$ (as obtained in the stable model) leaves room for different semantic options [16]. The first option is about the justification for the truth of $\texttt{s}$ in step (1). A theory atom $\texttt{s}$ is called *defined* if its truth must be derived through the program's rules. Otherwise, $\texttt{s}$ is said to be *external*, and we are always free to add it to the program without further justification. A second semantic option has to do with how we treat $\texttt{s} \notin X$ for a stable model X in step (2). We say that $\texttt{s}$ is *strict* when $\texttt{s} \notin X$ implies that the theory check requires the "opposite" constraint of $\texttt{s}$ to be satisfied. Otherwise, we say that $\texttt{s}$ is *non-strict*, and the fact $\texttt{s} \notin X$ has no relevant effect during theory certification. Of course, in both cases, $\texttt{s} \in X$ implies that the constraint of $\texttt{s}$ is satisfied. As an example, suppose that $\texttt{s}$ is the strict atom $\texttt{\&sum}\{2*x; 4*y\}\texttt{<=6}$, and we get $\texttt{s} \notin X$ in some stable model X. Then, step (2) imposes the same constraint as if we had obtained atom $\texttt{\&sum}\{2*x; 4*y\}\texttt{>6}$ in X.

Janhunen et al. [14] argue that combinations (non-strict,defined) and (strict,external) are the only sensible ones for linear equation theories. With our focus on these theories, we follow this proposition, and partition the set $\mathcal{T}$ of theory atoms into two disjoint sets, $\mathcal{E}$ and $\mathcal{F}$, standing for *external* and *founded* atoms, respectively. Intuitively, each external atom in $\mathcal{E}$ is interpreted as (strict,external), so its truth requires no justification, whereas if we decide to make it false, then its corresponding opposite constraint must be considered in step (2). On the other hand, founded atoms in $\mathcal{F}$ are those interpreted in a (non-strict,defined) manner, so they must be derived by the $\mathcal{T}$-logic program, and when they remain eventually false, they do not impose any additional constraint on the final solution. Additionally, we require that founded atoms do not occur in the rule bodies, that is, $B(P) \cap \mathcal{F} = \emptyset$, for any program P. We do allow external atoms to occur in the head, but, as we see later, they can always be equivalently shifted to the body. For the sake of brevity, we assume that an atom that only occurs in rule heads (but not in bodies) is founded unless explicitly stated otherwise. Let us emphasize that any atom that occurs in some body is mandatorily external, as discussed above. The signature of a program, P, as described above, is denoted as $\langle \mathcal{A}, \mathcal{T}, \mathcal{E} \rangle$, leaving $\mathcal{F}$ implicit.

With this restriction to two interpretations of theory atoms, we can formulate `clingo`'s semantics for $\mathcal{T}$-logic programs as follows [16]: Informally speaking, a $\mathfrak{T}$-solution S is a subset $S \subseteq \mathcal{T}$ of theory atoms whose associated constraints are "sanctioned" by abstract theory $\mathfrak{T}$; this is made precise in Section 3. A set $X \subseteq \mathcal{A} \cup \mathcal{T}$ of atoms is a $\mathfrak{T}$-stable model of a $\mathcal{T}$-logic program P over $\langle \mathcal{A}, \mathcal{T}, \mathcal{E} \rangle$, if there is some $\mathfrak{T}$-solution S such that X is a stable model of the logic program

$$P \cup \{\texttt{s} \leftarrow \mid \texttt{s} \in (S \cap \mathcal{E})\} \cup \{\bot \leftarrow \texttt{s} \mid \texttt{s} \in (\mathcal{T} \cap H(P)) \setminus S\} \,. \tag{3}$$

That is, external atoms in S are added as facts to the program, whereas for any theory atom $\texttt{s}$ occurring as a rule head but not in S, we add the constraint $\bot \leftarrow \texttt{s}$. For illustration, consider the `clingcon`-program $P_{(4/5)}$:

$$\texttt{a} \leftarrow \texttt{\&sum}\{x; y\} = 4 \tag{4}$$

$$\texttt{\&sum}\{y; z\} = 2 \leftarrow \texttt{a} \tag{5}$$

This program contains two theory atoms: $\texttt{\&sum}\{x; y\} = 4$ and $\texttt{\&sum}\{y; z\} = 2$. We refer to them by $\texttt{s}_1$ and $\texttt{s}_2$, respectively. Theory atom $\texttt{s}_1$ is external since it occurs in the body of a rule, whereas we decide that $\texttt{s}_2$ is founded.

As explained above, the absence of any external atom, $\texttt{s} \in \mathcal{T}$, in our candidate set, $\texttt{s} \notin S$, imposes its complementary constraint to be satisfied. This is the so-called "strict" behavior. For this reason, we assume that, for any external atom, there always exists another external atom that is associated with that complementary constraint, even if this

complementary atom does not occur in the program. We formalize this intuition in Section 3 below. In our example, since $\mathtt{s}_1$ is external, we require a second, implicit external atom $\&\mathtt{sum}\{x;y\}\ \mathtt{!=}4$, which we name $\mathtt{s}_3$, for short, and which stands for the complement of $\mathtt{s}_1$. As a result, our example deals with two external atoms $\mathcal{E} = \{\mathtt{s}_1, \mathtt{s}_3\}$ that are complementary to each other and one defined atom $\mathcal{F} = \{\mathtt{s}_2\}$. Another consequence of the strict behavior is that any candidate $\mathfrak{T}$-solution S must contain either $\mathtt{s}_1$ or $\mathtt{s}_3$, because the absence of one of them implies forcing the other, whereas the two of them cannot be selected at the same time because they are inconsistent altogether.

To obtain the stable models of $P_{(4/5)}$, we must decide first which candidate set S of theory atoms are actually $\mathfrak{T}$-solutions. Although the definition of $\mathfrak{T}$-stable models starts by considering each $\mathfrak{T}$-solution S and builds an associated program (3) afterward, in practice, we can just start from any possible candidate set of theory atoms S to form program (3) and just check if they form a $\mathfrak{T}$-solution once we obtain the stable models X for that program. In our example, we are free to either add the external atom $\mathtt{s}_1$ or its complement $\mathtt{s}_3$ as a fact. If we decide to leave $\mathtt{s}_2 \notin S$, we must add the constraint $\leftarrow \mathtt{s}_2$. This amounts to two possibilities: $X_1 = \{\mathtt{s}_1, \mathtt{s}_2, \mathtt{a}\}$ and $X_2 = \{\mathtt{s}_3\}$. If $\mathfrak{T}$ is the theory of linear equations, it is not difficult to see that $S_1 = X_1 \cap \mathcal{T} = \{\mathtt{s}_1, \mathtt{s}_2\}$ is a $\mathfrak{T}$-solution because the associated constraints $x + y = 4$ and $y + z = 2$ are satisfiable: as a justification take, for instance, the variable assignment $\{x \mapsto 2, y \mapsto 2, z \mapsto 0\}$. When we use a variable assignment in this way, we say that it constitutes a *witness* of satisfiability for the set of constraints. In the case of $S_2 = X_2 \cap \mathcal{T} = \{\mathtt{s}_3\}$, we must simply satisfy the constraint $x + y \neq 4$ and we can use, for instance, the witness $\{x \mapsto 0, y \mapsto 0, z \mapsto n\}$ with n being any integer (note that the value of z is irrelevant for satisfying this constraint).

3. Transformation-Based Semantics Revisited

Now, we provide a characterization of the theory reasoning framework of `clingo` that allows for formal elaborations of $\mathcal{T}$-logic programs. To this end, we introduce the concept of an *abstract theory* $\mathfrak{T}$, which is our key instrument for establishing more fine-grained formal foundations. With it, we revisit the transformation-based approach and give a formal account of $\mathfrak{T}$-solutions.

3.1. Abstract Theories

An *abstract theory* $\mathfrak{T}$ is a triple $\langle \mathcal{T}, \mathit{Sat}, \widehat{\cdot} \rangle$ where $\mathcal{T}$ is the set of *theory atoms* we can handle in the theory, $\mathit{Sat} \subseteq 2^{\mathcal{T}}$ is the set of $\mathfrak{T}$-*satisfiable* sets of theory atoms, and $\widehat{\cdot} : \mathcal{T} \to \mathcal{T}$ is a function mapping theory atoms to their *complement* such that $\widehat{\widehat{\mathtt{s}}} = \mathtt{s}$ for any $\mathtt{s} \in \mathcal{T}$. We define $\widehat{S} = \{\widehat{\mathtt{s}} \mid \mathtt{s} \in S\}$ for any set $S \subseteq \mathcal{T}$. Note that the set *Sat* acts as an oracle whose rationality is beyond our reach: to see if some set S of theory atoms is satisfiable, we may only check $S \in \mathit{Sat}$.

Despite the limited structure of such theories, some simple properties can be formulated: a set $S \subseteq \mathcal{T}$ of theory atoms is

- *inconsistent*, if both $\mathtt{s} \in S$ and $\widehat{\mathtt{s}} \in S$ for some atom $\mathtt{s}$, (*consistent* otherwise)
- $\mathcal{Z}$*-complete* for some subset $\mathcal{Z} \subseteq \mathcal{T}$, if for all $\mathtt{s} \in \mathcal{Z}$, either $\mathtt{s} \in S$ or $\widehat{\mathtt{s}} \in S$, and
- *closed*, if $\mathtt{s} \in S$ implies $\widehat{\mathtt{s}} \in S$.

The definition of completeness is relative to some subset $\mathcal{Z} \subseteq \mathcal{T}$ of theory atoms, but when $\mathcal{Z} = \mathcal{T}$, we simply say that S is *complete*. An abstract theory $\mathfrak{T} = \langle \mathcal{T}, \mathit{Sat}, \widehat{\cdot} \rangle$ is *consistent* or *complete* if all its $\mathfrak{T}$-satisfiable sets $S \in \mathit{Sat}$ are *consistent* or *complete*, respectively.

Note that any non-empty closed set $S \neq \emptyset$ is inconsistent but can be $\mathfrak{T}$-satisfiable (e.g., for paraconsistent theories). We mostly use closed sets to describe subtypes of theory atoms rather than satisfiable solutions.

As an example, consider an abstract theory $\mathfrak{L} = \langle \mathcal{T}, \mathit{Sat}, \widehat{\cdot} \rangle$ of linear equations that can be used to capture `clingcon`programs, where

- $\mathcal{T}$ is the set of all expressions of form (1),

- *Sat* is the set of all subsets $S \subseteq \mathcal{T}$ of expressions of form (1) for which there exists an assignment of integer values to their variables that satisfies all the linear equations in S with their usual meaning, and
- the complement function is defined such as $\widehat{\texttt{\&sum}\{\dots\} \prec c}$ is $\texttt{\&sum}\{\dots\} \mathbin{\widehat{\prec}} c$ with $\widehat{\prec}$ defined according to the following table:

$\prec$	<=	=	!=	<	>	>=
$\widehat{\prec}$	>	!=	=	>=	<=	<

Note that the set of theory atoms in $\mathfrak{L}$ is closed.

Similarly, to cover `clingo`[DL]-programs, we introduce abstract theory $\mathfrak{D}$ capturing *difference constraints* over integers, which is a subset of the already seen abstract theory $\mathfrak{L}$ where theory atoms have the fixed form $\texttt{\&sum}\{1 * x; (-1) * y\} <= k$ but are rewritten instead as:

$$\texttt{\&diff}\{x - y\} <= k \tag{6}$$

where x and y are integer variables and $k \in \mathbb{Z}$.

Even though this definition of abstract theories is limited to a minimum set of requirements (no particular syntax, a set *Sat* used for satisfiability testing and a complement function), we can already define an abstract form of entailment as follows.

Definition 1 (Abstract Entailment)**.** *Let $\mathfrak{T} = \langle \mathcal{T}, Sat, \widehat{\cdot} \rangle$ be an abstract theory. We say that a set $S \subseteq \mathcal{T}$ of theory atoms entails a theory atom $s \in \mathcal{T}$, written $S \mid\!\approx_{\mathfrak{T}} s$, when $S \cup \{\widehat{s}\} \notin Sat$.*

In other words, $S \mid\!\approx_{\mathfrak{T}} s$ when adding the complement of s to S becomes inconsistent. When $S = \{s'\}$ is a singleton, we normally remove the set braces and just write $s' \mid\!\approx_{\mathfrak{T}} s$ instead of $\{s'\} \mid\!\approx_{\mathfrak{T}} s$. For example, it is easy to see that

$$\texttt{\&sum}\{x; y\} >= 1 \quad \mid\!\approx_{\mathfrak{L}} \quad \texttt{\&sum}\{x; y\} >= 0$$

because the pair of atoms $\texttt{\&sum}\{x; y\} >= 1$ and $\texttt{\&sum}\{x; y\} < 0$ (the complement of $\texttt{\&sum}\{x; y\} >= 0$) are in no satisfiable set in *Sat* under the usual meaning of linear equations.

3.2. Stable Models of Logic Programs with Abstract Theories

With a firm definition of what constitutes a theory $\mathfrak{T}$, we can now make precise the definition of a $\mathfrak{T}$-solution and $\mathfrak{T}$-stable model of a $\mathcal{T}$-logic program given in (3). For clarity, we refine those definitions by further specifying the subset of external theory atoms in $\mathcal{E}$. Moreover, given any set S of theory atoms, we define its *(complemented) completion* with respect to external atoms $\mathcal{E}$, denoted by $Comp_{\mathcal{E}}(S)$, as the following superset of S:

$$Comp_{\mathcal{E}}(S) \stackrel{\text{def}}{=} S \cup \widehat{(\mathcal{E} \setminus S)}$$

In other words, we add the complement atom $\widehat{\mathtt{s}}$ for every external atom $\mathtt{s}$ that does not occur explicitly in S. Note that S is $\mathcal{E}$-complete iff $Comp_{\mathcal{E}}(S) = S$. As an example, consider an abstract theory from $\mathfrak{L}$ with the theory atoms $\mathcal{T} = \{\mathtt{s}_1, \mathtt{s}_2, \mathtt{s}_3, \mathtt{s}_4\}$ given above with

$$\mathtt{s}_1 = (\texttt{\&sum}\{x; y\} = 4) \qquad \mathtt{s}_3 = (\texttt{\&sum}\{x; y\}\ \texttt{!=}\ 4)$$
$$\mathtt{s}_2 = (\texttt{\&sum}\{y; z\} = 2) \qquad \mathtt{s}_4 = (\texttt{\&sum}\{y; z\}\ \texttt{!=}\ 2)$$

Suppose we only have one external atom $\mathcal{E} = \{\mathtt{s}_1\}$ and assume that the complements $\widehat{\mathtt{s}_1} = \mathtt{s}_3$ and $\widehat{\mathtt{s}_2} = \mathtt{s}_4$ are defined as expected. Then, set $S = \{\mathtt{s}_2\}$ is $\mathcal{E}$-incomplete because neither $\mathtt{s}_1 \in S$ nor $\widehat{\mathtt{s}_1} = \mathtt{s}_3 \in S$. The completion of S corresponds to

$$Comp_{\mathcal{E}}(S) = S \cup \widehat{(\mathcal{E} \setminus S)} = \{\mathtt{s}_2, \widehat{\mathtt{s}_1}\} = \{\mathtt{s}_2, \mathtt{s}_3\}$$

so we eventually add $\mathtt{s}_3$. On the other hand, set $S = \{\mathtt{s}_1, \mathtt{s}_2\}$ is $\mathcal{E}$-complete because we do have information about the external atom $\mathtt{s}_1$, and so, $\widehat{\mathcal{E} \setminus S}$ does not provide any additional information.

Definition 2 ($\langle \mathfrak{T}, \mathcal{E} \rangle$-solution)**.** *Given an abstract theory $\mathfrak{T} = \langle \mathcal{T}, Sat, \widehat{\cdot} \rangle$ and a set $\mathcal{E} \subseteq \mathcal{T}$ of external theory atoms, we define $S \in Sat$ as a $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution, if $Comp_{\mathcal{E}}(S) \in Sat$, that is, the completion of S is $\mathfrak{T}$-satisfiable.*

The next result shows that $\mathcal{E}$-complete solutions suffice when considering the existence of a solution (The proofs for all propositions can be found in Appendix B):

Proposition 1. *Let $\mathfrak{T} = \langle \mathcal{T}, Sat, \widehat{\cdot} \rangle$ be an abstract theory and $\mathcal{E} \subseteq \mathcal{T}$ be a set of external atoms. For any $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution $S \subseteq \mathcal{T}$, we have:*

1. *$Comp_{\mathcal{E}}(S)$ is $\mathcal{E}$-complete and is also a $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution.*
2. *If S is $\mathcal{E}$-complete, then S is $\mathfrak{T}$-satisfiable.*

Moreover, we can identify a quite general family of consistent abstract theories for which *all* their solutions are always $\mathcal{E}$-complete. Consider atoms $\mathtt{s}_1 = (\,\&\mathtt{sum}\{x; y\} = 4\,)$ and its complement $\widehat{\mathtt{s}_1} = \mathtt{s}_3 = (\,\&\mathtt{sum}\{x; y\}\ \mathtt{!=} 4\,)$ from the above example. Assume that we have some set $S \subseteq \mathcal{T}$ that contains none of the two. If the two atoms are considered external, that is, $\{\mathtt{s}_1, \mathtt{s}_3\} \subseteq \mathcal{E}$, then $Comp_{\mathcal{E}}(S)$ must include their complements, which are the two atoms themselves again, since $\widehat{\widehat{\mathtt{s}_1}} = \mathtt{s}_1$. However, then, $Comp_{\mathcal{E}}(S)$ becomes inconsistent, and so it cannot be a $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution for any consistent abstract theory. As a result, if the theory is consistent and $\mathcal{E}$ is closed, that is, it contains all the complements of its elements, then all solutions must be $\mathcal{E}$-complete:

Proposition 2. *Let $\mathfrak{T} = \langle \mathcal{T}, Sat, \widehat{\cdot} \rangle$ be a consistent abstract theory with a* closed *set of external atoms $\mathcal{E} \subseteq \mathcal{T}$. Then, all $\langle \mathfrak{T}, \mathcal{E} \rangle$-solutions $S \subseteq \mathcal{T}$ are $\mathcal{E}$-complete.*

In analogy to Section 2, a set $X \subseteq \mathcal{A} \cup \mathcal{T}$ of atoms is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-*stable model* of a $\mathcal{T}$-logic program P, if there is some $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution S such that X is a stable model of the program in (3). As an example, take again the `clingcon`-program consisting of rules (4) and (5) with abstract theory $\mathfrak{L}$, the already seen theory atoms $\mathcal{T} = \{\mathtt{s}_1, \mathtt{s}_2, \mathtt{s}_3, \mathtt{s}_4\}$, and the closed subset of external atoms $\mathcal{E} = \{\mathtt{s}_1, \mathtt{s}_3\}$. This program has two $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable models:

$$\begin{aligned} X_1 &= \{\mathtt{a},\ \mathtt{s}_1,\ \mathtt{s}_2\} = \{\mathtt{a},\ \&\mathtt{sum}\{x; y\} = 4,\ \&\mathtt{sum}\{y; z\} = 2\} \text{ and} \\ X_2 &= \{\mathtt{s}_3\} = \{\&\mathtt{sum}\{x; y\}\ \mathtt{!=} 4\} \end{aligned} \tag{7}$$

To verify X_1, take the $\langle \mathfrak{L}, \mathcal{E} \rangle$-solution $\{\mathtt{s}_1, \mathtt{s}_2\} = \{\&\mathtt{sum}\{x; y\} = 4,\ \&\mathtt{sum}\{y; z\} = 2\}$ and the resulting program transformation

$$\mathtt{a} \leftarrow \&\mathtt{sum}\{x; y\} = 4 \tag{8}$$

$$\&\mathtt{sum}\{y; z\} = 2 \leftarrow \mathtt{a} \tag{9}$$

$$\&\mathtt{sum}\{x; y\} = 4 \leftarrow \tag{10}$$

We see that X_1 is a stable model of the logic program and, as such, also a $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable model. Similarly, with $\langle \mathfrak{L}, \mathcal{E} \rangle$-solution $\{\mathtt{s}_3\} = \{\&\mathtt{sum}\{x; y\}\ \mathtt{!=} 4\}$, we get program

$$\mathtt{a} \leftarrow \&\mathtt{sum}\{x; y\} = 4$$

$$\&\mathtt{sum}\{y; z\} = 2 \leftarrow \mathtt{a} \tag{11}$$

$$\&\mathtt{sum}\{x; y\}\ \mathtt{!=} 4 \leftarrow$$

$$\leftarrow \&\mathtt{sum}\{y; z\} = 2 \tag{12}$$

Again, we have X_2 as a stable model, confirming it as a $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable model. Notice that, in this case, the $\langle \mathfrak{L}, \mathcal{E} \rangle$-solution $\{\mathtt{s}_3\}$ corresponding to X_2 is not unique. For instance, we could also have the $\langle \mathfrak{L}, \mathcal{E} \rangle$-solution $\{\mathtt{s}_2, \mathtt{s}_3\} = \{\&\mathtt{sum}\{y;z\} = 2,\ \&\mathtt{sum}\{x;y\}\ \mathtt{!=} 4\}$, which results in program (11).

This program also has X_2 as a stable model. As a third alternative, we could have the $\langle \mathfrak{L}, \mathcal{E} \rangle$-solution $\{\mathtt{s}_3, \mathtt{s}_4\} = \{\&\mathtt{sum}\{x;y\}\ \mathtt{!=} 4,\ \&\mathtt{sum}\{y;z\}\ \mathtt{!=} 2\}$. This leads to the same program as $\langle \mathfrak{L}, \mathcal{E} \rangle$-solution $\{\mathtt{s}_3\}$ and, therefore, it is also a valid witness for stable model X_2. This example illustrates that, for founded theory atoms not derived by any rule, the represented linear equation, its complement, or none of the two can be in the $\langle \mathfrak{L}, \mathcal{E} \rangle$-solution used as a witness for a particular $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable model.

As mentioned in Section 2, we may freely shift any external theory atom in the head to the body (after negating it). This is formalized in the following proposition, whose proof can be found in Appendix B.

Proposition 3. *Let P be a $\mathcal{T}$-logic program and $\mathfrak{T} = \langle \mathcal{T}, Sat, \widehat{\cdot} \rangle$ be an abstract theory. Let $\mathcal{E} \subseteq \mathcal{T}$ be a set of external atoms, and let $\mathtt{s} \in \mathcal{E}$ be one of those external atoms. Then, programs $P \cup \{\mathtt{s} \leftarrow B\}$ and $P \cup \{\bot \leftarrow \neg\mathtt{s}, B\}$ have the same $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable models for any list B of literals.*

4. Structured Theories and Answer Sets

The notion of $\langle \mathfrak{T}, \mathcal{E} \rangle$-*stable model* introduced in Section 3 precisely characterizes the semantics of `clingo` 5 [16]. The strength of this semantic is two-fold. First, it provides an effective way to develop algorithms that connect answer set solvers with abstract theories. Second, it is very general as it puts very few requirements on the abstract theories it can be applied to. This applies to constraint answer set solvers based on `clingo`, such as the mentioned `clingo`[DL], `clingcon` and `clingo`[LP].

The price to pay for this generality is that some important features of these constraint answer set solvers cannot be captured by these two concepts alone. In many practical applications of hybrid systems, we are interested in the assignment to some variables rather than the theory atoms that are satisfied. For instance, we discussed above that a program consisting of rules (4) and (5) has two stable models, $X_1 = \{\mathtt{s}_1, \mathtt{s}_2, \mathtt{a}\}$ and $X_2 = \{\mathtt{s}_3\}$. However, if we use this program as an input to `clingcon`, typing the command:

```
echo "a :- &sum{ x; y } = 4.  \
      &sum{ y; z } = 2 :- a." | clingcon --heuristic=Berkmin
```

we obtain the output

```
Answer: 1
a
Assignment:
x=4 y=0 z=2

Answer: 2
a
Assignment:
x=3 y=1 z=1
...
```

Both answers shown above correspond to stable model X_1, but only the regular atom a is printed. In addition, each answer is accompanied by an assignment of values to the variables. This assignment represents a *witness* for the satisfiability of the pair of constraints corresponding to theory atoms $\mathtt{s}_1$ and $\mathtt{s}_2$ in the stable model. In fact, there are infinitely many different witnesses for stable model X_1, and `clingcon` enumerates it one by one, if asked for more solutions.

In this section, we define an *answer set* as each of the individual answers, such as the ones shown above (that is, regular atoms and variable assignments), generated from the stable models of the program. To make the idea of a witness more precise, we first need to

formalize the use of *variables* in theory atoms, something that has not been required so far. To this aim, we next introduce a quite general class of abstract theories called *compositional*.

4.1. Structured and Compositional Theories

We have seen that the set of satisfiable sentences *Sat* of an abstract theory $\mathfrak{T}$ is treated as an oracle or a blackbox: knowing its inner structure or behavior (if it has so) is not required at all for defining $\mathfrak{T}$-stable models. There may be cases where deciding whether some set S of theory atoms is satisfiable ($S \in Sat$) depends on the whole set together and cannot be decided in a compositional way, that is, in terms of some constructive condition about the satisfiability of the elements $s \in S$. For instance, most abstract theories based on non-monotonic formalisms are non-compositional. Example A1 in Appendix A illustrates the case in which we use *Equilibrium Logic* [17] as the external theory: since equilibrium models are a kind of minimal models of some theories, their existence depends on the whole set of formulas in that theory. However, in most cases, the set *Sat* is defined in a compositional way, the following conditions imposed on the structure and elements used inside the theory atoms, as happens with linear equations containing integer variables and arithmetic operations. Trying to keep as much generality as possible, we introduce the use of variables inside theory atoms but rather than describing the semantics of operations among them, we just require the existence of some denotational semantics. Such theory-specific structures allow us to establish properties of abstract theories and ultimately characterize their integration into ASP.

Definition 3 (Structure). *Given an abstract theory* $\mathfrak{T} = \langle \mathcal{T}, Sat, \widehat{\cdot} \rangle$*, we define a* structure *as a tuple* $(\mathcal{X}, \mathcal{D}, vars, [\![\cdot]\!])$*, where*

1. $\mathcal{X}$ *is a set of variables,*
2. $\mathcal{D}$ *is a set of domain elements,*
3. $vars : \mathcal{T} \to 2^{\mathcal{X}}$ *is a function giving the set of variables contained in a theory atom such that* $vars(\mathtt{s}) = vars(\widehat{\mathtt{s}})$ *for all theory atoms* $\mathtt{s} \in \mathcal{T}$*,*
4. $\mathcal{V} = \{v \mid v : \mathcal{X} \to \mathcal{D}\}$ *is the set of all valuations over* $\mathcal{X}$ *and* $\mathcal{D}$*, and*
5. $[\![\cdot]\!] : \mathcal{T} \to 2^{\mathcal{V}}$ *is a function mapping theory atoms to sets of valuations such that*

$$v \in [\![\mathtt{s}]\!] \text{ iff } w \in [\![\mathtt{s}]\!]$$

for all theory atoms $\mathtt{s} \in \mathcal{T}$ *and every pair of valuations* v, w *agreeing on the value of all variables* $vars(\mathtt{s})$ *occurring in* $\mathtt{s}$*.*

Whenever an abstract theory $\mathfrak{T}$ is associated with such a structure, we call it *structured* (rather than abstract).

In what follows, we represent a valuation v as a set of pairs $\{x_1 = d_1, \ldots, x_n = d_n\}$ where $x_i \in \mathcal{X}$ and $d_i \in \mathcal{D}$ for all $i = 1, \ldots, n$. Since the valuation stands for a function, we do not allow $(x = d) \in v$ and $(x = d') \in v$ for two different values $d \neq d'$. On the other hand, we are sometimes interested in *partial valuations* $v : \mathcal{X} \nrightarrow \mathcal{D}$ that, when represented as sets of pairs happen to be incomplete. That is, there may be variables $x \in \mathcal{X}$ for which no pair $(x = d)$ is included in v (and so the variable is undefined). Given any valuation v, we define its domain as $domain(v) \stackrel{\text{def}}{=} \{x \in \mathcal{X} \mid \exists d \in \mathcal{D}, (x = d) \in v\}$, that is, the set of variables that are defined in v. Total valuations satisfy $domain(v) = \mathcal{X}$.

Given a set S of theory atoms, we define its denotation as $[\![S]\!] \stackrel{\text{def}}{=} \bigcap_{\mathtt{s} \in S} [\![\mathtt{s}]\!]$. We say that a theory $\mathfrak{T} = \langle \mathcal{T}, Sat, \widehat{\cdot} \rangle$ structured by $(\mathcal{X}, \mathcal{D}, vars, [\![\cdot]\!])$ is *compositional* if $Sat = \{S \subseteq \mathcal{T} \mid [\![S]\!] \neq \emptyset\}$. Therefore, a set S is $\mathfrak{T}$-satisfiable iff its denotation is not empty. As an example, let us associate the theory of linear equations $\mathfrak{L}$ with the structure $(\mathcal{X}_{\mathfrak{L}}, \mathcal{D}_{\mathfrak{L}}, vars_{\mathfrak{L}}, [\![\cdot]\!]_{\mathfrak{L}})$, where

- $\mathcal{X}_{\mathfrak{L}}$ is an infinite set of integer variables,
- $\mathcal{D}_{\mathfrak{L}} = \mathbb{Z}$,
- $vars_{\mathfrak{L}}(\&\mathtt{sum}\{k_1 * x_1; \ldots; k_n * x_n\} \prec k_0) = \{x_1, \cdots, x_n\}$, and

- $[\![\, \texttt{\&sum}\{k_1 * x_1; \ldots; k_n * x_n\} \prec k_0 \,]\!]_{\mathfrak{L}} =$

$$\{v \in \mathcal{V}_{\mathfrak{L}} \mid \{k_1, v(x_1), \ldots k_n, v(x_n)\} \subseteq \mathbb{Z} \text{ and } \textstyle\sum_{1 \leq i \leq n} k_i * v(x_i) \prec k_0\}.$$

With $\mathfrak{L}$, a set S of theory atoms capturing linear equations is $\mathfrak{L}$-satisfiable whenever $[\![\, S \,]\!]_{\mathfrak{L}}$ is non-empty. Once $\mathfrak{L}$ is structured in this way, we can establish the following properties.

Proposition 4. *Theory $\mathfrak{L}$ structured by $(\mathcal{X}_{\mathfrak{L}}, \mathcal{D}_{\mathfrak{L}}, vars_{\mathfrak{L}}, [\![\cdot]\!]_{\mathfrak{L}})$ is compositional and consistent.*

If a theory $\mathfrak{T}$ is compositional, we can define an associated entailment relation as follows.

Definition 4 (Compositional entailment)**.** *Let $\mathfrak{T} = \langle \mathcal{T}, Sat, \widehat{\cdot} \rangle$ be a compositional theory structured by $(\mathcal{X}, \mathcal{D}, vars, [\![\cdot]\!])$. We say that a set of theory atoms $S \subseteq \mathcal{T}$* (compositionally) entails *a theory atom $\mathtt{s} \in \mathcal{T}$, written $S \models \mathtt{s}$, when $[\![\, S \,]\!] \subseteq [\![\, \mathtt{s} \,]\!]$.*

Again, for a singleton $S = \{\mathtt{s}'\}$, we omit braces and simply write $\mathtt{s}' \models \mathtt{s}$.

Back to the example we used to illustrate abstract entailment $\mid\!\approx_{\mathfrak{T}}$, we can easily see now that

$$\texttt{\&sum}\{x; y\}\texttt{>=1} \quad \models_{\mathfrak{L}} \quad \texttt{\&sum}\{x; y\}\texttt{>=0}$$

since any integer valuation v such that $v(x) + v(y) \geq 1$ must satisfy $v(x) + v(y) \geq 0$ as well. Unlike $\mid\!\approx_{\mathfrak{T}}$, this compositional entailment $\models_{\mathfrak{T}}$ is always monotonic; that is, $S \models \mathtt{s}$ implies $S \cup S' \models \mathtt{s}$. This is because, when $[\![\, S \,]\!] \subseteq [\![\, \mathtt{s} \,]\!]$, we obtain $[\![\, S \cup S' \,]\!] = [\![\, S \,]\!] \cap [\![\, S' \,]\!] \subseteq [\![\, \mathtt{s} \,]\!]$ too. As we explained before, in general, most non-monotonic formalisms are non-compositional in the sense that their satisfiability condition $S \in Sat$ usually depends on the whole set S of theory atoms and cannot be described in terms of the individual satisfiability of each atom $\mathtt{s} \in S$. Compositional theories are interesting from an implementation point of view, since they allow for handling partial assignments that can be extended monotonically. The following result proves that, for a given consistent, compositional abstract theory $\mathfrak{T}$, compositional entailment implies abstract entailment.

Proposition 5. *For any compositional and consistent theory $\mathfrak{T}$, $S \models_{\mathfrak{T}} s$ implies $S \mid\!\approx_{\mathfrak{T}} s$.*

In general, however, the opposite may not hold; that is, abstract entailment is weaker than compositional entailment. Example A3 in Appendix A illustrates a consistent, compositional theory where we may have $S \mid\!\approx_{\mathfrak{T}} s$ but not $S \models_{\mathfrak{T}} s$.

Some compositional theories have a complement $\widehat{\mathtt{s}}$ whose denotation is precisely the set complement of $[\![\, \mathtt{s} \,]\!]$, that is, $[\![\, \widehat{\mathtt{s}} \,]\!] = \mathcal{V} \setminus [\![\, \mathtt{s} \,]\!]$. In this case, we say that the complement is *absolute*. This is, in fact, the case of linear constraints $\mathfrak{L}$ complement: For instance, $[\![\, \widehat{\texttt{\&sum}\{x; y\}\texttt{=4}} \,]\!] = [\![\, \texttt{\&sum}\{x; y\}\ \texttt{!=4} \,]\!] = \mathcal{V}_{\mathfrak{L}} \setminus [\![\, \texttt{\&sum}\{x; y\}\texttt{=4} \,]\!]$. Examples of non-absolute complements may arise when the abstract theory is, for instance, a multi-valued logic (See Examples A2 and A4 with Kleene's three-valued logic and Fuzzy Logic based on Łukasiewizc's operators in Appendix A, respectively) where we may have valuations that are not models of a formula nor its complement (assuming we use negation for that role). Having an absolute complement directly implies that the theory is consistent. This is because the denotations $[\![\, \mathtt{s} \,]\!]$ and $[\![\, \widehat{\mathtt{s}} \,]\!]$ are disjoint, and so any set including $\{\mathtt{s}, \widehat{\mathtt{s}}\}$ is $\mathfrak{T}$-unsatisfiable.

Proposition 6. *For any compositional theory $\mathfrak{T} = \langle \mathcal{T}, Sat, \widehat{\cdot} \rangle$ with an absolute complement $\widehat{\cdot}$, any $S \subseteq \mathcal{T}$ and $\mathtt{s} \in \mathcal{T}$, we have that $S \models \mathtt{s}$ iff $(S \cup \{\widehat{\mathtt{s}}\}) \notin Sat$.*

4.2. Answer Sets of Logic Programs with Compositional Theories

We can formalize the idea of an answer set as a pair containing a set of regular atoms plus an (possibly partial) assignment of values to the theory variables. The definition presented here is not tailored to any particular system but only requires that the structured theory is *consistent*, *compositional*, and has an *absolute complement*. These are three properties that the theories of the three mentioned systems share, and that may very well cover future hybrid systems based on `clingo` 5.

To formally establish the concept of answer sets, let us introduce some notations. Given any valuation $v \in \mathcal{V}$ and a subset of variables $\Sigma \subseteq \mathcal{X}$, the function $v|_\Sigma : \Sigma \to \mathcal{D}$ stands for the restriction of v to Σ. Accordingly, for any set of valuations $V \subseteq \mathcal{V}$ we define their restriction to Σ as $V|_\Sigma \stackrel{\text{def}}{=} \{v|_\Sigma \mid v \in V\}$. Given a structured theory $\mathfrak{T} = \langle \mathcal{T}, Sat, \widehat{\cdot} \rangle$, if $X \subseteq \mathcal{A} \cup \mathcal{T}$ is a set of atoms, we define

$$ans(X) \quad \stackrel{\text{def}}{=} \quad \{\, (Y, v|_\Sigma) \mid Y = X \cap \mathcal{A},\ \Sigma = vars(X \cap \mathcal{T}),\ v \in [\![\, X \cap \mathcal{T} \,]\!] \,\}$$

that is, $ans(X)$ collects all pairs (Y, v') where Y is fixed to the regular atoms in X, and v' varies among all valuations in $[\![\, X \cap \mathcal{T} \,]\!]$ from the theory atoms in X but restricted to the variables occurring in those theory atoms. Note that v is only defined for a subset of variables $\Sigma = vars(X \cap \mathcal{T}) \subseteq \mathcal{X}$. We can also understand v as a partial valuation $v : \mathcal{X} \nrightarrow \mathcal{D}$, where $domain(v) = \Sigma$, leaving all variables in $\mathcal{X} \setminus \Sigma$ undefined. In fact, the values of these variables are irrelevant due to Condition 5 in Definition 3 for a structured theory.

Definition 5 (Answer Set)**.** *If X is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model of some program P and (Y, v) belongs to $ans(X)$, then we say that (Y, v) is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-answer set of P.*

For instance, in our running example, the first answer set printed by `clingcon` corresponds to the $\langle \mathfrak{L}, \mathcal{E} \rangle$-*answer set* (Y_1, v_{11}) where $Y_1 = \{a\}$ is a set of regular atoms and v_{11} is the valuation $\{x = 4,\ y = 0,\ z = 2\}$. Recall that the theory atoms in X_1 are $\mathtt{s}_1$ and $\mathtt{s}_2$, and note that v_{11} satisfies both. Similarly, the second answer set printed by `clingcon` corresponds to the $\langle \mathfrak{L}, \mathcal{E} \rangle$-*answer set* (Y_1, v_{12}) where v_{12} is the valuation $\{x = 3,\ y = 1,\ z = 1\}$. As stated in (7), the other stable model of the program was $X_2 = \{\mathtt{s}_3\} = \{\texttt{\&sum}\{x; y\}\ \texttt{!=}\, 4\}$. The answer sets for this stable model have the form $(\emptyset, v)$ where v is assignments for the subset of variables $\{x, y\}$ occurring in $\mathtt{s}_3$, so that $v(x) + v(y) \neq 4$, such as, for instance, $\{x = 0, y = 0\}$ or $\{x = 3, y = 2\}$, to put a pair of examples. Note that the value of z in these cases is irrelevant.

Let us examine an example program that is accepted by `clingo`[DL] and its answer sets.

$$\begin{aligned} \mathtt{a} &\leftarrow \neg\neg \mathtt{a} \\ \texttt{\&diff}\{\mathtt{x} - \mathtt{y}\} \texttt{<=} -2 &\leftarrow \mathtt{a} \end{aligned} \tag{13}$$

The only theory atom in this example is founded, and the answer sets of `clingo`[DL] are the following:

```
Answer: 1

Answer: 2
dl(x,0) dl(y,2) a
```

Here, the first answer corresponds to the $\langle \mathfrak{D}, \emptyset \rangle$-stable model $\emptyset$ and the $\langle \mathfrak{D}, \emptyset \rangle$-answer set $(\emptyset, \emptyset)$, and the second answer corresponds to the $\langle \mathfrak{D}, \emptyset \rangle$-stable model

$$\{\mathtt{a}, \texttt{\&diff}\{\mathtt{x} - \mathtt{y}\} \texttt{<=} -2\}$$

and the $\langle \mathfrak{D}, \emptyset \rangle$-answer set $(\{a\}, \{x = 0, y = 2\})$. Note that the first answer set has an empty valuation since no theory atoms are true, and, therefore, no values for `x` or `y`

are actually required (any arbitrary assignment forms a $\mathfrak{D}$-solution). Further, there are obviously infinitely many answer sets in $ans(\{\texttt{a}, \texttt{\&diff}\{\texttt{x} - \texttt{y}\} <= -2\})$, but `clingo`[DL] selects a particular one for output, which we further discuss in Section 5.

As may be expected, given a consistent, compositional theory $\mathfrak{T}$, for every $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model, we always obtain at least one $\langle \mathfrak{T}, \mathcal{E} \rangle$-answer set.

Proposition 7. *Let $\mathfrak{T} = \langle \mathcal{T}, Sat, \hat{\cdot} \rangle$ be a compositional theory and let $\mathcal{E} \subseteq \mathcal{T}$ be a closed set of external atoms. If X is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model of some program P, then $ans(X) \neq \emptyset$.*

As we show above, $ans(X)$ is, in general, not a singleton, so the correspondence between the $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model and $\langle \mathfrak{T}, \mathcal{E} \rangle$-answer sets is one-to-many. However, if we focus on consistent, compositional theories that have an absolute complement, then we can reconstruct the $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model from any of its corresponding $\langle \mathfrak{T}, \mathcal{E} \rangle$-answer sets. In fact, we can establish a one-to-one correspondence between the $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model of a program and a kind of equivalence class of its $\langle \mathfrak{T}, \mathcal{E} \rangle$-answer sets. Each of the equivalence classes may contain many $\langle \mathfrak{T}, \mathcal{E} \rangle$-answer sets, but any of them has enough information to reconstruct the corresponding $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model. We say that an answer set (Y, v) *satisfies* an atom $b \in \mathcal{A} \cup \mathcal{T}$, written $(Y, v) \models b$, when:

- $b \in Y$ for regular atoms $b \in \mathcal{A}$;
- $v \in [\![b]\!]|_{domain(v)}$ for theory atoms $b \in \mathcal{T}$.

Note that the second condition restricts the denotation $[\![b]\!]$ to $domain(v)$. This is because v can be partial and if so, we just require that there exists some complete valuation in $[\![b]\!]$ that agrees with the values assigned by v to its defined variables $domain(v)$. For any negative literal $\neg b$, we say that $(Y, v) \models \neg b$ simply when $(Y, v) \not\models b$. If B is a rule body, we write $(Y, v) \models B$ to stand for $(Y, v) \models L$ for every literal L in B.

Formally, for a program P and an $\langle \mathfrak{T}, \mathcal{E} \rangle$-*answer set* (Y, v), we define

$$stb_P(Y, v) \;=\; Y \cup \{\, \mathtt{s} \in \mathcal{E} \mid v \in [\![\mathtt{s}]\!]|_{domain(v)} \,\} \cup \{\, \mathtt{s} \in \mathcal{F} \mid (B \rightarrow \mathtt{s}) \in P \text{ and } (Y, v) \models B \,\}$$

We also write $(Y_1, v_1) \sim (Y_2, v_2)$ if $stb_P(Y_1, v_1) = stb_P(Y_2, v_2)$ and say that (Y_1, v_1) and (Y_2, v_2) belong to the same equivalence class with respect to P.

Proposition 8. *Let $\mathfrak{T} = \langle \mathcal{T}, Sat, \hat{\cdot} \rangle$ be a consistent compositional theory with an absolute complement and let $\mathcal{E} \subseteq \mathcal{T}$ be a closed set of external atoms. Then, there is a one-to-one correspondence between the $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable models of some program P and the equivalence classes with respect to P of its $\langle \mathfrak{T}, \mathcal{E} \rangle$-answer sets. Furthermore, if (Y, v) is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-answer set, then $stb_P(Y, v)$ is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model of P and (Y, v) belongs to $ans(stb_P(Y, v))$.*

Despite this one-to-one correspondence, there is a crucial difference between $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable models and $\langle \mathfrak{T}, \mathcal{E} \rangle$-answer sets: it is possible for two different programs to have the same $\langle \mathfrak{T}, \mathcal{E} \rangle$-answer sets but different $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable models. This is a result of the use of the program to build the corresponding $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model in function $stb_P(\cdot)$. For instance, program

$$\texttt{a} \leftarrow \tag{14}$$

$$\bot \leftarrow \texttt{a}, \texttt{\&sum}\{x\}\texttt{!=4} \tag{15}$$

has a unique answer set (Y, v) with $Y = \{a\}$ and $v = \{x = 4\}$. This is also the unique answer set of the program obtained by extending the above program with rule

$$\texttt{\&sum}\{x\}\texttt{>=4} \leftarrow \texttt{a} \tag{16}$$

However, we can easily see that (14) and (15) have the unique stable model $\{\texttt{a}, \texttt{\&sum}\{x\} = 4\}$, whereas (14)–(16) has the unique stable model $\{\texttt{a}, \texttt{\&sum}\{x\} = 4, \texttt{\&sum}\{x\} >= 4\}$. This shows that stable models are more sensible to changes in the program than answer sets.

This difference becomes crucial, for instance, if we want to consider strong equivalence between programs.

5. Answer Set Solving Modulo Linear Equations

Now that we have provided a characterization of AMT with compositional theories, we finally show how our formalism can be used to capture the semantics of several `clingo` extensions with linear equations. At first, we use the structured theory $\mathfrak{L}$ of linear equations to describe the semantics of `clingcon`. Then, we introduce structured theories $\mathfrak{D}$ and $\mathfrak{R}$ to analogously capture `clingo`[DL] and `clingo`[LP]. This allows us to compare the systems and their features.

5.1. *clingcon*

In `clingcon`, all theory atoms are external, that is $\mathcal{E} = \mathcal{T}$, and may indistinctly occur in the head or in the body. As we already mentioned, one interesting feature of `clingcon` is that it does not only show the $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable models but also allows for enumerating all $\langle \mathfrak{L}, \mathcal{E} \rangle$-answer sets for those stable models.

To illustrate differences between $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable models and $\langle \mathfrak{L}, \mathcal{E} \rangle$-answer sets, consider Listing 1, which contains a program calculating the overall taxes in the language of `clingcon`. To interpret this program in our notation, we consider "`.`" to be the separation between the rules of the program, "`:-`" is the implication $\leftarrow$, "`not`" is the negation $\neg$, and the choice rule `eligible` is an abbreviation for the rule `eligible`$\leftarrow \neg\neg$`eligible`. The program contains one propositional variable, `eligible`, signifying whether one is eligible for a tax deduction, and three integer variables `tax`, `deduction`, and `overall` that are used in theory atoms and represent taxes before deduction, possible tax deduction, and the overall taxes to pay, respectively. Lines 1 and 2 describe the domain of the integer variables `tax` and `deduction` via theory atoms that state that `tax` is greater or equal to zero and smaller or equal to two, and `deduction` is greater or equal to zero and at most, the value of `tax`. Line 3 allows propositional atoms `eligible` to be freely chosen. Overall, Lines 1 to 3 describe the solution space with six overall combinations of values for the integer variables that are doubled by the possible truth values of `eligible` to a total of twelve combinations. In practice, these values would, of course, be subject to more complex computations. Lines 5 and 6 then calculate the value of the integer variable `overall`. The former, by subtracting the value of `deduction` from `tax` if `eligible`, is true, and the latter sets `overall` equal to `tax` if `eligible` is false.

Listing 1. `clingcon`-program calculating taxes

```
1 &sum{tax}>=0. &sum{tax}<=2.
2 &sum{deduction}>=0. &sum{deduction}<=tax.
3 {eligible}.
4
5 &sum{tax; -deduction} = overall :- eligible.
6 &sum{tax} = overall             :- not eligible.
```

In the following, the set $\mathcal{T}$ is comprised of the theory atoms occurring in Listing 1 and their complement. The `clingcon`-program has four $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable models:

```
{&sum{tax}>=0, &sum{tax}<=2,
 &sum{deduction}>=0, &sum{deduction}<=tax,
 &sum{tax;-deduction}=overall, &sum{tax}!=overall, eligible}

{&sum{tax}>=0, &sum{tax}<=2,
 &sum{deduction}>=0, &sum{deduction}<=tax,
 &sum{tax;-deduction}=overall, &sum{tax}=overall, eligible}
```

```
{&sum{tax}>=0, &sum{tax}<=2,
 &sum{deduction}>=0, &sum{deduction}<=tax,
 &sum{tax}=overall, &sum{tax;-deduction}!=overall}
```

and

```
{&sum{tax}>=0, &sum{tax}<=2,
 &sum{deduction}>=0, &sum{deduction}<=tax,
 &sum{tax}=overall, &sum{tax;-deduction}=overall}
```

We have four stable models since variable `deduction` can be zero, so whether we are eligible or not, we may have `tax`=`tax`−`deduction`=`overall`. These four $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable models yield the following twelve answer sets:

$$\begin{aligned}
(\emptyset, \{\texttt{tax} = 0, \texttt{deduction} = 0, \texttt{overall} = 0\} \\
(\emptyset, \{\texttt{tax} = 1, \texttt{deduction} = 0, \texttt{overall} = 1\} \\
(\emptyset, \{\texttt{tax} = 1, \texttt{deduction} = 1, \texttt{overall} = 1\} \\
(\emptyset, \{\texttt{tax} = 2, \texttt{deduction} = 0, \texttt{overall} = 2\} \\
(\emptyset, \{\texttt{tax} = 2, \texttt{deduction} = 1, \texttt{overall} = 2\} \\
(\emptyset, \{\texttt{tax} = 2, \texttt{deduction} = 2, \texttt{overall} = 2\} \\
(\{\texttt{eligible}\}, \{\texttt{tax} = 1, \texttt{deduction} = 1, \texttt{overall} = 0\} \\
(\{\texttt{eligible}\}, \{\texttt{tax} = 1, \texttt{deduction} = 0, \texttt{overall} = 1\} \\
(\{\texttt{eligible}\}, \{\texttt{tax} = 2, \texttt{deduction} = 1, \texttt{overall} = 1\} \\
(\{\texttt{eligible}\}, \{\texttt{tax} = 2, \texttt{deduction} = 0, \texttt{overall} = 2\} \\
(\{\texttt{eligible}\}, \{\texttt{tax} = 2, \texttt{deduction} = 2, \texttt{overall} = 0\} \\
(\{\texttt{eligible}\}, \{\texttt{tax} = 0, \texttt{deduction} = 0, \texttt{overall} = 0\}
\end{aligned} \tag{17}$$

In contrast to $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable models, $\langle \mathfrak{L}, \mathcal{E} \rangle$-answer sets provide the precise value associated to each variable. This is usually the information that the user is interested in.

Answer sets also provide a better object of study regarding program transformations. For instance, we may replace Line 5 in Listing 1 with the following two rules:

```
&sum{tax; -deduction} <= overall :- eligible.
&sum{tax; -deduction} >= overall :- eligible.
```

Intuitively, this replacement leads to an equivalent program. Indeed, this is the case when we look at $\langle \mathfrak{L}, \mathcal{E} \rangle$-answer sets (both programs have the same $\langle \mathfrak{L}, \mathcal{E} \rangle$-answer sets), but it is not the case when we look at $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable models. The program obtained after this replacement has the following four $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable models:

```
{...,&sum{tax;-deduction}<=overall, eligible
   &sum{tax;-deduction}>=overall, &sum{tax}!=overall}

{...,&sum{tax;-deduction}<=overall, eligible
   &sum{tax;-deduction}>=overall, &sum{tax}=overall}

{...,&sum{tax;-deduction}<overall,
   &sum{tax;-deduction}<=overall, &sum{tax}=overall}
```

and

```
{...,&sum{tax;-deduction}>=overall,
   &sum{tax;-deduction}<=overall, &sum{tax}=overall}
```

Note that the $\langle \mathfrak{L}, \mathcal{E} \rangle$-answer sets are independent of what theory atoms are syntactically used. The twelve answer sets are exactly the same in both programs. On the other hand, $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable models are different, depending on the particular theory atoms used. For

instance, the first $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable model of both programs captures the same information, but they are slightly different: the one corresponding to the second program is obtained from the first $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable model by replacing theory atom `&sum{tax;-deduction}=overall}` by atoms `&sum{tax;-deduction}<=overall}` and `&sum{tax;-deduction}>=overall}`. That is, the syntactic changes in the program are carried to the $\langle \mathfrak{L}, \mathcal{E} \rangle$-stable mode but not to the $\langle \mathfrak{L}, \mathcal{E} \rangle$-answer sets.

It is also worth mentioning that, since all theory atoms in `clingcon` are external, we can *always* apply Proposition 3 to constrain atoms in the head and shift them to the body. As an example, we can safely replace Line 5 in Listing 1 with the following constraint and retain the same stable models:

```
:- not &sum{tax; -deduction} = overall, eligible.
```

As we discuss below, this is not true for the remaining systems that combine external and founded atoms.

5.2. `clingo`[DL]

Let us now return to the `clingo`[DL] system that uses the theory $\mathfrak{D}$ in which theory atoms have the form `&diff`$\{x - y\}$ `<=` k, as introduced in (6). As with $\mathfrak{L}$, the abstract theory $\mathfrak{D}$ also has a complete complement operation. However, while in linear constraints, the complementary atom would be obtained by just switching the ordering relation from '<=' to '>', in the case of $\mathfrak{D}$, the resulting construct `&diff`$\{x - y\}$`>`k is not exactly in the format of a difference constraint, where we must always use the '<=' relation. Fortunately, as we deal with integer numbers, we can reformulate the complement simply as `&diff`$\{y - x\}$ `<=` $-k-1$. For example, the complement of `&diff`$\{x - y\}$ `<=` 5 would be `&diff`$\{y - x\}$ `<=` -6.

In `clingo`[DL], the set of external atoms $\mathcal{E}$ consists of all body atoms: atoms that do not occur in rule bodies are founded. One difference with respect to `clingcon` is that, in this case, we do have a one-to-one correspondence between the $\mathfrak{D}$-solutions and the $\langle \mathfrak{D}, \mathcal{E} \rangle$-stable models of a `clingo`[DL]-logic program P. Another important difference is that, for a given stable model, `clingo`[DL] does not enumerate all possible $\langle \mathfrak{D}, \mathcal{E} \rangle$-answer sets but only provides one of them. This answer set corresponds to the case where all integer variables are assigned a non-negative integer with the minimal possible value. That is, `clingo`[DL]-answer sets are a selection among the $\langle \mathfrak{D}, \mathcal{E} \rangle$-answer sets of the program. Formally, to characterize this minimal answer set, we start defining a partial order $\leq_{\texttt{dl}}$ between two $\langle \mathfrak{D}, \mathcal{E} \rangle$-answer sets (Y, v) and (Y', v') as follows:

$$(Y, v) \leq_{\texttt{dl}} (Y', v') \text{ iff}$$
$$Y = Y' \text{ and } \mathit{domain}(v) = \mathit{domain}(v') \text{ and } v(x) \leq v'(x) \text{ for all } x \in \mathit{domain}(v)$$

As usual, we write $(Y, v) <_{\texttt{dl}} (Y', v')$ when $(Y, v) \leq_{\texttt{dl}} (Y', v')$ and $v \neq v'$.

Definition 6 (`clingo`[DL]-answer set). *A $\langle \mathfrak{D}, \mathcal{E} \rangle$-answer set (Y, v) is called a `clingo`[DL]-answer set of P if (Y, v) is $<_{\texttt{dl}}$-minimal among the $\langle \mathfrak{D}, \mathcal{E} \rangle$-answer sets of P that do not assign a negative value to any variable.*

Note that if a set of difference constraints has a solution, then it also has a solution that assigns non-negative integer values to all variables. As a simple example consider a `clingo`[DL]-program that just consists of the fact

$$\texttt{\&diff}\{x - y\} \texttt{ <= } -2 \tag{18}$$

This program is imposing the condition $x + 2 \leq y$ and has infinitely many answer sets: we can choose any integer value for x and then fix y as any value greater than or equal to $x + 2$. However, since `clingo`[DL]-answer sets do not admit negative numbers, we must further pick $x \geq 0$, leaving the possible answer sets shown in Figure 1, where each pair of numbers (c, d) represents the valuation $x = c$, $y = d$. The arrows show the $<_{\texttt{dl}}$ relations

among them. In this case, there exists a unique `clingo`[DL]-answer set corresponding to the valuation $x = 0, y = 2$.

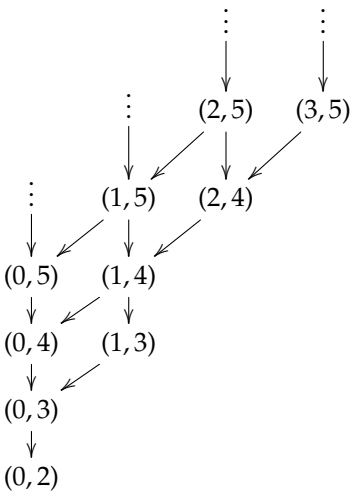

Figure 1. Answer sets for (18) without negative values.

5.3. `clingo`[LP]

A third AMT system covered by our formalization is `clingo`[LP]. In this case, the abstract theory $\mathfrak{R}$ is about linear equations over reals. It is identical to $\mathfrak{L}$ but structured by $(\mathcal{X}_\mathfrak{R}, \mathcal{D}_\mathfrak{R}, vars_\mathfrak{L}, [\![\cdot]\!]_\mathfrak{L})$ where $\mathcal{X}_\mathfrak{R}$ is an infinite set of real variables and the domain $\mathcal{D}_\mathfrak{R}$ is the set of real numbers $\mathbb{R}$. Previous versions of `clingo`[LP] treated either all theory atoms as external, $\mathcal{E} = \mathcal{T}$, or founded, $\mathcal{E} = \emptyset$. Since it imposes no restriction on the occurrence of linear equation atoms; the counter-intuitive behavior identified in [14] may emerge. The current version behaves like `clingo`[DL] and treats body atoms as external and head atoms as founded. Another commonality with `clingo`[DL] is a selection among the $\langle \mathfrak{R}, \mathcal{E} \rangle$-answer sets that are printed. In the case of `clingo`[LP], the selection is configurable by the user via a directive with a linear term:

$$\texttt{\&minimize}\{w_1 * x_1; ...; w_n * x_1\}$$

where $\{w_1, \ldots, w_n\} \subseteq \mathbb{R}$ and $\{x_1, \ldots, x_n\} \subseteq \mathcal{X}_\mathfrak{R}$. Then, the partial order $\leq_{\texttt{lp}}$ between two $\langle \mathfrak{R}, \mathcal{E} \rangle$-answer sets (Y, v) and (Y', v') is defined as:

$$\begin{aligned}
&(Y, v) \leq_{\texttt{lp}} (Y', v') \text{ iff} \\
&Y = Y' \text{ and } \text{domain}(v) = \text{domain}(v') \text{ and} \\
&w_1 * v(x_1) + \cdots + w_n * v(x_n) \leq w_1 * v'(x_1) + \cdots + w_n * v'(x_n)
\end{aligned}$$

Similarly, we write $(Y, v) <_{\texttt{lp}} (Y', v')$ when $(Y, v) \leq_{\texttt{lp}} (Y', v')$ and $v \neq v'$.

Definition 7 (`clingo`[LP]-answer set). *A $\langle \mathfrak{R}, \mathcal{E} \rangle$-answer set (Y, v) is called a* `clingo`[LP]*-answer set of P if (Y, v) is $<_{\texttt{lp}}$-minimal among the $\langle \mathfrak{R}, \mathcal{E} \rangle$-answer sets of P.*

If no minimize directive is provided, the linear term $x_1 + \cdots + x_n$ with $\{x_1, \ldots, x_n\} = \mathcal{X}_\mathfrak{R}$ is assumed by default. That is, the answer set with the smallest sum of variables is selected. For instance, given the program $\texttt{\&sum}\{x\} >= 0$, `clingo`[LP] only outputs the answer set where $\texttt{x} = 0$.

6. Discussion and Related Work

There exist different hybrid approaches in the literature that combine logical inference with non-Boolean external theories that, in most cases, are related to some form of numerical constraints. When we combine logic programs and external theories, we may find two

different philosophies that we can call *black box* versus *white box* orientations, respectively. In black box approaches, the intricacies of the external theory are mostly unknown from the logic program perspective: this is, in fact, the AMT approach followed in this paper. The white box style relies instead on extending the logic programming syntax to incorporate new structures and defining their corresponding semantics, treating constraints and their variables as first-class citizens inside the logic programming paradigm. Each orientation has its pros and cons. Black box approaches allow for a more flexible and homogeneous integration of new external theories, plus a simpler adaptation to new versions or features of the solvers. White box approaches are more convenient if we want to study formal properties of hybrid systems, such as equivalence among different representations or the soundness of syntactic transformations.

We proceed now to describe related work classified under these two orientations.

6.1. Black Box Orientation

Black box approaches follow somehow similar steps to what happened in SAT with the *SAT modulo theories* (SMT) [7] variant. SMT is defined as an extension to SAT where some atoms are replaced by formulas from some external theory expressible in first-order logic with equality, but usually without quantifiers (this includes numerical constraints but also covers data structures such as lists or strings). The key point in SMT is that the SAT solver does not really need to inspect the behavior of the external theory, so it can be treated as a black box. It is worth mentioning that, in fact, under the formalization proposed in the current paper, SMT can be seen as a particular case of AMT where all theory atoms are external (and we have no undefined variables) whereas, for each regular atom `a`, we include a choice rule such as (13), as we would do when emulating SAT inside ASP.

Following similar steps to SMT, the combination of constraint programming (CP) and ASP gave birth to *Constraint Answer Set Programming* (CASP). As explained in survey [18], CASP covers a family of approaches whose implementation may rely either on integration of CP inside a native ASP engine or on a translation of CASP programs into SMT or CP instead. From the semantic point of view, most of them treat theory atoms as external. Moreover, a choice for the constraint atom or its complement is usually included in the definitions. Moreover, many approaches only allow constraint atoms in bodies, e.g., in [5,19], or when they are used in rule heads [9], they just stand for syntactic sugar and can be shifted to bodies, as performed for `clingcon`-programs.

Next, we briefly list some of the existing CASP systems (see [18] for more examples) and their distinctive features with respect to the AMT framework proposed in this paper. Three examples of tools based on translation to non-ASP solvers are `dingo` [20], `mingo` [21], and `ezsmt` [22]. As happens with `clingo`[DL], system `dingo` [20] deals with ASP modulo difference logic but is actually based on a translation into SMT. In `dingo`, theory atoms are not external (they need to be derived), but they do not completely fit into our category of *defined* either, in the sense that SMT variables cannot be undefined. System `ezsmt` [22] is also based on a translation into SMT but accepts linear and non-linear constraints over mixed real and integer variables, whereas `mingo` [21] translates ASP modulo mixed-integer linear programming (MILP) to plain MILP. In these two systems, all theory atoms are external and may only appear in the body.

When solvers rely on the integration of constraints into native ASP instead, it is more common to find theory atoms that are defined rather than external. A prototypical example of this is `ezcsp` [19], where all theory atoms are defined and may only occur in the head, deviating from the standard CASP definition.

Another black box approach that extends ASP with external sources is *HEX programs* [23]. One of the main features of HEX programs is that, when consulting an external source, they allow higher-order parameters using the name of a logic program predicate as an argument for the external call. Another important feature is that HEX theory atoms allow for the invention of constants that are provided by the external source and were not originally present in the HEX program. This exceeds the ground-level definition used in the

current paper. Yet, if we focus on ground HEX programs, we can observe that theory atoms only occur in rule bodies, and they do not require justification, so they would vaguely correspond to our external atom category. However, there are two important differences. First, it is unclear what the "complement" of a HEX theory atom would be or even if such a complement would make sense. Second, the truth of a HEX theory atom depends on the current interpretation handled by the logic program. Generally speaking, the information flow in most AMT systems is from the abstract theory $\mathfrak{T}$ to the logic program but not in the other direction, that is, we usually have an abstract theory $\mathfrak{T}$ whose behavior is known beforehand and is independent of the current stable model X. In principle, our definitions do not prevent dealing with a parameterized theory $\mathfrak{T}_X$, but this option deserves to be explored in detail and will be part of future work.

One final and, perhaps, extreme example of a black box approach is the case of *Multi-Context Systems* (MCS) [24] where, rather than using ASP as basic formalism, we may actually deal with multiple contexts, each one with a potentially different logical formalism (classical logic, default logic, ASP, description logics, etc.). Like our description of abstract theories, the MCS logical formalisms are described at a very abstract level, only requiring the set of well-formed knowledge bases (the syntax), the set of belief sets, and the semantics in terms of a function that assigns the set of acceptable beliefs for each knowledge base. These contexts are connected altogether using so-called *bridge rules*, and the final semantics assigned to the system are defined in terms of *equilibria* reached among the belief sets from the contexts and the application of the bridge rules. The main difference with respect to MCS is that in AMT, we assume that our base formalism is ASP extended with other external theories and that, in principle, there is no particular interrelation among those theories or their shared beliefs.

6.2. White Box Orientation

In this second group of approaches, constraints are incorporated as first-class citizens inside the logic programming formalism, so their semantics are no longer external or abstract but are captured in an extended definition of answer sets. Examples of this orientation are a family of approaches [25–27] based on the incorporation of intensional or non-Herbrand functions in ASP. This line was initiated in [25] with the introduction of partial intensional functions in Quantified Equilibrium Logic, something that was later extended to sets and aggregates in [28]. This approach is very expressive as it provides full first-order semantics for arbitrary first-order formulas dealing with equality, functions, sets, and aggregates, but no implementation has followed from these definitions so far. Extending ASP with intensional functions was also considered in [29] and later on gave rise to the language $ASP\{f\}$ [26]. These semantics were proven in [30] to be a particular case of [25] for a syntactic fragment where function nesting is restricted but had the advantage of allowing implementation with the system `clingo`$\{f\}$ [31]. In these approaches, the treatment of functions is closer to the defined external atoms in the current paper: that is, functions can be undefined, and they can be used in rule heads to derive their value. However, in general, these semantics are more general in the sense that defined functions can also be used in rule bodies without the requirement of converting them into external. Another strongly related approach based on intensional functions is ASP(MT) [27], whose main difference with respect to [25] is that partial functions are not allowed (functions are always assigned a value). Again, Ref. [30] explains the relation between the two formalisms in detail and establishes a correspondence for a syntactic fragment where functions are not nested.

Another approach that can be classified as a white box system is s(CASP) [32], which is more aligned with the tradition of Constraint Logic Programming (CLP) inherited from Prolog. s(CASP) is a top-down query-based CASP system where constraint atoms may only occur unnegated in rule bodies. Here, constraints are defined in terms of predefined (usually arithmetic) operations performed on regular logic program variables, whose values are assumed to be just numeric elements in the Herbrand domain. This means that the

definition of constraints only makes sense at a non-ground level. The semantics in CLP systems is normally defined in operational terms, and constraints are stored and solved on the fly during the goal-directed execution of rules. Comparison to AMT is difficult since constraints are represented in a substantially different manner: in AMT, they constitute external atoms that can be treated at the ground level (at least from the logic program perspective), whereas in s(CASP), constraints are always defined in terms of first-order variables. One difference that can be mentioned is that since logical variables cannot be undefined, s(CASP) does not produce results in which functions have no assigned value.

Finally, another approach that proposes a full, white-box treatment of constraints is $ASP(\mathcal{AC})$, recently introduced by [33]. $ASP(\mathcal{AC})$ relies on an extension of the logic HT to deal with hybrid logic programs. Here, the orientation is the opposite of the previous approaches in this subsection. Rather than incorporating new operations in a first-order logical language, the logical connectives are generalized as particular cases of more general operations on weighted formulas over semi-rings. This means that, for instance, logical conjunction $\wedge$ becomes just one more possible operation that can be combined with others, such as addition or multiplication (depending on the underlying semi-ring). This provides a very expressive and powerful formalism but at the price of a more complex semantic and the requirement of a semi-ring structure. Still, $ASP(\mathcal{AC})$ covers a wide range of constructs, such as aggregates over non-Boolean variables, linear constraints, or provenance in positive Datalog programs.

7. Summary

We elaborated upon the formal foundations of hybrid ASP systems, focusing on extensions of `clingo` with different forms of linear equations. To this end, we extended logic programs with very general abstract theories without any predefined syntax. To characterize the interplay between theory atoms and their corresponding constraints, we extended the semantics of such programs and distinguished between defined and external theory atoms. Defined theory atoms have to be derived by the logic program for the represented constraint to take effect, while external theory atoms can be seen as oracles for which no justification inside the logic program is needed. A further refinement of abstract theories to structured ones allowed us to distinguish variables inside theory atoms and to assign values to them, leading to a denotation-based semantic. Moreover, this allows us to refine our extended concept of stable models with witnesses provided by assignments for theory variables. We then used the resulting concept of answer sets to characterize the semantic foundations of `clingo`'s hybrid extensions. Finally, we analyzed commonalities and differences to existing approaches of extending logic programs with external non-Boolean theories. These can be separated into black-box and white-box orientations, where in the former, the logic program is largely unaware of the intricacies of the external theory, while in the latter, the external theory is integrated into the logic programming syntax and semantics. The approach in this paper follows a black-box orientation, as the syntax of abstract and structured theories is completely left open and stable models are merely sanctioned by the external theory.

Author Contributions: P.C., J.F., T.S. and P.W. participated evenly in all stages of producing this work. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by DFG grant SCHA 550/11, Germany, by grant PID2020-116201GB-I00 funded by MCIN/AEI/ 10.13039/501100011033, Spain, by Xunta de Galicia and the European Union, GPC ED431B 2022/33, by European COST action CA17124 DigForASP, EU, and by the National Science Foundation (NSF 95-3101-0060-402), USA.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Examples of Abstract Theories

Next, we define a family of abstract theories that encode some propositional logic, having a common set of theory atoms $\mathcal{T}_{pr}$ of the form $\&\mathtt{prop}\{\varphi\}$ for any propositional formula φ that can be formed for a set of variables $\mathcal{X}_{pr} = \{p, q, \dots\}$, standing for propositional atoms, and with a fixed complement $\widehat{\&\mathtt{prop}\{\varphi\}} \stackrel{\text{def}}{=} \&\mathtt{prop}\{\neg\varphi\}$.

Example A1 (Equilibrium logic)**.** *Take the abstract theory $\mathfrak{E} = \langle \mathcal{T}_{pr}, \mathcal{S}, \widehat{\cdot} \rangle$ with set $\mathcal{S}$ contains every set S of atoms whose set of formulas $\Gamma = \{\varphi \mid \&\mathtt{prop}\{\varphi\} \in S\}$ has some equilibrium model and negation as complement, as explained above. For a formal definition of Equilibrium Logic, we refer to [17]: for theories in the form of logic programs (as the ones we comment on below), it suffices to observe that equilibrium models coincide with stable models. As we can see, in this context, theory atoms talk about the inclusion of formulas φ (or their negation $\neg\varphi$) in an external logical theory Γ. For instance, we could write a program like:*

$$\&\mathtt{prop}\{\neg q \to p\}$$
$$\mathtt{a} \leftarrow \&\mathtt{prop}\{\neg p \to q\}$$

Theory atom $\&\mathtt{prop}\{\neg p \to q\}$ is external: if we add it to the program, we obtain a stable model X_1 containing both theory atoms plus the propositional atom $\mathtt{a}$. In fact, X_1 is also a $\mathfrak{E}$-stable model since the induced logical theory $\Gamma_1 = \{(\neg p \to q), (\neg q \to p)\}$ has (two) equilibrium models. If we do not include the external atom, stable model X_2 would just contain the defined atom $\&\mathtt{prop}\{\neg q \to p\}$. This stable model does not pass the $\mathfrak{E}$-check since when we add the complement of the external atom, we get a logical theory $\Gamma_2 = \{(\neg p \to q), \neg(\neg q \to p)\}$ that has no equilibrium model (The formula $\neg(\neg q \to p)$ is HT-equivalent to $\neg q \wedge \neg p$).

Note that the abstract entailment relation $\mid\!\approx_{\mathfrak{T}}$ for this abstract theory does not satisfy monotonicity. For instance, $\&\mathtt{prop}\{\neg p \to q\} \mid\!\approx_{\mathfrak{T}} \&\mathtt{prop}\{q\}$ because adding the complement $\&\mathtt{prop}\{\neg q\}$ to $\&\mathtt{prop}\{\neg p \to q\}$ produces no equilibrium model. Yet, if we extend the theory with atom p, this is not true anymore, namely $\{\&\mathtt{prop}\{\neg p \to q\}, \&\mathtt{prop}\{p\}\} \mid\!\approx_{\mathfrak{T}} \&\mathtt{prop}\{q\}$ does not hold.

Example A2 (Kleene's three-valued logic)**.** *As an example of compositional theory without an absolute complement, consider $\mathfrak{K}$ for capturing Kleene's three-valued propositional logic. We define $\mathfrak{K}$ with the same syntax (theory atoms and complement) as $\mathfrak{E}$, with the only difference that $\mathcal{S}$ now contains every set S of atoms such that $\Gamma = \{\varphi \mid \&\mathtt{prop}\{\varphi\} \in S\}$ has some three-valued model. For the denotations, we have $D_{\mathfrak{U}} = \{0, 1, \frac{1}{2}\}$ and $v \in [\![\, \varphi \,]\!]$ iff $f_v(\varphi) = 1$ for the evaluation function f_v recursively defined as:*

- $f_v(\bot) \stackrel{\text{def}}{=} 0$, $f_v(\top) \stackrel{\text{def}}{=} 1$
- $f_v(x) \stackrel{\text{def}}{=} v(x)$ *for atoms* $x \in \mathcal{X}_{pr}$
- $f_v(\varphi \wedge \psi) \stackrel{\text{def}}{=} \min(f_v(\varphi), f_v(\psi))$
- $f_v(\varphi \vee \psi) \stackrel{\text{def}}{=} \max(f_v(\varphi), f_v(\psi))$
- $f_v(\neg\varphi) \stackrel{\text{def}}{=} 1 - f_v(\varphi)$
- $f_v(\varphi \to \psi) \stackrel{\text{def}}{=} f_v(\neg\varphi \vee \psi)$

As an example, the denotation of atom p would correspond to:

$$[\![\, \&\mathtt{prop}\{p\} \,]\!]_{\mathfrak{K}} = \{(p = 1, q = 0), (p = 1, q = 1/2), (p = 1, q = 1)\}$$

and the denotation of its complement is:

$$[\![\, \&\mathtt{prop}\{\neg p\} \,]\!]_{\mathfrak{K}} = \{(p = 0, q = 0), (p = 0, q = 1/2), (p = 0, q = 1)\}$$

but as we can see, there are valuations, such as $(p = 1/2, q = 0)$, that are not included in either of the two.

Example A3. *Imagine that, in $\mathfrak{K}$, we added the operator $M\varphi$ such that*

$$f_v(M\varphi) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } f_v(\varphi) = 0 \\ 1 & \text{otherwise} \end{cases}$$

that can then be read as "p is not false". Then $\texttt{\&prop}\{Mp\} \mid\!\approx_{\mathfrak{K}} \texttt{\&prop}\{p\}$ since $\Gamma = \{Mp, \neg p\}$ is unsatisfiable because no valuation v can assign $f_v(Mp) = 1$ and $f_v(\neg p) = 1$ simultaneously. However, regular entailment is not satisfied, $\texttt{\&prop}\{Mp\} \not\models_{\mathfrak{K}} \texttt{\&prop}\{p\}$, because there are valuations in $[\![\, \texttt{\&prop}\{Mp\} \,]\!]_{\mathfrak{K}}$ (anyone assigning $1/2$ to p) that are not in $[\![\, \texttt{\&prop}\{p\} \,]\!]_{\mathfrak{K}}$.

Example A4 (Łukasiewicz Fuzzy logic)**.** *Let us consider $\mathfrak{F}$ for capturing Łukasiewicz Fuzzy Logic. We define $\mathfrak{F}$ with the same syntax (theory atoms and complement) as before, but $\mathcal{S}$ now contains every set S of atoms such that $\Gamma = \{\varphi \mid \texttt{\&prop}\{\varphi\} \in S\}$ has some model in Fuzzy Logic. For the denotations, we have $D_{\mathfrak{F}} = [0, 1]$ as an interval over real numbers, and $v \in [\![\, \varphi \,]\!]$ iff $f_v(\varphi) = 1$ for the evaluation function f_v recursively defined as:*

- $f_v(\bot) \stackrel{\text{def}}{=} 0$, $f_v(\top) \stackrel{\text{def}}{=} 1$
- $f_v(x) \stackrel{\text{def}}{=} v(x)$ *for atoms* $x \in \mathcal{X}_{pr}$
- $f_v(\varphi \wedge \psi) \stackrel{\text{def}}{=} \min(f_v(\varphi), f_v(\psi))$
- $f_v(\varphi \vee \psi) \stackrel{\text{def}}{=} \max(f_v(\varphi), f_v(\psi))$
- $f_v(\neg\varphi) \stackrel{\text{def}}{=} 1 - f_v(\varphi)$
- $f_v(\varphi \to \psi) \stackrel{\text{def}}{=} \min(1, 1 - f_v(\varphi) + f_v(\psi))$

We conclude with an example of abstract theory that covers Description Logic $\mathcal{ALC}$ (other variants of Description Logics can be analogously defined). The most relevant feature, in this case, is that both variables and domain elements are partitioned into objects, concepts, and roles.

Example A5 (Description Logic $\mathcal{ALC}$)**.** *Let us consider $\mathfrak{D}$ for capturing Description Logics (DL). The set of variables $\mathcal{X}$, in this case, is partitioned into object names $\mathcal{X}_o$, concept names $\mathcal{X}_c$, and role names $\mathcal{X}_r$. The set $\mathcal{D}$ is also partitioned into $\Delta \cup 2^{\Delta} \cup 2^{\Delta \times \Delta}$, where Δ is a non-empty set representing all the objects in the Description Logic domain. Moreover, valuations $v : \mathcal{X} \to \mathcal{D}$ are also partitioned so that $v : \mathcal{X}_o \to \Delta$ maps each object name to some element in Δ, $v : \mathcal{X}_c \to 2^{\Delta}$ maps each concept name to some subset of Δ, and $v : \mathcal{X}_r \to 2^{\Delta \times \Delta}$ maps each role name to some binary relation or set of pairs of elements from Δ. The theory atoms have the form $\texttt{\&dl}\{\varphi\}$ where φ is an assertion corresponding to one of the expressions:*

$$\varphi ::= C \sqsubseteq C \mid C \equiv C \mid o : C \mid (o, o') : R$$

being $o, o' \in \mathcal{X}_o$ and C some concept expression following the grammar:

$$C \quad ::= \quad A \mid C \sqcap C \mid \forall R.C \mid \exists R \mid \neg C$$

for any $A \in \mathcal{X}_c$ and $R \in \mathcal{X}_r$. As we did before, to define the denotations, we use an evaluation function f_v that, in this case, is used to map any concept expression C into some set of elements in Δ, that is, $f_v(C) \subseteq \Delta$. This function is recursively defined as follows:

$$\begin{aligned}
f_v(A) &\stackrel{\text{def}}{=} v(A) \qquad \text{for any concept name } A \in \mathcal{X}_c \\
f_v(C \sqcap D) &\stackrel{\text{def}}{=} f_v(C) \cap f_v(D) \\
f_v(\forall R.C) &\stackrel{\text{def}}{=} \{x \in \Delta \mid \text{for all } y : (x, y) \in v(R) \text{ implies } y \in f_v(C)\} \\
f_v(\exists R) &\stackrel{\text{def}}{=} \{x \in \Delta \mid \text{there exists } y \text{ such that } (x, y) \in v(R)\} \\
f_v(\neg C) &\stackrel{\text{def}}{=} \Delta \setminus f_v(C)
\end{aligned}$$

Finally, the denotation of a DL theory atom $[\![\texttt{\&dl}\{\varphi\}]\!]$ *is then defined as follows:*

$$\begin{aligned}
[\![\texttt{\&dl}\{C \sqsubseteq D\}]\!] &\stackrel{\text{def}}{=} \{v \mid f_v(C) \subseteq f_v(D)\} \\
[\![\texttt{\&dl}\{C \equiv D\}]\!] &\stackrel{\text{def}}{=} \{v \mid f_v(C) = f_v(D)\} \\
[\![\texttt{\&dl}\{o : C\}]\!] &\stackrel{\text{def}}{=} \{v \mid v(o) \in f_v(C)\} \\
[\![\texttt{\&dl}\{(o, o') : R\}]\!] &\stackrel{\text{def}}{=} \{v \mid (v(o), v(o')) \in v(R)\}
\end{aligned}$$

Appendix B. Proofs of Results

Proof of Proposition 1. It is enough to show that

$$(S \cup \widehat{(\mathcal{E} \setminus S)}) \cup \widehat{(\mathcal{E} \setminus (\mathcal{S} \cup \widehat{(\mathcal{E} \setminus S)}))} = (S \cup \widehat{(\mathcal{E} \setminus S)})$$

Note that $\widehat{A \cup B} = \widehat{A} \cup \widehat{B}$ and $\widehat{A \setminus B} = \widehat{A} \setminus \widehat{B}$. Then,

$$(\mathcal{E} \setminus \widehat{(\mathcal{S} \cup \widehat{(\mathcal{E} \setminus S)})}) = (\widehat{\mathcal{E}} \setminus (\widehat{S} \cup (\widehat{\widehat{\mathcal{E}}} \setminus \widehat{\widehat{S}}))$$

Pick $a \in (\widehat{\mathcal{E}} \setminus (\widehat{S} \cup (\widehat{\widehat{\mathcal{E}}} \setminus \widehat{\widehat{S}})))$. Then, $a \in \widehat{\mathcal{E}}$ and $a \notin (\widehat{S} \cup (\widehat{\widehat{\mathcal{E}}} \setminus \widehat{\widehat{S}}))$. The latter implies that $a \notin \widehat{S}$. Hence, $a \in \widehat{\mathcal{E}} \setminus \widehat{S} = \widehat{\mathcal{E} \setminus S}$.

Then, we have that S is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution that implies $S \cup \widehat{(\mathcal{E} \setminus S)}$ is $\mathfrak{T}$-satisfiable by definition. Due to $(S \cup \widehat{(\mathcal{E} \setminus S)}) \cup \widehat{(\mathcal{E} \setminus (\mathcal{S} \cup \widehat{(\mathcal{E} \setminus S)}))} = (S \cup \widehat{(\mathcal{E} \setminus S)})$, we know that $S \cup \widehat{(\mathcal{E} \setminus S)}$ is $\mathcal{E}$-complete and $(S \cup \widehat{(\mathcal{E} \setminus S)}) \cup \widehat{(\mathcal{E} \setminus (\mathcal{S} \cup \widehat{(\mathcal{E} \setminus S)}))}$ is also $\mathfrak{T}$-satisfiable, then $(S \cup \widehat{(\mathcal{E} \setminus S)})$ is a $\mathcal{E}$-complete $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution.

Conversely, if S is a $\mathcal{E}$-complete $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution, then $S \cup \widehat{(\mathcal{E} \setminus S)} = S$ is $\mathfrak{T}$-satisfiable by definition of $\mathcal{E}$-complete and $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution. □

Proof of Proposition 2. Suppose, for the sake of contradiction, that there exists an incomplete $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution S. Then, $S \neq S \cup \widehat{(\mathcal{E} \setminus S)}$ and, thus, there exists a theory atom $\mathtt{s} \in \widehat{(\mathcal{E} \setminus S)}$ with $\mathtt{s} \in \widehat{\mathcal{E}}$, $\mathtt{s} \notin \widehat{S}$ and $\mathtt{s} \notin S$. Furthermore, $\mathtt{s} \in \widehat{\mathcal{E}}$ implies $\{\mathtt{s}, \widehat{\mathtt{s}}\} \subseteq \mathcal{E}$ since $\mathcal{E}$ is closed. Then, $\widehat{\mathtt{s}} \in \widehat{(\mathcal{E} \setminus S)}$ since $\mathtt{s} \notin S$ and $\mathtt{s} \in \mathcal{E}$. This leads to a contradiction because $\{\mathtt{s}, \widehat{\mathtt{s}}\} \subseteq \widehat{(\mathcal{E} \setminus S)}$, and, therefore, $S \cup \widehat{(\mathcal{E} \setminus S)}$ is not $\mathfrak{T}$-satisfiable. Since $\mathfrak{T}$ is consistent, this implies that S is not a $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution, which is a contradiction with the assumption that S is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution. Consequently, all $\langle \mathfrak{T}, \mathcal{E} \rangle$-solutions S are $\mathcal{E}$-complete for consistent abstract theory $\mathfrak{T}$ and closed $\mathcal{E}$. □

Lemma A1. *Let P be a $\mathcal{T}$-logic program and $\mathfrak{T} = \langle \mathcal{T}, \mathcal{S}, \widehat{\cdot} \rangle$ be an abstract theory. Let $\mathcal{E} \subseteq \mathcal{T}$ be a set of external atoms, and let $\mathtt{s} \in \mathcal{E}$ be one of those external atoms. Let S be a $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution and let P' be the program in (3). Then, X is a stable model of program*

$$P' \cup \{\mathtt{s} \leftarrow B\} \cup \{\mathtt{s} \leftarrow \mid \mathtt{s} \in S\} \cup \{\bot \leftarrow \mathtt{s} \mid \mathtt{s} \notin S\} \tag{A1}$$

iff X is a stable model of program

$$P' \cup \{\bot \leftarrow \neg\mathtt{s}, B\} \cup \{\mathtt{s} \leftarrow \mid \mathtt{s} \in S\} \tag{A2}$$

Proof. We proceed by cases. *Case 1.* Assume that $\mathtt{s} \in S$. Then, (A1) is

$$P' \cup \{\mathtt{s} \leftarrow B\} \cup \{\mathtt{s} \leftarrow\}$$

which is strongly equivalent [34] to $P' \cup \{\mathtt{s} \leftarrow\}$. Similarly, (A2) is

$$P' \cup \{\bot \leftarrow \neg\mathtt{s}, B\} \cup \{\mathtt{s} \leftarrow\}$$

which is also strongly equivalent to $P' \cup \{\ \mathtt{s} \leftarrow\ \}$. *Case 2.* Assume that $\mathtt{s} \notin S$. Then, (A1) is

$$P' \cup \{\mathtt{s} \leftarrow B\} \cup \{\bot \leftarrow \mathtt{s}\}$$

which is strongly equivalent to

$$P' \cup \{\bot \leftarrow B\} \cup \{\bot \leftarrow \mathtt{s}\} \tag{A3}$$

In this case, (A2) is

$$P' \cup \{\bot \leftarrow \neg\mathtt{s}, B\} \tag{A4}$$

Case 2.1. Assume that $\mathtt{s} \in H(P)$. Then, $(\bot \leftarrow \mathtt{s}) \in P'$ and, thus, (A4) can be rewritten as

$$P' \cup \{\bot \leftarrow \neg\mathtt{s}, B\} \cup \{\bot \leftarrow \mathtt{s}\} \tag{A5}$$

Programs (A3) and (A5) are strongly equivalent. *Case 2.2.* Assume that $\mathtt{s} \notin H(P)$. Then, any stable model X of (A3) and (A4) must satisfy $\mathtt{s} \notin X$. Hence, X is a stable model of (A3) and a stable model of (A4). □

Proof of Proposition 3. Let $X \subseteq \mathcal{A} \cup \mathcal{T}$ be a set of atoms. We prove that X is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model of program $P_1 = P \cup \{\mathtt{s} \leftarrow B\}$ iff X is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model of $P_2 = P \cup \{\bot \leftarrow \neg\mathtt{s}, B\}$. Let P' be the program in (3). By definition, X is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model of $P_1 = P \cup \{\mathtt{s} \leftarrow B\}$ iff there is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution S such that X is a stable model of program

$$\begin{aligned} & P_1 \cup \{\mathtt{s}' \leftarrow \mid \mathtt{s}' \in (S \cap \mathcal{E})\} \cup \{\bot \leftarrow \mathtt{s}' \mid \mathtt{s}' \in (\mathcal{T} \cap H(P_1)) \setminus S\} \\ = {} & P' \cup \{\mathtt{s} \leftarrow B\} \cup \{\mathtt{s} \leftarrow \mid \mathtt{s} \in (S \cap \mathcal{E})\} \cup \{\bot \leftarrow \mathtt{s} \mid \mathtt{s} \in \mathcal{T} \setminus S\} \\ = {} & P' \cup \{\mathtt{s} \leftarrow B\} \cup \{\mathtt{s} \leftarrow \mid \mathtt{s} \in S\} \cup \{\bot \leftarrow \mathtt{s} \mid \mathtt{s} \notin S\} \end{aligned}$$

if and only if (Lemma A1) there is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution S such that X is a stable model of program

$$\begin{aligned} & P_2 \cup \{\mathtt{s}' \leftarrow \mid \mathtt{s}' \in (S \cap \mathcal{E})\} \cup \{\bot \leftarrow \mathtt{s}' \mid \mathtt{s}' \in (\mathcal{T} \cap H(P_2)) \setminus S\} \\ = {} & P' \cup \{\leftarrow \neg\mathtt{s}, B\} \cup \{\mathtt{s} \leftarrow \mid \mathtt{s} \in (S \cap \mathcal{E})\} \\ = {} & P' \cup \{\leftarrow \neg\mathtt{s}, B\} \cup \{\mathtt{s} \leftarrow \mid \mathtt{s} \in S\} \end{aligned}$$

iff X is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model $P_2 = P \cup \{\bot \leftarrow \neg\mathtt{s}, B\}$. □

Proof of Proposition 4. By construction, for every $\mathfrak{L}$-satisfiable set $S \in \mathcal{S}$, we have $\bigcap_{\mathtt{s} \in S} [\![\, \mathtt{s} \,]\!]_{\mathfrak{L}} \neq \emptyset$. Then, there exists a valuation $v \in \bigcap_{\mathtt{s} \in S} [\![\, \mathtt{s} \,]\!]_{\mathfrak{L}}$ and $v \in [\![\, \mathtt{s} \,]\!]_{\mathfrak{L}}$ for every $\mathtt{s} \in S$. This implies that $\mathfrak{L}$ is compositional.

Let us show that $\mathfrak{L}$ is consistent. Suppose, for the sake of contradiction, that there exists a $\mathfrak{L}$-satisfiable set $S \in \mathcal{S}$ such that $\{\mathtt{s}, \widehat{\mathtt{s}}\} \in S$ for a $s \in \mathcal{T}$. Then, since $\mathfrak{L}$ is compositional (as shown above), there has to exist a valuation, v, such that $v \in [\![\, \mathtt{s} \,]\!]$ and $v \in [\![\, \widehat{\mathtt{s}} \,]\!]$. By definition of the denotation $[\![\, \cdot \,]\!]_{\mathfrak{T}}$, this implies that v needs to satisfy both $\sum_{1 \leq i \leq n} c_i * v(x_i) \prec c_0$ and $\sum_{1 \leq i \leq n} c_i * v(x_i) \widehat{\prec} c_0$, which is a contradiction. Therefore, every $\mathfrak{L}$-satisfiable set $S \in \mathcal{S}$ is consistent, and then $\mathfrak{L}$ is consistent. □

Proof of Proposition 5. We proceed by contraposition: suppose $[\![\, S \,]\!]_{\mathfrak{T}} \subseteq [\![\, s \,]\!]_{\mathfrak{T}}$ and $S \not\approx_{\mathfrak{T}} s$. Then $S \cup \widehat{s}$ is satisfiable, that is, there exists a valuation $v \in [\![\, S \,]\!]_{\mathfrak{T}} \cap [\![\, \widehat{s} \,]\!]$. However, then, on the one hand, $v \in \bigcap_{s' \in S} [\![\, s' \,]\!]_{\mathfrak{T}}$ and the latter is included in $[\![\, s \,]\!]_{\mathfrak{T}}$, so we obtain $v \in [\![\, s \,]\!]_{\mathfrak{T}}$. On the other hand, $v \in [\![\, \widehat{s} \,]\!]_{\mathfrak{T}}$ and so $v \in [\![\, s \,]\!]_{\mathfrak{T}} \cap [\![\, \widehat{s} \,]\!]_{\mathfrak{T}}$, meaning that $\{s, \widehat{s}\}$ is satisfiable and contradicts the consistency of $\mathfrak{T}$. □

Proof of Proposition 6. Since $\mathfrak{T}$ is compositional, proving $(S \cup \{\widehat{\mathtt{s}}\}) \notin \mathcal{S}$ amounts to checking $[\![\, S \cup \{\widehat{\mathtt{s}}\} \,]\!]_{\mathfrak{T}} = \emptyset$.

For the left to right direction, $S \models_{\mathfrak{T}} s$ is equivalent to $[\![S]\!]_{\mathfrak{T}} \subseteq [\![\mathtt{s}]\!]_{\mathfrak{T}}$. However, then $[\![S]\!]_{\mathfrak{T}} \cap [\![\widehat{\mathtt{s}}]\!]_{\mathfrak{T}} \subseteq [\![\mathtt{s}]\!]_{\mathfrak{T}} \cap [\![\widehat{\mathtt{s}}]\!]_{\mathfrak{T}} = \emptyset$ and so $[\![S]\!]_{\mathfrak{T}} \cap [\![\widehat{\mathtt{s}}]\!]_{\mathfrak{T}} = [\![S \cup \{\widehat{\mathtt{s}}\}]\!]_{\mathfrak{T}} = \emptyset$.

For the right to left direction, we proceed by contraposition. Suppose there is some $v \in [\![S]\!]_{\mathfrak{T}}$ such that $v \notin [\![\mathtt{s}]\!]_{\mathfrak{T}}$. If the complement is absolute, the latter means $v \in [\![\widehat{\mathtt{s}}]\!]_{\mathfrak{T}}$ and so $v \in [\![S]\!]_{\mathfrak{T}} \cap [\![\widehat{\mathtt{s}}]\!]_{\mathfrak{T}} = [\![S \cup \{\widehat{\mathtt{s}}\}]\!]_{\mathfrak{T}}$ and so $[\![S \cup \{\widehat{\mathtt{s}}\}]\!]_{\mathfrak{T}} \neq \emptyset$. □

Proof of Proposition 7. Let X be some $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model. By definition, there is some $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution S such that X is a stable model of the program in (3). By construction of this program, it follows $S \cap \mathcal{E} = X \cap \mathcal{E}$ and $S \cap \mathcal{F} \supseteq X \cap \mathcal{F}$. In addition, since S is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution, it follows that S is $\mathfrak{T}$-satisfiable (S is complete by Proposition 2 and the fact that $\mathcal{E}$ is closed). Let $S' = X \cap \mathcal{T}$. Then, $S' \subseteq S$ and, since $\mathfrak{T}$ is compositional and S is $\mathfrak{T}$-satisfiable, it follows that S' is also $\mathfrak{T}$-satisfiable. That is, there is some valuation, v, such that $v \in [\![\mathtt{s}]\!]$ for all $\mathtt{s} \in S' = X \cap \mathcal{T}$. Let v' be the partial valuation that is only defined in $vars(S')$, and that satisfies $v'(x) = v(x)$ for all $x \in vars(S')$. Then, $v'|_{vars(X \cap \mathcal{T})} \in [\![\mathtt{s}]\!]_{\mathfrak{T}}|_{vars(X \cap \mathcal{T})}$ for all $\mathtt{s} \in S' = X \cap \mathcal{T}$ and, thus, pair $(X \cap \mathcal{A}, v')$ belongs to $ans(X)$. □

Proof of Proposition 8. We start by showing that, if (Y, v) is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-answer set, then $stb_P(Y, v)$ is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model X of P such that (Y, v) belongs to $ans(X)$.

Since (Y, v) is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-answer set, there is a stable model X such that (Y, v) belongs to $ans(X)$. We will show that $stb_P(Y, v) = X$. By definition,

$$\begin{aligned} stb_P(Y,v) \cap \mathcal{A} &= Y = X \cap \mathcal{A} \\ stb_P(Y,v) \cap \mathcal{E} &= \{\, \mathtt{s} \in \mathcal{E} \mid v \in [\![\mathtt{s}]\!]_{\mathfrak{T}}|_{\text{domain}(v)} \,\} \supseteq X \cap \mathcal{E} \\ stb_P(Y,v) \cap \mathcal{F} &= \{\, \mathtt{s} \in \mathcal{F} \mid (B \rightarrow \mathtt{s}) \in P \text{ and } (Y,v) \models_{\mathfrak{T}} B \,\} \end{aligned}$$

Let us now show that

$$stb_P(Y,v) \cap \mathcal{E} = \{\, \mathtt{s} \in \mathcal{E} \mid v \in [\![\mathtt{s}]\!]_{\mathfrak{T}}|_{\text{domain}(v)} \,\} \subseteq X \cap \mathcal{E}$$

Pick any theory atom $\mathtt{s} \in \mathcal{E}$ such that $\mathtt{s} \notin X$. Since X is a stable model, there is some $\langle \mathfrak{T}, \mathcal{E} \rangle$-solution S such that X is a stable model of the program in (3). By construction of program (3), this implies $\mathtt{s} \notin S$. Furthermore, since $\mathcal{E}$ is closed, S is complete (Proposition 2) and thus, both $\widehat{\mathtt{s}} \in S$ and $\widehat{\mathtt{s}} \in X$ follow. Then, $v \in [\![\widehat{\mathtt{s}}]\!]_{\mathfrak{T}}|_{\text{domain}(v)}$ and since the complement is absolute, it follows that $v \notin [\![\mathtt{s}]\!]_{\mathfrak{T}}|_{\text{domain}(v)}$. Hence, $\mathtt{s} \notin stb_P(Y, v) \cap \mathcal{E}$ and

$$stb_P(Y,v) \cap \mathcal{E} = X \cap \mathcal{E}$$

Let us now show that

$$stb_P(Y,v) \cap \mathcal{F} = \{\, \mathtt{s} \in \mathcal{F} \mid (B \rightarrow \mathtt{s}) \in P \text{ and } (Y,v) \models_{\mathfrak{T}} B \,\} = X \cap \mathcal{F} \tag{A6}$$

Pick $\mathtt{s} \in X \cap \mathcal{F}$. Then, there is a rule $(B \rightarrow \mathtt{s}) \in P$ such that $b \in X$ for $b \in B \cap \mathcal{A}$ and $b' \notin X$ for $\neg b' \in B$ and $b' \in \mathcal{A}$. This implies that $b \in Y$ for $b \in B \cap \mathcal{A}$ and $b' \notin Y$ for $\neg b' \in B$ and $b' \in \mathcal{A}$ and, thus, $(Y, v) \models_{\mathfrak{T}} b$ and $(Y, v) \models_{\mathfrak{T}} \neg b'$. Furthermore, for every $b \in B \cap \mathcal{T}$, we have $b \in X$ and, by construction, $v \in [\![\mathtt{s}]\!]_{\mathfrak{T}}|_{\text{domain}(v)}$ and thus $(Y, v) \models b$. Similarly, for every $\neg b \in B$ and $b \in \mathcal{T}$ it holds that $v \notin [\![\mathtt{s}]\!]_{\mathfrak{T}}|_{\text{domain}(v)}$ and, therefore, $(Y, v) \models \neg b$. This implies that $(Y, v) \models_{\mathfrak{T}} B$ and, thus, that $\mathtt{s} \in stb_P(Y, v) \cap \mathcal{F}$. Now, the other way around. Pick $\mathtt{s} \in \mathcal{F}$ such that $(B \rightarrow \mathtt{s}) \in P$ and $(Y, v) \models_{\mathfrak{T}} B$. Then, $B \cap \mathcal{A} \subseteq Y \subseteq X$, $b' \notin Y$ for $\neg b' \in B$ and $b' \in \mathcal{A}$ implying $b' \notin X$, $v \in [\![b]\!]_{\mathfrak{T}}|_{\text{domain}(v)}$ for $b \in B \cap \mathcal{T}$ implying $b \in X \cap \mathcal{T}$, and, $v \notin [\![b']\!]_{\mathfrak{T}}|_{\text{domain}(v)}$ for $\neg b' \in B$ and $b' \in \mathcal{T}$ implying $b' \notin X \cap \mathcal{T}$. Therefore, X satisfies rule body B and must include $\mathtt{s}$ since X is a stable model of P. Hence, $\mathtt{s} \in X \cap \mathcal{F}$ and $stb_P(Y, v) \cap \mathcal{F} = X \cap \mathcal{F}$ follows. Consequently, $stb_P(Y, v) = X$ is a $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable model of P.

Therefore, we get the following correspondences

$$(Y, v) \in ans(stb_P(Y, v)) \qquad \text{for any } \langle \mathfrak{T}, \mathcal{E} \rangle\text{-answer set } (Y, v)$$
$$\{stb_P(Y, v) \mid (Y, v) \in ans(X)\} = \{X\} \qquad \text{for any } \langle \mathfrak{T}, \mathcal{E} \rangle\text{-stable model } X$$

It is easy to see that $stb_P(\cdot)$ and $ans(\cdot)$ establish one-to-one correspondence between the $\langle \mathfrak{T}, \mathcal{E} \rangle$-stable models of P and the equivalence classes with respect to P of its $\langle \mathfrak{T}, \mathcal{E} \rangle$-answer sets. □

References

1. Lifschitz, V. Thirteen Definitions of a Stable Model. In *Fields of Logic and Computation, Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*; Lecture Notes in Computer Science; Blass, A., Dershowitz, N., Reisig, W., Eds.; Springer: Berlin, Germany, 2010; Volume 6300, pp. 488–503. [CrossRef]
2. Gelfond, M.; Lifschitz, V. The Stable Model Semantics for Logic Programming. In Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88), Seattle, WA, USA, 15–19 August 1988; Kowalski, R., Bowen, K., Eds.; MIT Press: Cambridge, MA, USA, 1988; pp. 1070–1080. [CrossRef]
3. Gebser, M.; Harrison, A.; Kaminski, R.; Lifschitz, V.; Schaub, T. Abstract Gringo. *Theory Pract. Log. Program.* **2015**, *15*, 449–463. [CrossRef]
4. Falkner, A.; Friedrich, G.; Schekotihin, K.; Taupe, R.; Teppan, E. Industrial Applications of Answer Set Programming. *Künstliche Intell.* **2018**, *32*, 165–176. [CrossRef]
5. Lierler, Y. Relating constraint answer set programming languages and algorithms. *Artif. Intell.* **2014**, *207*, 1–22. [CrossRef]
6. Kaminski, R.; Schaub, T.; Wanko, P. A Tutorial on Hybrid Answer Set Solving with clingo. In Proceedings of the Thirteenth International Summer School of the Reasoning Web, London, UK, 7–11 July 2017; Lecture Notes in Computer Science; Ianni, G., Lembo, D., Bertossi, L., Faber, W., Glimm, B., Gottlob, G., Staab, S., Eds.; Springer: Berlin, Germany, 2017; Volume 10370, pp. 167–203. [CrossRef]
7. Barrett, C.; Sebastiani, R.; Seshia, S.; Tinelli, C. Satisfiability Modulo Theories. In *Handbook of Satisfiability*; Frontiers in Artificial Intelligence and Applications; Chapter 26; Biere, A., Heule, M., van Maaren, H., Walsh, T., Eds.; IOS Press: Amsterdam, The Netherlands, 2009; Volume 185, pp. 825–885. [CrossRef]
8. Elkabani, I.; Pontelli, E.; Son, T. Smodels with CLP and Its Applications: A Simple and Effective Approach to Aggregates in ASP. In Proceedings of the Twentieth International Conference on Logic Programming (ICLP'04), Saint-Malo, France, 6–10 September 2004; Lecture Notes in Computer Science; Demoen, B., Lifschitz, V., Eds.; Springer: Berlin, Germany, 2004; Volume 3132, pp. 73–89. [CrossRef]
9. Gebser, M.; Ostrowski, M.; Schaub, T. Constraint Answer Set Solving. In Proceedings of the Twenty-Fifth International Conference on Logic Programming (ICLP'09), Pasadena, CA, USA, 14–17 July 2009; Lecture Notes in Computer Science; Hill, P., Warren, D., Eds.; Springer: Berlin, Germany, 2009; Volume 5649, pp. 235–249. [CrossRef]
10. Drescher, C.; Walsh, T. A Translational Approach to Constraint Answer Set Solving. *Theory Pract. Log. Program.* **2010**, *10*, 465–480. [CrossRef]
11. Ostrowski, M.; Schaub, T. ASP modulo CSP: The clingcon system. *Theory Pract. Log. Program.* **2012**, *12*, 485–503. [CrossRef]
12. De Rosis, A.; Eiter, T.; Redl, C.; Ricca, F. Constraint Answer Set Programming Based on HEX-Programs. In Proceedings of the Eighth Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP'15), Cork, Ireland, 13 August 2015; Inclezan, D., Maratea, M., Eds.; 2015.
13. Banbara, M.; Kaufmann, B.; Ostrowski, M.; Schaub, T. Clingcon: The Next Generation. *Theory Pract. Log. Program.* **2017**, *17*, 408–461. [CrossRef]
14. Janhunen, T.; Kaminski, R.; Ostrowski, M.; Schaub, T.; Schellhorn, S.; Wanko, P. Clingo goes Linear Constraints over Reals and Integers. *Theory Pract. Log. Program.* **2017**, *17*, 872–888. [CrossRef]
15. Eiter, T.; Germano, S.; Ianni, G.; Kaminski, T.; Redl, C.; Schüller, P.; Weinzierl, A. The DLVHEX System. *Künstliche Intell.* **2018**, *32*, 187–189. [CrossRef]
16. Gebser, M.; Kaminski, R.; Kaufmann, B.; Ostrowski, M.; Schaub, T.; Wanko, P. Theory Solving Made Easy with Clingo 5. In Proceedings of the Technical Communications of the Thirty-Second International Conference on Logic Programming (ICLP'16), New York, NY, USA, 16–21 October 2016; pp. 2:1–2:15.
17. Pearce, D. Equilibrium logic. *Ann. Math. Artif. Intell.* **2006**, *47*, 3–41. [CrossRef]
18. Lierler, Y. Constraint Answer Set Programming: Integrational and Translational (or SMT-based) Approaches. *Theory Pract. Log. Program.* **2023**, *23*, 195–225. [CrossRef]
19. Balduccini, M.; Lierler, Y. Constraint answer set solver EZCSP and why integration schemas matter. *Theory Pract. Log. Program.* **2017**, *17*, 462–515. [CrossRef]

20. Janhunen, T.; Niemelä, I.; Sevalnev, M. Computing Stable Models via Reductions to Difference Logic. In Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09), Potsdam, Germany, 14–18 September 2009; Lecture Notes in Artificial Intelligence; Erdem, E., Lin, F., Schaub, T., Eds.; Springer: Berlin, Germany, 2009; Volume 5753, pp. 142–154.
21. Liu, G.; Janhunen, T.; Niemelä, I. Answer Set Programming via Mixed Integer Programming. In Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'12), Rome, Italy, 10–14 June 2012; Brewka, G., Eiter, T., McIlraith, S., Eds.; AAAI Press: Palo Alto, CA, USA, 2012; pp. 32–42.
22. Lierler, Y.; Susman, B. SMT-Based Constraint Answer Set Solver EZSMT (System Description). *Comput. Sci. Fac. Proc. Present.* **2016**, *46*, 1:1–1:15.
23. Eiter, T.; Ianni, G.; Schindlauer, R.; Tompits, H. A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer-Set Programming. In Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05), Scotland, UK, 30 July–5 August 2005; Kaelbling, L., Saffiotti, A., Eds.; Morgan Kaufmann Publishers Inc.: Burlington, VT, USA, 2005; pp. 90–96.
24. Brewka, G.; Eiter, T. Equilibria in Heterogeneous Nonmonotonic Multi-Context Systems. In Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI'07), Vancouver, BC, Canada, 22–26 July 2007; AAAI Press: Palo Alto, CA, USA 2007; pp. 385–390.
25. Cabalar, P. Functional answer set programming. *Theory Pract. Log. Program.* **2011**, *11*, 203–233. [CrossRef]
26. Balduccini, M.; Gelfond, M. Language ASP{f} with Arithmetic Expressions and Consistency-Restoring Rules. *arXiv* **2013**, arXiv:1301.1387.
27. Bartholomew, M.; Lee, J. First-order stable model semantics with intensional functions. *Artif. Intell.* **2019**, *273*, 56–93. [CrossRef]
28. Cabalar, P.; Fandinno, J.; Fariñas del Cerro, L.; Pearce, D. Functional ASP with Intensional Sets: Application to Gelfond-Zhang Aggregates. *Theory Pract. Log. Program.* **2018**, *18*, 390–405. [CrossRef]
29. Balduccini, M. A "conservative" approach to extending answer set programming with nonherbrand functions. In *Correct Reasoning: Essays on Logic-Based AI in Honour of Vladimir Lifschitz*; Lecture Notes in Computer Science; Erdem, E., Lee, J., Lierler, Y., Pearce, D., Eds.; Springer: Berlin, Germany, 2012; Volume 7265, pp. 24–39.
30. Bartholomew, M.; Lee, J. On the stable model semantics for intensional functions. *Theory Pract. Log. Program.* **2013**, *13*, 863–876. [CrossRef]
31. Balduccini, M. ASP with non-herbrand partial functions: A language and system for practical use. *Theory Pract. Log. Program.* **2013**, *13*, 547–561. [CrossRef]
32. Arias, J.; Carro, M.; Salatar, E.; Marple, K.; Gupta, G. Constraint Answer Set Programming without Grounding. *Theory Pract. Log. Program.* **2018**, *18*, 337–354. [CrossRef]
33. Eiter, T.; Kiesel, R. ASP($\mathcal{AC}$): Answer Set Programming with Algebraic Constraints. *Theory Pract. Log. Program.* **2020**, *20*, 895–910. [CrossRef]
34. Lifschitz, V.; Pearce, D.; Valverde, A. Strongly equivalent logic programs. *ACM Trans. Comput. Log.* **2001**, *2*, 526–541. [CrossRef]