A simple proof-theoretic characterization of Stable Models

Enrico Giunchiglia[1], Marco Maratea[2] and Marco Mochi[1]

[1]DIBRIS, University of Genova, Italy
[2]University of Calabria, Rende, Italy

Abstract
Stable models of logic programs have been studied and characterized also in comparison with other formalisms by many researchers. As already argued, such characterizations are interesting for many reasons, including the possibility of leading to new algorithms for computing stable models.

In this paper we provide a simple characterization of stable models which can be seen as the proof-theoretic counterpart of the standard model-theoretic definition. We show how it can be naturally encoded in difference logic. Our encoding, compared to the existing reductions to classical logics, does not explicitly rely on a previous computation of Clark's completion, and it does not involve any Boolean variable.

Keywords
Logic programming, stable models, answer set programming, difference logic

1. Introduction

Stable models of logic programs [2, 3, 4, 5, 6, 7], a.k.a. answer set programming (ASP), have been studied and characterized also in comparison with other formalisms by many researchers, given its widespread use to solve application problems, even in an industrial setting [8, 9, 10, 11, 12, 13, 14]. As already argued (see, e.g., [15]), such characterizations are interesting for many reasons, including the possibility of leading to new algorithms for computing stable models.

In this paper we introduce stable derivations as a new characterization of stable models and show how it can be naturally encoded in difference logic, i.e., in quantifier free first order formulas whose atoms have the form

$$(x \bowtie y + c),$$

where x and y are variables ranging over the reals/rationals or the integers, $\bowtie \in \{=, \neq, \leq, <, \geq, >\}$ and c is a numeric constant. While this is neither the first alternative to the standard definition of stable models (see, e.g., [15]) nor the first reduction to difference logic (see, e.g., [16]),

1. our definition of stable derivation is simple, as witnessed by the fact that it is rather short, and
2. its corresponding reduction to difference logic uses only one numeric variable per atom in the program, without the need to include any Boolean variable, given that it does not explicitly rely on a previous computation of Clark's completion [17].

Given the correspondence we establish between stable derivations and stable models, the former can be seen as the proof-theoretic counterpart of the standard model-theoretic definition of the latter.

IPS-RCRA-SPIRIT 2023: Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy. November 7-9th, 2023, Rome, Italy [1]

giunchiglia@unige.it (E. Giunchiglia); marco.maratea@unical.it (M. Maratea); marco.mochi@edu.unige.it (M. Mochi)
http://www.star.dist.unige.it/~marco/ (M. Maratea); https://www.marcomochi.me (M. Mochi)
0000-0002-9034-2527 (M. Maratea); 0000-0002-5849-3667 (M. Mochi)

The paper is structured as follows. First, Section 2 introduces needed preliminaries. Then, Section 3 presents our characterization, while the reduction to difference logic is shown in Section 4. The paper ends by discussing (further) related work in Section 5, and by drawing some conclusion and possible topics for future research in Section 6.

2. Stable models, Clark's completion and ordering constraints

Let V be a countable set of *atoms*. By $V^\perp$ we mean the set obtained adding the atom $\perp$ denoting falsity to V, i.e., $V^\perp = V \cup \{\perp\}$. A *rule* is an expression of the form

$$A \leftarrow A_1, \ldots, A_m, \neg A_{m+1}, \ldots, \neg A_n \tag{1}$$

($0 \leq m \leq n$) where $A, A_1, \ldots, A_n$ are atoms in $V^\perp$ and $\neg$ is the symbol for negation. We assume that in (1), $A_i \neq A_j$ for $1 \leq i < j \leq n$. A *(logic) program* is a set of rules. Given a rule r of the form (1), $head(r) = A$ is the *head*, $body^+(r) = \{A_1, \ldots, A_m\}$ is the set of positive body atoms, $body^-(r) = \{A_{m+1}, \ldots, A_n\}$ is the set of negative body atoms, and $body(r) = \{A_1, \ldots, A_m, \neg A_{m+1}, \ldots, \neg A_n\}$ is the *body* of r. If $A = \perp$ then (1) is said to be a *constraint*.

Consider a logic program Π. A *(truth) assignment* is a subset of the set V of atoms, thus not containing $\perp$. A truth assignment M *satisfies*

1. an atom A if $A \in M$,
2. a negated atom $\neg A$ if $A \notin M$,
3. a set of atoms and negated atoms if M satisfies all the elements in the set,
4. a rule if M satisfies the head whenever M satisfies $body(r)$, and
5. a program Π if M satisfies all the rules in Π, in which case M is also said to be a *model* of Π.

A model M of Π is *minimal* if Π has no other model which is a subset of M. As standard, for a suitable concept S, we write $M \models S$ to mean that M satisfies S.

For any truth assignment M, the *reduct* Π^M of Π relative to M is the set of rules obtained from Π by considering each rule $r \in \Pi$ of the form (1) and dropping r if at least one of the atoms in $A_{m+1}, \ldots, A_n$ is in M, and then dropping $\neg A_{m+1}, \ldots, \neg A_n$ otherwise, i.e.,

$$\Pi^M = \{head(r) \leftarrow body^+(r) : r \in \Pi, M \cap body^-(r) = \emptyset\}.$$

A truth assignment M is a *stable model* of Π if it is the minimal model of the reduct of Π relative to M.

Example 1. *Consider the program Π in the three atoms A, B, C whose rules are:*

$$\begin{array}{c} A \leftarrow B, \\ B \leftarrow A, \\ A \leftarrow \neg C, \\ C \leftarrow C. \end{array} \tag{2}$$

Π has the three models $\{A, B, C\}$, $\{A, B\}$ and $\{C\}$, of which only the last two are minimal and only the second one is stable.

Such definition of stable model is due to [4] and has the property that each stable model M of Π is a *supported model* of Π, i.e., for each atom $A \in V^\perp$, $A \in M$ if and only if there exists a rule $r \in \Pi$ such that $head(r) = A$ and $M \models body(r)$. Thus, all the stable models are also supported while the converse is not necessarily true [18]. [19] proved that also the converse is true if Π is *tight*, i.e., if the *positive dependency graph* of Π

1. having one node for each atom in V, and
2. an edge from A to each atom $A_1, \ldots, A_m$ for each rule (1) in Π,

does not contain any loop.

Theorem 1 ([19]). *Let Π be a program. A stable model of Π is also a supported model of Π, and if Π is tight then a supported model of Π is also a stable model of Π.*

If for each atom A there are finitely many rules with head A, the supported models of Π coincide with the models of the Clark's completion $Comp(\Pi)$ [17]. Assuming Π is finite, the *Clark's completion* $Comp(\Pi)$ of Π is defined to be the set of formulas in propositional logic consisting of

$$A \equiv \bigvee_{r: r \in \Pi, head(r) = A} \bigwedge_{L \in body(r)} L, \tag{3}$$

for each atom $A \in V^{\perp}$.

Note that (3) is included in $Comp(\Pi)$ for every $A \in V^{\perp}$, even when $A = \perp$ or A is not the head of any rule in Π. In the former case, (3) is equivalent to

$$\neg \bigvee_{r: r \in \Pi, head(r) = \perp} \bigwedge_{L \in body(r)} L$$

and, in the latter case, (3) is equivalent to $\neg A$. Clark's completion provides a reduction to classical logic for tight programs. $Comp(\Pi)$ has $|V|$ Boolean variables and size $O(||\Pi||)$, where $||\Pi||$ is the size of Π.

[20] and later [21] generalized Fages' [19] result to programs Π *tight on a set $M \subseteq V$ of atoms*, defined as the programs for which there exists a function λ^{Π} mapping each atom in M to an ordinal such that for each rule r in Π, if M satisfies the head and the body of the rule then, for each atom A in the positive body of the rule, $\lambda(head(r)) > \lambda(A)$. A program is tight according to Fages' [19] definition, if it is tight on every set of atoms.

Theorem 2 ([21]). *Let Π be a program. Let M be a supported model of Π. If Π is tight on M then M is a stable model of Π.*

Example 2. *Consider the rules in (2). If Π consists of the second and third rules then Π is tight and $Comp(\Pi)$ consists of the formulas*

$$\begin{gathered} A \equiv \neg C, \\ B \equiv A, \\ \neg C. \end{gathered}$$

If Π consists of the last three rules in (2), then (i) Π is not tight, (ii) $Comp(\Pi)$ consists of the first two of the above formulas, (iii) we can conclude that $M = \{A, B\}$ is a stable model since Π is tight on M, but (iv) we are not allowed to conclude, on the basis of Theorem 2, that $\{C\}$ is not a stable model.

If Π consists of all the rules in (2), then (i) Π is not tight, (ii) $Comp(\Pi)$ consists of the formulas

$$\begin{gathered} A \equiv (B \vee \neg C), \\ B \equiv A, \\ C \equiv C, \end{gathered} \tag{4}$$

(iii) the set of models of $Comp(\Pi)$ is $\{\{A, B, C\}, \{A, B\}, \{C\}\}$, and (iv) Π is not tight on any of the models of $Comp(\Pi)$.

If the program Π is non tight, several authors showed how it possible to add extra constraints in order to rule out the supported models which are not stable, see for instance [22, 23, 24].

Here, in the following, we give a brief overview of the approaches which are more related to our work. Other related works are briefly discussed in Section 6.

Janhunen [25] proved that in order to rule out the models of the completion which are not stable, it is sufficient to add suitable level ordering constraints. For any truth assignment $M \subseteq V$, define the set of *supporting rules of* M to be $\Pi_M = \{r \in \Pi : M \models \mathit{body}(r)\}$. Given an assignment M, a *level numbering* of M for Π is a function $\lambda^{\Pi} : M \cup \Pi_M \mapsto \mathbb{N}$ such that for each atom $A \in M$,

$$\lambda(A) = min\{\lambda(r) : r \in \Pi_M, head(r) = A\}$$

and, for each rule $r \in \Pi_M$,

$$\lambda(r) = max\{0, max\{\lambda(A) : A \in \mathit{body}^+(r)\}\} + 1.$$

Theorem 3 ([25])**.** *Let* Π *be a program. Let* M *be a supported model of* Π*.* M *is a stable model of* Π *if and only if there exists a level numbering of* M *for* Π*.*

In the same work, Janhunen [25] proved that for a supported model M of Π there is at most one level numbering, and also showed –assuming V is finite– how to encode level numbering in propositional logic using $\lceil log_2(|V|+2)\rceil$ bits. Thanks to Theorem 3, there exists a one-one-correspondence between the stable models of Π and the models of the set $J(\Pi)$ of propositional formulas consisting of the encoding of the level numbering and of $Comp(\Pi)$. $J(\Pi)$ has $O(|V| \times \lceil log_2(|V|)\rceil))$ Boolean variables and size $O(||\Pi|| \times log_2(|V|))$.

A few years later, Niemelä [16] introduced *level ranking* of an assignment M for Π to be a function $\lambda^{\Pi} : M \mapsto \mathbb{N}$ such that for each atom $A \in M$, there exists a rule $r \in \Pi_M$ such that $head(r) = A$ and for each atom $B \in \mathit{body}^+(r)$, $\lambda(A) \geq \lambda(B) + 1$. [16] also showed that if we add the restrictions to level rankings saying that for each $A \in M$

1. $\lambda(A) = 1$ whenever there is a rule $r \in \Pi_M$ with $head(r) = A$ and $\mathit{body}^+(r) = \emptyset$, and
2. for every rule $r \in \Pi_M$ with $head(r) = A$ and $\mathit{body}^+(r) \neq \emptyset$, there exists $B \in \mathit{body}^+(r)$ with $\lambda(A) \leq \lambda(B) + 1$,

then we have a one-to-one correspondence between level ranking and level numbering. Level rankings satisfying such additional restrictions are said to be *strong*.

Theorem 4 ([16])**.** *Let* Π *be a program. Let* M *be a supported model of* Π*.* M *is a stable model of* Π *if and only if there exists a (strong) level ranking of* M *for* Π*.*

Like level numbering, for each supported model M, there is at most one strong level ranking. The strong level ranking can be thus used to produce compact encodings in propositional logic as in [25], but without the need of encoding the level associated to the rules.

(Strong) level rankings can be encoded in difference logic, defined as the extension to propositional logic in which the set of atomic formulas is extended in order to allow for expressions of the form $x \bowtie y+c$, where x and y are variables ranging over a numeric unbounded domain (usually the integers or the rationals/reals), c is a numeric constant and $\bowtie \in \{=, \neq, \leq, <, \geq, >\}$. Then, an *interpretation* σ maps each numeric variable to a value in its domain, and σ *satisfies* an atomic formula $x \bowtie y + c$ iff $\sigma(x) \bowtie \sigma(y) + c$[1]. Thanks to Theorem 4, the stable models of Π can be computed as the models

[1]It is possible to distinguish between *rational/real difference logic* and *integer difference logic*; in the former, variables take values in the rationals/reals while in the latter case variables are assumed to take integer values. The distinction is useful as, e.g., the satisfiability of $0 < x - y < 1$ depends on the domain of x and y. However, in this paper such distinction is useless since we are going to consider formulas whose satisfiability does not depend on the chosen domain.

of $N(\Pi)$, where $N(\Pi)$ is the set of formulas in difference logic consisting of $Comp(\Pi)$ and of the encoding of the (strong) level ranking. $N(\Pi)$ has $|V|$ Boolean variables, $|V|$ numeric variables and size $O(||\Pi||)$, though the introduction of additional Boolean variables may produce a more compact encoding, but still in $O(||\Pi||)$.

Example 3. *Let Π be the set rules in (2). For each atom $A \in V$, we assume to have a numeric variable $\lambda_N(A)$ in the difference logic encoding. Then, $N(\Pi)$, as defined in [16], is equivalent to*

$$\begin{gathered} A \equiv (B \vee \neg C), \\ B \equiv A, \\ C \equiv C, \\ A \rightarrow ((B \wedge \lambda_N(A) \geq \lambda_N(B) + 1) \vee \neg C), \\ B \rightarrow A \wedge \lambda_N(B) \geq \lambda_N(A) + 1, \\ C \rightarrow C \wedge \lambda_N(C) \geq \lambda_N(C) + 1, \end{gathered} \tag{5}$$

where the first 3 formulas correspond to $Comp(\Pi)$ and the other ones are the encoding of the level ranking conditions. Any model of the above formulas satisfy $A, B, \neg C, \lambda_N(B) \geq \lambda_N(A) + 1$.

The formulas encoding the additional conditions on strong level ranking are

$$\begin{gathered} A \rightarrow (\neg B \vee \lambda_N(A) \leq \lambda_N(B) + 1) \wedge (C \vee \lambda_N(A) = \lambda_N(\top)), \\ B \rightarrow \neg A \vee \lambda_N(B) \leq \lambda_N(A) + 1, \\ C \rightarrow \lambda_N(C) \leq \lambda_N(C) + 1, \end{gathered} \tag{6}$$

where $\lambda_N(\top)$ is a "dummy" variable necessary in order to respect the syntax of difference logic and whose intended interpretation is 1. The encoding in difference logic of the strong level ranking corresponds to the formulas in (5) and (6) which impose, assuming the intended interpretation of $\lambda_N(\top)$, that the models satisfy

$$\begin{gathered} \lambda_N(A) = 1, \\ \lambda_N(B) = 2. \end{gathered}$$

As expected, strong level rankings (like level numbering) are unique: each atom in the stable model has a uniquely associated level, while the constraints say nothing about the value of the variables associated to the atoms not belonging to the stable model, in this case $\lambda_N(C)$.

Several other characterizations and corresponding reductions can be introduced on the basis of Niemelä's [16] level ranking. For instance, in the same paper Niemelä [16] defines other reductions based on the strongly connected components (SCCs) of the positive dependency graph associated to Π, and [26] show how it is possible to encode (strong) level rankings (also exploiting SCCs) in SAT modulo acyclicity.

3. A simple proof-theoretic characterization of stable models

This section presents our characterization of stable models.

Consider a program Π. A *stable derivation* is a function λ^Π mapping each atom $A \in V^\perp$ to an ordinal such that $\lambda(A) < \lambda(\perp)$ if and only if there exists a rule $r \in \Pi$ with head A and

1. for each atom $B \in body^+(r)$, $\lambda(A) > \lambda(B)$, and
2. for each atom $B \in body^-(r)$, $\lambda(B) \geq \lambda(\perp)$.

Given a stable derivation λ^Π, the set of atoms *stably derived by* λ^Π is $\{A : \lambda(A) < \lambda(\perp)\}$. A set of atoms M is *stably derivable (from* Π) if there exists a stable derivation of M. From the above definitions, it immediately follows that, in a stable derivation, $\lambda(\perp) = 0$ only if the set of stably derivable atoms is empty.

Example 4. *In the case of the logic program (2), every stable derivation λ^{Π} is such that $\lambda(A) < \lambda(B) < \lambda(\bot) \leq \lambda(C)$ and the only stably derivable set of atoms is $\{A, B\}$. In general, there is more than one stably derivable set of atoms, as in the case of the program*

$$A \leftarrow \neg B,$$
$$B \leftarrow \neg A$$

whose stable derivations satisfy either

$$\lambda(A) < \lambda(\bot) \leq \lambda(B)$$

or

$$\lambda(B) < \lambda(\bot) \leq \lambda(A)$$

and the two corresponding stably derivable sets of atoms are $\{A\}$ and $\{B\}$.

A set of atoms is stably derivable if and only if it is a stable model.

Theorem 5. *Let Π be a program. A set of atoms M is a stable model of Π if and only if M is stably derivable from Π.*

Proof. For the left to right direction, assume M is a stable model. We define a stable derivation for Π via the operator $T_{\Pi^M} : 2^V \mapsto 2^V$ defined, for an arbitrary program Π, as

$$T_{\Pi}(I) = \{head(r) : r \in \Pi, I \models body(r)\},$$

and considering the following sequence of subsets of V

$$T_{\Pi}\uparrow^{0} = \emptyset,$$

and, for each $i \geq 0$,

$$T_{\Pi}\uparrow^{i+1} = T_{\Pi}(T_{\Pi}\uparrow^{i}).$$

Now, since M is a stable model of Π, for each $A \in M$ there is a unique i such that $A \in T_{\Pi^M}\uparrow^{i} \setminus T_{\Pi^M}\uparrow^{i-1}$, and we set $\lambda(A) = i$ iff $A \in T_{\Pi^M}\uparrow^{i} \setminus T_{\Pi^M}\uparrow^{i-1}$. For $A \notin M$, we set $\lambda(A) = \lambda(\bot) = \omega$, the first limit ordinal. Then, $M = T_{\Pi^M}\uparrow^{\omega}$ and for each $A \in V$, $\lambda(A) < \lambda(\bot)$ iff $A \in M$. Clearly, λ^{Π} is a stable derivation for Π: if $\lambda(A) = i < \omega$ then $A \in T_{\Pi^M}\uparrow^{i} \setminus T_{\Pi^M}\uparrow^{i-1}$ and thus there exists a rule $r \in \Pi$ such that (i) for each $B \in body^{+}(r)$, $B \in T_{\Pi^M}\uparrow^{i-1}$ and thus $\lambda(B) < \lambda(A)$, and (ii) for each $B \in body^{-}(r)$, $B \notin M$ and thus $\lambda(B) = \lambda(\bot) = \omega$.

For the right to left direction, suppose there is a stable derivation λ^{Π} for Π. We show that $M = \{A : \lambda(A) < \lambda(\bot)\}$ is the minimal model of Π^M, which implies that M is a stable model of Π. We first show that M is a model of Π^M. Assume it is not. Then, there exists a rule r of the form (1) such that $M \models \{A_1, \dots, A_m, \neg A_{m+1}, \dots, \neg A_n, \neg A\}$ i.e., $\lambda(A_1) < \lambda(\bot)$, …, $\lambda(A_m) < \lambda(\bot)$ while $\lambda(A_{m+1}) \geq \lambda(\bot)$, …, $\lambda(A_n) \geq \lambda(\bot)$, $\lambda(A) \geq \lambda(\bot)$, which implies $\lambda(A_1) < \lambda(A)$, …, $\lambda(A_m) < \lambda(A)$ and $\lambda(A_{m+1}) \geq \lambda(\bot)$, …, $\lambda(A_n) \geq \lambda(\bot)$, $\lambda(A) \geq \lambda(\bot)$, which is not possible since λ^{Π} is a stable derivation. Now we prove that M is minimal. Assume it is not. Then, there is another model $M' \subset M$ of Π^M and an atom $A \in M \setminus M'$ with the lowest value $\lambda(A)$ among the atoms in $M \setminus M'$. Since $A \in M$ then $\lambda(A) < \lambda(\bot)$ and there exists a rule $r \in \Pi$ such that $M \models body(r)$. But, for each $B \in body^{+}(r)$, $B \in M'$ since $\lambda(B) < \lambda(A)$, and for each $B \in body^{-}(r)$, $B \notin M'$ since $M' \subset M$. Thus, $M' \models body(r)$ and then, since M' is a model of Π, $A \in M'$, contradicting the assumption. $\square$

The term "stable derivation" has been used given (i) the analogy with the standard definition of derivation in classical logic, and (ii) the correspondence, as established by Theorem 5, with stable models. Indeed, a stable derivation can be seen as a sequence of applications of rules as in a standard derivation in classical logic, once

1. each rule r is interpreted as the inference rule $head(r) \leftarrow body^+(r)$ carrying the restriction that the whole derivation must not contain the atoms in $body^-(r)$, and
2. each applicable rule r is applied in the derivation.

Differently from classical logic, given the restrictions of the rules, the later application of an applicable rule r may invalidate the (stability of the) derivation. These two differences make the stably derivable relation nonmonotonic –while classical logic is indeed monotonic– but thanks to Theorem 5 we have a nice correspondence between the standard "model-theoretic" definition of stable model and this "proof-theoretic" definition of stable derivation, again similarly to what happens in classical logic.

Comparing the statements of Theorem 2, Theorem 3, and Theorem 4 with Theorem 5, our characterization of stable models does not assume that the starting assignment M is a supported model. If instead we consider our definition of stable derivation, since for any two ordinals α and β the condition $\alpha > \beta$ is equivalent to $\alpha \geq \beta + 1$, it is easy to check the correspondence between the first condition on stable derivation and the condition on level ranking.

As it happens for level rankings, given the freedom in selecting the ordinal associated to each atom, the number of stable derivations is, in general, infinite even in the case of finite programs. If we consider two stable derivations λ_1^Π and λ_2^Π to be *equivalent* if, for each pair of atoms $A, B \in V^\perp$, $\lambda_1^\Pi(A) < \lambda_1^\Pi(B)$ if and only if $\lambda_2^\Pi(A) < \lambda_2^\Pi(B)$, then, whenever $V^\perp$ is finite, there are finitely many non equivalent stable derivations. It is however still possible that there exists two non equivalent stable derivations having the same set of stably derivable atoms.

Example 5. *Consider the logic program Π obtained adding $B \leftarrow \neg C$ to (2). In this case there are three sets of non equivalent stable derivations λ_1^Π, λ_2^Π and λ_3^Π for Π, characterized by $\lambda_1^\Pi(A) < \lambda_1^\Pi(B) < \lambda_1^\Pi(\perp) \leq \lambda_1^\Pi(C)$, $\lambda_2^\Pi(B) < \lambda_2^\Pi(A) < \lambda_2^\Pi(\perp) \leq \lambda_2^\Pi(C)$ and $\lambda_3^\Pi(A) = \lambda^\Pi(B) < \lambda^\Pi(\perp) \leq \lambda^\Pi(C)$. However, all the stable derivations lead to the same set $\{A, B\}$ of stably derivable atoms.*

We can thus define a weaker notion of equivalence, and say that two stable derivations are *weakly equivalent* if they have the same set of stably derivable atoms. Of course, two equivalent stable derivations are also weakly equivalent. Further, similarly to what has been done in [16] for level rankings, we can impose additional restrictions on stable derivations in order to enforce that any two of them are either not weakly equivalent or map $\perp$ to a different ordinal. We do this by introducing strict stable derivations. A stable derivation λ is *strict* if it maps each atom $A \in V$ to an ordinal $\lambda(A)$ such that $\lambda(A) \leq \lambda(\perp)$ and either $\lambda(A) = 1$ or for each rule $r \in \Pi$ with head A,

1. either there exists an atom $B \in body^+(r)$ with $\lambda(A) \leq \lambda(B) + 1$, or
2. there exists an atom $B \in body^-(r)$ with $\lambda(B) < \lambda(\perp)$.

Notice that the above conditions trivially hold when $A = \perp$ and, thus, in a strict stable derivation they hold for each $A \in V^\perp$. Further, it is easy to check that in a strict stable derivation, for each $A \in V^\perp$, $\lambda(A) = 0$ only if $\lambda(\perp) = 0$ and thus only if the set of stably derived atoms is empty.

Theorem 6. *Let Π be a program in the set V of atoms. For any stable derivation λ of Π there is a strict stable derivation λ_1 of Π which is equivalent to λ and such that $\lambda_1(\perp) = |V| + 1$ if V is finite, and $\lambda_1(\perp) = \omega$, otherwise. Any two distinct weakly equivalent strict stable derivations of Π differ only in the ordinal associated to $\perp$.*

Proof. Let M be the set of atoms stably derived by λ. Consider the stable derivation λ_1 having M as stably derived set of atoms constructed in the "left to right" direction of the proof of Theorem 5. By construction λ_1 is strict, equivalent to λ and satisfies $\lambda_1(\perp) = \omega$. On the other hand, it is clear that if V is finite, then the proof still holds if in the proof we replace $M = T_{\Pi^M}\uparrow^\omega$ with $M = T_{\Pi^M}\uparrow^{|V|+1}$ and impose $\lambda_1(\perp) = |V| + 1$.

Assume there are two weakly equivalent strict stable derivations λ_1 and λ_2 and an atom $A \in V$ with $\lambda_1(A) \neq \lambda_2(A)$, and either $\lambda_1(A) \neq \lambda_1(\perp)$ or $\lambda_2(A) \neq \lambda_2(\perp)$. Since λ_1 and λ_2 are weakly equivalent then $\lambda_1(A) < \lambda_1(\bot)$ and $\lambda_2(A) < \lambda_2(\bot)$. Take such atom A to be such that for each atom $B \in V$ with $\lambda_1(B) \neq \lambda_2(B)$, $min(\lambda_1(A), \lambda_2(A)) \leq min(\lambda_1(B), \lambda_2(B))$. Assume $min(\lambda_1(A), \lambda_2(A)) = \lambda_1(A)$ (analogous proof can be done for the other case). Thus, for each atom B with $\lambda_1(B) < \lambda_1(A)$, $\lambda_1(B) = \lambda_2(B)$. Further, from $\lambda_1(A) < \lambda_2(A)$, it follows $\lambda_2(A) > 1$. Then, $\lambda_1(A) < \lambda_1(\bot)$, $\lambda_2(A) < \lambda_2(\bot)$ and the equivalence between λ_1 and λ_2 implies the existence of a rule r with $head(r) = A$ and

1. for each $B \in body^+(r)$, $\lambda_1(B) = \lambda_2(B) < \lambda_1(A) < \lambda_2(A)$,
2. for each $B \in body^-(r)$, $\lambda_1(B) = \lambda_1(\bot)$ and $\lambda_2(B) = \lambda_2(\bot)$, and
3. since $\lambda_2(A) > 1$, there exists $B \in body^+(r)$, $\lambda_1(B) + 1 = \lambda_2(B) + 1 \geq \lambda_2(A) > \lambda_1(A) \geq \lambda_1(B) + 1$ which is not possible.

□

It is worth observing that our definition of strict stable derivation explicitly imposes that for each atom $A \in V$, $\lambda(A) \leq \lambda(\bot)$. However, $\lambda(A) \leq \lambda(\bot)$ is already entailed by the definition of stable derivation for those atoms A for which the rule $A \leftarrow \bot$ is in Π.

4. A simple reduction of stable derivations/models to difference logic

The simple definition of (strict) stable derivation has a correspondingly simple reduction to difference logic, which, thanks to Theorem 5, characterize also stable models.

Consider a finite program Π in a finite set V of variables. In the reduction of Π to difference logic we have a variable $\lambda_{dl}(A)$ for each atom $A \in V^\bot$, while the set $\lambda_{dl}(\Pi)$ of formulas corresponding to Π consists of the formula

$$\lambda_{dl}(A) < \lambda_{dl}(\bot) \equiv \bigvee_{r:r\in\Pi, A=head(r)}(\bigwedge_{B\in body^+(r)}(\lambda_{dl}(A) > \lambda_{dl}(B)) \wedge \bigwedge_{B\in body^-(r)}(\lambda_{dl}(B) \geq \lambda_{dl}(\bot))), \tag{7}$$

for each $A \in V^\bot$. For a set of atoms M, define $\lambda_{dl}(M)$ to be the set of formulas in difference logic

$$\lambda_{dl}(M) = \{\lambda_{dl}(A) < \lambda_{dl}(\bot) : A \in M\} \cup \{\lambda_{dl}(A) \geq \lambda_{dl}(\bot) : A \notin M\}.$$

Theorem 7. *Let Π be a finite program. A set of atoms M is a stable model of Π or, equivalently, is stably derivable from Π if and only if $\lambda_{dl}(\Pi) \cup \lambda_{dl}(M)$ is satisfiable in difference logic.*

Proof. Each formula in $\lambda_{dl}(\Pi)$ is a direct translation of the corresponding condition in the definition of stable derivation.

Thus, for the left to right direction, every stable derivation λ corresponds to an interpretation σ_λ such that for each atom $A \in V$ with $\lambda(A) < \lambda(\bot)$, $\sigma_\lambda(\lambda_{dl}(A)) = \lambda(A)$ and for each atom $A \in V$ with $\lambda(A) \geq \lambda(\bot)$, $\sigma_\lambda(\lambda_{dl}(A)) = V_{max}$, where V_{max} is a value bigger than any value assigned to the variables A with $\lambda(A) < \lambda(\bot)$. σ_λ satisfies $\lambda_{sdl}(\Pi) \cup \lambda_{dl}(M)$ where M is the set of atoms stably derived by λ.

For the right to left direction, if σ is a satisfying interpretation of $\lambda_{dl}(\Pi) \cup \lambda_{dl}(M)$, we can put the set S of values assigned by σ in one to one correspondence with the first $|S|$ ordinals respecting the ordering. Then, if f is the function defining the correspondence between the two sets, for each atom $A \in V^\bot$, define $\lambda(A) = f(\sigma(\lambda_{dl}(A)))$. By construction, for each $A, B \in V^\bot$, $\lambda(A) \leq \lambda(B)$ if and only if $\sigma(\lambda_{dl}(A)) \leq \sigma(\lambda_{dl}(B))$ and, thus, given the correspondence between $\lambda_{dl}(\Pi)$ and the definition of stable derivation, λ is a stable derivation in which M is the set of atoms stably derived by λ. □

According to the above theorem, stably derivable set of atoms of a program Π can be computed with difference logic solvers. We can also provide the corresponding translation for the al conditions holding for strict stable derivations. However, difficulties arise if we consider variables ranging over the rationals/reals. In fact, in such cases, when $\lambda_{dl}(A) < \lambda_{dl}(\bot)$ we would like to impose additional conditions forcing $\lambda_{dl}(A)$ to be the "successor" value of some value $\lambda_{dl}(B) < \lambda_{dl}(A)$, and if $\lambda_{dl}(B)$ ranges over the rationals/reals there is no such successive value. Further, difference logic does not allow to impose that a given variable is greater or equal than a constant and the set of reals/rationals/integers do not have a minimum value.

A simple solution to all the above problems –providing a translation to the observations of the previous section– is to modify condition (7) to

$$\begin{aligned}\lambda_{dl}(A) < \lambda_{dl}(\bot) \equiv\ & \bigvee_{r:r\in\Pi, A=head(r)}(\bigwedge_{B\in body^+(r)}(\lambda_{dl}(A) \geq \lambda_{dl}(B)+1)\wedge \\ & \qquad\qquad\qquad \bigwedge_{B\in body^-(r)}(\lambda_{dl}(B) \geq \lambda_{dl}(\bot))) \\ & \vee \lambda_{dl}(A) > \lambda_{dl}(\bot),\end{aligned} \tag{8}$$

and then include the formula corresponding to the strictness conditions:

$$\begin{aligned}\bigwedge_{r:r\in\Pi, A=head(r)}(\quad & \bigvee_{B\in body^+(r)}(\lambda_{dl}(A) \leq \lambda_{dl}(B)+1)\vee \\ & \bigvee_{B\in body^-(r)}(\lambda_{dl}(B) < \lambda_{dl}(\bot))\vee \\ & \lambda_{dl}(A) = \lambda_{dl}(\top)\ \ ),\end{aligned} \tag{9}$$

where, as it has been the case of the encoding of strong level ranking in difference logic, $\lambda_{dl}(\top)$ is a new variable that is supposed to be interpreted as 1, and which is needed in order to respect the syntax of difference logic. Let $\lambda_{sdl}(\Pi)$ be the set consisting of the formulas (8) and (9), for each $A \in V$.

Theorem 8. *Let Π be a program in a finite set V of atoms. For each $A \in V$, $\lambda_{sdl}(\Pi)$ entails both $\lambda_{dl}(A) \leq \lambda_{dl}(\bot)$ and either $\lambda_{dl}(\top) < \lambda_{dl}(\bot)$ or $\lambda_{dl}(A) = \lambda_{dl}(\bot)$.*

Proof. $\lambda_{dl}(A) \leq \lambda_{dl}(\bot)$ is an easy consequence of (8). Assume there exists a model σ of $\lambda_{sdl}(\Pi)$ satisfying $\lambda_{dl}(A) < \lambda_{dl}(\bot)$ and $\lambda_{dl}(\top) \geq \lambda_{dl}(\bot)$ for some variable $\lambda_{dl}(A)$. Consider such variable to be the one with the lowest $\sigma(\lambda_{dl}(A))$ value. Since $\sigma(\lambda_{dl}(A)) < \sigma(\lambda_{dl}(\bot))$, by (8) there must be a rule r with head A, $body^+(r) = \emptyset$ (otherwise for each $B \in body^+(r)$, $\sigma(\lambda_{dl}(B)) < \sigma(\lambda_{dl}(A))$ contradicting that $\lambda_{dl}(A)$ is the variable with minimum $\sigma(\lambda_{dl}(A))$ value) and each $B \in body^-(r)$ with $\sigma(\lambda_{dl}(B)) = \sigma(\lambda_{dl}(\bot))$. Then, by (9), $\sigma(\lambda_{dl}(A)) = \sigma(\lambda_{dl}(\top))$. But $\sigma(\lambda_{dl}(\top)) = \sigma(\lambda_{dl}(A)) < \sigma(\lambda_{dl}(\bot))$ which contradicts the other initial hypothesis that σ satisfies $\lambda_{dl}(\top) \geq \lambda_{dl}(\bot)$. □

Theorem 9. *Let Π be a program in a finite set V of atoms. Let σ be an interpretation of $\lambda_{sdl}(\Pi)$ such that $\sigma(\lambda_{dl}(\top)) = 1$ and $\sigma(\lambda_{dl}(\bot)) = |V| + 1$. Let λ_σ be the function such that for each $A \in V^\bot$, $\lambda(A) = \sigma(\lambda_{sdl}(A))$. σ is a model of $\lambda_{sdl}(\Pi)$ if and only if λ_σ is a strict stable derivation.*

Proof. Formulas (8) and (9) are a direct translation of the corresponding condition in the definition of strict stable derivation. □

Example 6. *Let Π be the set rules in (2). The set of formulas in $\lambda_{dl}(\Pi)$ are*

$$\begin{gathered}\lambda_{dl}(A) < \lambda_{dl}(\bot) \equiv (\lambda_{dl}(B) < \lambda_{dl}(A) \vee \lambda_{dl}(C) \geq \lambda_{dl}(\bot)), \\ \lambda_{dl}(B) < \lambda_{dl}(\bot) \equiv \lambda_{dl}(A) < \lambda_{dl}(B), \\ \lambda_{dl}(C) < \lambda_{dl}(\bot) \equiv \lambda_{dl}(C) < \lambda_{dl}(C).\end{gathered}$$

Any model of the above formulas satisfies $\lambda_{dl}(A) < \lambda_{dl}(B) < \lambda_{dl}(\bot) \leq \lambda_{dl}(C)$.

The formulas in $\lambda_{sdl}(\Pi)$ encoding strict stable derivations are

$$\begin{gathered}
\lambda_{dl}(A) < \lambda_{dl}(\bot) \equiv (\lambda_{dl}(A) \geq \lambda_{dl}(B) + 1 \lor \lambda_{dl}(C) \geq \lambda_{dl}(\bot) \lor \lambda_{dl}(A) > \lambda_{dl}(\bot)), \\
\lambda_{dl}(B) < \lambda_{dl}(\bot) \equiv (\lambda_{dl}(B) \geq \lambda_{dl}(A) + 1 \lor \lambda_{dl}(B) > \lambda_{dl}(\bot)), \\
\lambda_{dl}(C) < \lambda_{dl}(\bot) \equiv (\lambda_{dl}(C) \geq \lambda_{dl}(C) + 1 \lor \lambda_{dl}(C) > \lambda_{dl}(\bot)), \\
(\lambda_{dl}(A) \leq \lambda_{dl}(B) + 1 \land \lambda_{dl}(C) < \lambda_{dl}(\bot)) \lor \lambda_{dl}(A) = \lambda_{dl}(\top), \\
\lambda_{dl}(B) \leq \lambda_{dl}(A) + 1 \lor \lambda_{dl}(B) = \lambda_{dl}(\top), \\
\lambda_{dl}(C) \leq \lambda_{dl}(C) + 1 \lor \lambda_{dl}(C) = \lambda_{dl}(\top),
\end{gathered}$$

and they entail $\lambda_{dl}(\top) = \lambda_{dl}(A) = \lambda_{dl}(B) - 1 < \lambda_{dl}(C) = \lambda_{dl}(\bot)$.

The proposed encoding in difference logic of (strict) stable derivations has thus $|V| + 1$ numeric variables and size $O(||\Pi||)$, as opposed to [16] encoding of (strong) level ranking which require $|V|$ Boolean variables and $|V|$ numeric variables. Given that we can restrict the range of strict stable derivation in $[1, |V| + 1]$ it is also possible to produce a corresponding encoding in propositional logic mimicking what has been done by [25]. Analogously, it is also possible to further improve the encoding by exploiting the strongly connected component of the positive dependency graph and/or provide "SAT modulo acyclity" encodings again mimicking what has been already proposed in the literature, see, e.g., [25, 16, 27, 26].

To the best of our knowledge, we are the first to provide a linear encoding, in the number of variables of the program, in classical logic (and more specifically in difference logic), without explicitly relying on Clark's completion.

5. Related Work

As mentioned in the Introduction, there have been several characterizations of stable models, with (possibly) corresponding reductions, and some of them are introduced through the paper. Here, we mention some of the main remaining charactizations/reductions to difference logic or other logic-based formalisms other than propositional satisfiability. [26] presented alternative target formalisms in which the acyclicity conditions can be checked using a linear representation. The input program is instrumented such that propositional models of its completion subject to an acyclicity condition checked on graph representation match the answer sets of the program. The required acyclicity can be represented as additional SAT formulas, including difference logic [27]. Such acyclicity conditions can be also linearly represented via SMT with Bit-Vector Logic [28], where the authors introduced additional constraints for atoms involved in SCCs by considering external and internal support for the rules, and Mixed Integer Programming [29]. [30] presented a reduction of programs with monotone and convex constraints to pseudo-Boolean constraints, based on loop formulas. The approach of [31] is based on a syntactic transformation which turns a logic program into a formula of second-order logic similar to the formula from the definition of circumscription. Reductions have been also presented for CASP, an extension of ASP with linear constraints [32], but often limited to difference constraints due to their usefulness in, e.g., scheduling applications, in contrast to native approaches to handle such extension natively (e.g., [33] and solver CLINGO[DL]). Reduction-based approaches include those implemented in EZCSP [34] and EZSMT [35], which rely on CSP and some SMT logics, including difference logic, respectively.

6. Conclusion

In this paper we strengthen the relation between ASP and classical logic [36, 37] by providing a new, simple proof-theoretic characterization of stable models, and a corresponding reduction from logic programs to difference logic, which does not rely on a previous computation of Clark's completion, and it does not involve any Boolean variable. As future work, we would like to implement the reduction as SMT formulas, and test it employing SMT solvers for difference logic (see, e.g., [38, 39, 40]) on benchmarks from the last ASP Competitions [41], against state-of-the-art ASP solvers such as CLINGO [42], LP2DIFF [27], and WASP [43].

References

[1] R. De Benedictis, N. Gatti, M. Maratea, A. Murano, E. Scala, L. Serafini, I. Serina, E. Tosello, A. Umbrico, M. Vallati, Preface to the Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (IPS-RCRA-SPIRIT 2023), in: Proceedings of the Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (IPS-RCRA-SPIRIT 2023) co-located with 22th International Conference of the Italian Association for Artificial Intelligence (AI* IA 2023), 2023.

[2] C. Baral, Knowledge Representation, Reasoning and Declarative Problem Solving, Cambridge University Press, 2003. doi:10.1017/cbo9780511543357.

[3] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Communications of the ACM 54 (2011) 92–103.

[4] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R. A. Kowalski, K. A. Bowen (Eds.), Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, USA, August 15-19, 1988 (2 Volumes), MIT Press, 1988, pp. 1070–1080.

[5] M. Gelfond, V. Lifschitz, Classical Negation in Logic Programs and Disjunctive Databases, New Generation Computing 9 (1991) 365–386.

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

[7] I. Niemelä, Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm, Annals of Mathematics and Artificial Intelligence 25 (1999) 241–273.

[8] E. Erdem, M. Gelfond, N. Leone, Applications of answer set programming, AI Magazine 37 (2016) 53–68.

[9] A. A. Falkner, G. Friedrich, K. Schekotihin, R. Taupe, E. C. Teppan, Industrial applications of answer set programming, Künstliche Intelligenz 32 (2018) 165–176.

[10] M. Gebser, P. Obermeier, T. Schaub, M. Ratsch-Heitmann, M. Runge, Routing driverless transport vehicles in car assembly with answer set programming, Theory and Practice of Logic Programming 18 (2018) 520–534.

[11] C. Dodaro, G. Galatà, A. Grioni, M. Maratea, M. Mochi, I. Porro, An ASP-based solution to the chemotherapy treatment scheduling problem, Theory and Practice of Logic Programming 21 (2021) 835–851.

[12] M. Alviano, C. Dodaro, M. Maratea, Nurse (re)scheduling via answer set programming, Intelligenza Artificiale 12 (2018) 109–124.

[13] C. Dodaro, G. Galatà, M. Maratea, I. Porro, Operating room scheduling via answer set programming, in: AI*IA, volume 11298 of *LNCS*, Springer, 2018, pp. 445–459.

[14] C. Dodaro, G. Galatà, M. K. Khan, M. Maratea, I. Porro, An ASP-based solution for operating room scheduling with beds management, in: P. Fodor, M. Montali, D. Calvanese, D. Roman (Eds.), Proceedings of the Third International Joint Conference on Rules and Reasoning (RuleML+RR 2019), volume 11784 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 67–81.

[15] V. Lifschitz, Thirteen definitions of a stable model, in: A. Blass, N. Dershowitz, W. Reisig (Eds.),

Fields of Logic and Computation, Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday, volume 6300 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 488–503.

[16] I. Niemelä, Stable models and difference logic, Annals of Mathematics and Artificial Intelligence 53 (2008) 313–329.

[17] K. L. Clark, Negation as failure, in: Logic and data bases, Springer, 1978, pp. 293–322.

[18] V. W. Marek, V. S. Subrahmanian, The relationship between stable, supported, default and autoepistemic semantics for general logic programs, Theoretical Computer Science 103 (1992) 365–386.

[19] F. Fages, Consistency of clark's completion and existence of stable models, Methods of Logic in Computer Science 1 (1994) 51–60.

[20] Y. Babovich, E. Erdem, V. Lifschitz, Fages' theorem and answer set programming, CoRR cs.AI/0003042 (2000). URL: https://arxiv.org/abs/cs/0003042.

[21] E. Erdem, V. Lifschitz, Tight logic programs, Theory and Practice of Logic Programming 3 (2003) 499–518.

[22] R. Ben-Eliyahu, R. Dechter, Propositional semantics for disjunctive logic programs, Annals of Mathematics and Artificial Intelligence 12 (1994) 53–87.

[23] F. Lin, J. Zhao, On tight logic programs and yet another translation from normal logic programs to propositional logic, in: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003), Morgan Kaufmann, 2003, pp. 853–858.

[24] F. Lin, Y. Zhao, ASSAT: computing answer sets of a logic program by SAT solvers, Artificial Intelligence 157 (2004) 115–137.

[25] T. Janhunen, Representing normal programs with clauses, in: R. L. de Mántaras, L. Saitta (Eds.), Proceedings of the 16th Eureopean Conference on Artificial Intelligence, (ECAI 2004), IOS Press, 2004, pp. 358–362.

[26] M. Gebser, T. Janhunen, J. Rintanen, Answer set programming as SAT modulo acyclicity, in: T. Schaub, G. Friedrich, B. O'Sullivan (Eds.), Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014), volume 263 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2014, pp. 351–356.

[27] T. Janhunen, I. Niemelä, M. Sevalnev, Computing stable models via reductions to difference logic, in: E. Erdem, F. Lin, T. Schaub (Eds.), Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2009), volume 5753 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 142–154.

[28] M. Nguyen, T. Janhunen, I. Niemelä, Translating answer-set programs into bit-vector logic, in: H. Tompits, S. Abreu, J. Oetsch, J. Pührer, D. Seipel, M. Umeda, A. Wolf (Eds.), Revised Selected Papers - 19th International Conference, (INAP 2011), and 25th Workshop on Logic Programming (WLP 2011) of Applications of Declarative Programming and Knowledge Management, volume 7773 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 95–113.

[29] G. Liu, T. Janhunen, I. Niemelä, Answer set programming via mixed integer programming, in: G. Brewka, T. Eiter, S. A. McIlraith (Eds.), Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning: Proceedings (KR 2012), AAAI Press, 2012.

[30] L. Liu, M. Truszczynski, Properties and applications of programs with monotone and convex constraints, Journal of Artificial Intelligence Research 27 (2006) 299–334.

[31] P. Ferraris, J. Lee, V. Lifschitz, A new perspective on stable models, in: M. M. Veloso (Ed.), Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), 2007, pp. 372–379.

[32] S. Baselice, P. A. Bonatti, M. Gelfond, Towards an integration of answer set and constraint solving, in: M. Gabbrielli, G. Gupta (Eds.), Proceedings of the 21st International Conference on Logic Programming (ICLP 2005), volume 3668 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 52–66.

[33] M. Banbara, B. Kaufmann, M. Ostrowski, T. Schaub, Clingcon: The next generation, Theory and Practice of Logic Programming 17 (2017) 408–461.

[34] M. Balduccini, Y. Lierler, Constraint answer set solver EZCSP and why integration schemas matter, Theory and Practice of Logic Programming 17 (2017) 462–515.

[35] D. Shen, Y. Lierler, Smt-based constraint answer set solver EZSMT+ for non-tight programs, in: M. Thielscher, F. Toni, F. Wolter (Eds.), Proceedings of the Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2018), AAAI Press, 2018, pp. 67–71.

[36] E. Giunchiglia, M. Maratea, On the Relation Between Answer Set and SAT Procedures (or, Between cmodels and smodels), in: ICLP, volume 3668 of *LNCS*, Springer, 2005, pp. 37–51.

[37] E. Giunchiglia, N. Leone, M. Maratea, On the relation among answer set solvers, Ann. Math. Artif. Intell. 53 (2008) 169–204.

[38] A. Armando, C. Castellini, E. Giunchiglia, M. Idini, M. Maratea, TSAT++: an open platform for satisfiability modulo theories, in: W. Ahrendt, P. Baumgartner, H. de Nivelle, S. Ranise, C. Tinelli (Eds.), Selected Papers from the Workshops on Disproving, D@IJCAR 2004, and the Second International Workshop on Pragmatics of Decision Procedures, PDPAR@IJCAR 2004, volume 125 of *Electronic Notes in Theoretical Computer Science*, Elsevier, 2004, pp. 25–36.

[39] L. M. de Moura, N. S. Bjørner, Z3: an efficient SMT solver, in: C. R. Ramakrishnan, J. Rehof (Eds.), Proc. of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008), volume 4963 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 337–340.

[40] B. Dutertre, Yices 2.2, in: A. Biere, R. Bloem (Eds.), Proc. of the Computer Aided Verification - 26th International Conference (CAV 2014), volume 8559 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 737–744.

[41] F. Calimeri, M. Gebser, M. Maratea, F. Ricca, The design of the fifth answer set programming competition, CoRR abs/1405.3710 (2014). URL: http://arxiv.org/abs/1405.3710. `arXiv:1405.3710`.

[42] M. Gebser, B. Kaufmann, T. Schaub, Conflict-driven answer set solving: From theory to practice, Artificial Intelligence 187 (2012) 52–89.

[43] M. Alviano, G. Amendola, C. Dodaro, N. Leone, M. Maratea, F. Ricca, Evaluation of disjunctive programs in WASP, in: M. Balduccini, Y. Lierler, S. Woltran (Eds.), LPNMR, volume 11481 of *LNCS*, Springer, 2019, pp. 241–255.