Progression Semantics for Disjunctive Logic Programs

Yi Zhou
Intelligent Systems Laboratory
School of Computing and Mathematics
University of Western Sydney, NSW, Australia

Yan Zhang
Intelligent Systems Laboratory
School of Computing and Mathematics
University of Western Sydney, NSW, Australia

Abstract

In this paper, we extend the progression semantics for first-order disjunctive logic programs and show that it coincides with the stable model semantics. Based on it, we further show how disjunctive answer set programming is related to Satisfiability Modulo Theories.

Introduction

This work is about Answer Set Programming (ASP), a promising paradigm for declarative problem solving. Since its inception, the syntax of ASP is presented in a first-order language. However, its semantics is traditionally defined in a propositional way by grounding.

Recently, several approaches have been proposed for defining the stable model (answer set) semantics for logic programs directly on the first-order level (Chen et al. 2006; Ferraris, Lee, and Lifschitz 2011; Lin and Zhou 2011; Pearce and Valverde 2004; Zhang and Zhou 2010). Work in this direction is not only theoretically interesting but also practically important for initiating the possibility of developing a first-order ASP solver to bypass the tedious grounding procedure.

Among them, the progression semantics (Zhang and Zhou 2010), similar to the standard semantics for Datalog, is a simple and natural one. According to the progression semantics, a structure $\mathcal{M}$ is a stable model (or an answer set) of a program Π if it is a fixed point of the progression of Π based on $\mathcal{M}$. More precisely, $\mathcal{M}$ coincides with the structure obtained by iteratively and increasingly applying the rules in Π, where the negative parts are fixed by $\mathcal{M}$ itself.

However, in (Zhang and Zhou 2010), the progression semantics is only defined for normal logic programs. As *disjunctive logic programs* is an important and useful extension of normal programs (Eiter, Gottlob, and Mannila 1997; Gelfond and Lifschitz 1991), it is a natural question to ask whether this progression semantics can be extended for disjunctive programs.

Unfortunately, this seems not an easy task since the heads of rules are turning to disjunctions of atoms. As a consequence, there might exist multiple choices when a rule is applied. A naive solution is to randomly pick up one in the head of a rule whenever its body is satisfied. However, this does not work. For instance, the program $\{a; b \leftarrow, a; c \leftarrow\}$ has two stable models, namely $\{a\}$ and $\{b, c\}$. Nevertheless, according to the above definition, $\{a, b\}$ is another one since b might be selected when applying the first rule.

In this paper, we address the difficulties of extending the progression semantics for disjunctive programs by using the following two key ideas. Firstly, we require that the progression is "guarded" by the candidate structure. That is, we always pick up those elements in heads that are already recognized by the original structure. More importantly, we take all possible progressions into account. Roughly speaking, a structure $\mathcal{M}$ is an a stable model of a "disjunctive" program Π iff "all" progressions of Π based on $\mathcal{M}$ converge to $\mathcal{M}$.

This semantics sheds new insights into the stable model (answer set) semantics of (first-order) disjunctive logic programs. As an example, we present an alternative characterization of the stable model semantics in second-order logic augmented with linear arithmetic based on the progression semantics.

Preliminary

We assume that the readers have some basic knowledge about first-order logic. We consider a first-order language with equality but without function symbols. Let $\mathcal{M}$ be a finite structure[1] whose signature contains another signature σ. The *reduct* of $\mathcal{M}$ on σ, denoted by $\mathcal{M}|\sigma$, is the σ-structure that agrees with $\mathcal{M}$ on all interpretations of predicates and constants in σ. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two structures of two disjoint signatures σ_1 and σ_2 respectively, by $\mathcal{M}_1 \cup \mathcal{M}_2$, we denote the $\sigma_1 \cup \sigma_2$-structure that agree with $\mathcal{M}_1$ ($\mathcal{M}_2$) on all interpretations of predicates and constants in σ_1 (σ_2 resp.).

Let A be a domain. A *ground atom* on A is of the form $P(\overline{a})$, where P is a predicate and $\overline{a}$ a tuple of elements that matches the arity of P. Let $\mathcal{M}$ be a structure. For convenience, we also use $P(\overline{a}) \in \mathcal{M}$, or $\mathcal{M} \models P(\overline{a})$, to denote $\overline{a} \in P^{\mathcal{M}}$. Let $\mathcal{H}$ be a set of ground atoms. We use $\mathcal{H} \subseteq \mathcal{M}$, or $\mathcal{M} \models \mathcal{H}$, to denote that for every $P(\overline{a}) \in \mathcal{H}$, $\mathcal{M} \models P(\overline{a})$. For convenience, we also use $\mathcal{M} \cup \mathcal{H}$ to denote the structure obtained from $\mathcal{M}$ by extending every interpretation of predicates with the corresponding ground atoms in $\mathcal{H}$, i.e., for every predicate P, $P^{\mathcal{M} \cup \mathcal{H}} = P^{\mathcal{M}} \cup \{\overline{a} \mid P(\overline{a}) \in \mathcal{H}\}$.

[1] For simplicity and clarity, we only consider finite structures.

Syntax of disjunctive logic programs

An *extended disjunctive rule* (*rule* for short) is of the form:

$$\alpha_1; \ldots; \alpha_m \leftarrow \beta_{m+1}, \ldots, \beta_n, \mathsf{not}\, \gamma_{n+1}, \ldots, \mathsf{not}\, \gamma_k, \mathsf{not\ not}\, \gamma_{k+1}, \ldots, \mathsf{not\ not}\, \gamma_l, \quad (1)$$

where $0 \leq m \leq n \leq k \leq l$, $\alpha_i, 1 \leq i \leq m$ is a non-equality atom, β_j $(m+1 \leq j \leq n)$ and γ_s $(n+1 \leq s \leq l)$ are atoms. Let r be a rule of form (1). The *head* of r, denoted by $Head(r)$, is $\{\alpha_1, \ldots, \alpha_m\}$; the *body* of r, denoted by $Body(r)$, is $\{\beta_{m+1}, \ldots, \beta_n, \mathsf{not}\, \gamma_{n+1}, \ldots, \mathsf{not}\, \gamma_k, \mathsf{not\ not}\, \gamma_{k+1}, \ldots, \mathsf{not\ not}\, \gamma_l\}$. For convenience, we use $Pos(r)$ to denote $\{\beta_{m+1}, \ldots, \beta_n\}$, the *positive body* of r, and $Neg(r)$ to denote $\{\mathsf{not}\, \gamma_{n+1}, \ldots, \mathsf{not}\, \gamma_k, \mathsf{not\ not}\, \gamma_{k+1}, \ldots, \mathsf{not\ not}\, \gamma_l\}$, the *negative body* of r. An *extended disjunctive logic program* (*program* for short if clear from the context) is a finite set of rules. A rule is said to be *normal* if $m = 1$ and $k = l$. Then, a program is said to be *normal* if all the rules in it are normal.

Here, we distinguish between intentional and extensional predicates. A predicate P in a program Π is said to be *intentional* if P occurs in the head of some rules in Π, and *extensional* otherwise. Let Π be a program. We use $\tau(\Pi)$ to denote the signature of Π, including all predicates and constants occurring in Π. In particular, we use $\tau_{ext}(\Pi)$ to denote the extensional signature of Π, including all constants and extensional predicates, and $\tau_{int}(\Pi)$ to denote the intentional signature of Π, including all intentional predicates.

Translational semantics for disjunctive programs

The stable model (answer set) semantics for first-order logic programs has been well discussed in the literature. Here, we briefly review the translational semantics by translating to second-order logic.

Given a program Π, let $\Omega_\Pi = \{Q_1, \ldots, Q_n\}$ be the set of all intentional predicates of Π. Let $\Omega^*_\Pi = \{Q^*_1, \ldots, Q^*_n\}$ be a new set of predicates corresponding to Ω_Π. Given a rule r in Π of the form (1), by $\widehat{r}$, we denote the universal closure of the formula

$$\beta_{m+1} \wedge \cdots \wedge \beta_n \wedge \neg\gamma_{n+1} \wedge \cdots \wedge \neg\gamma_k \wedge \neg\neg\gamma_{k+1} \wedge \cdots \wedge \neg\neg\gamma_l \rightarrow \alpha_1 \vee \cdots \vee \alpha_m;$$

by r^*, we denote the universal closure of the formula

$$\beta^*_{m+1} \wedge \cdots \wedge \beta^*_n \wedge \neg\gamma_{n+1} \wedge \cdots \wedge \neg\gamma_k \wedge \neg\neg\gamma_{k+1} \wedge \cdots \wedge \neg\neg\gamma_l \rightarrow \alpha^*_1 \vee \cdots \vee \alpha^*_m;$$

where $\alpha^*_i = Q^*(\overline{x})$ if $\alpha_i = Q(\overline{x})$ and

$$\beta^*_i, (m+1 \leq i \leq n) = \begin{cases} Q^*_j(\overline{t_j}) & \text{if } \beta_i = Q_j(\overline{t_j}), Q_j \in \Omega_\Pi \\ \beta_i & \text{otherwise.} \end{cases}$$

By $\widehat{\Pi}$, we denote the first-order sentence $\bigwedge_{r\in\Pi} \hat{r}$; by Π^*, we denote the first-order sentence $\bigwedge_{r\in\Pi} r^*$.

Let Π be a program. We use $SM(\Pi)$ to denote the following second-order sentence:

$$\widehat{\Pi} \wedge \neg\exists\Omega^*_\Pi((\Omega^*_\Pi < \Omega_\Pi) \wedge \Pi^*),$$

where $\Omega^*_\Pi < \Omega_\Pi$ is the abbreviation of the formula

$$\bigwedge_{1\leq i\leq n} \forall\overline{x} Q^*_i(\overline{x}) \rightarrow Q_i(\overline{x}) \wedge \neg \bigwedge_{1\leq i\leq n} \forall\overline{x} Q_i(\overline{x}) \rightarrow Q^*_i(\overline{x}).$$

Definition 1 *A structure $\mathcal{M}$ of $\tau(\Pi)$ is called a* stable model *of Π if it is a model of $SM(\Pi)$.*

Note that this translational semantics is essentially the same as the one proposed in (Ferraris, Lee, and Lifschitz 2011) and (Lin and Zhou 2011). Here, we introduce this simplified version because we are focused on extended disjunctive programs rather than arbitrary sentences.

Progression semantics for normal logic programs

The progression semantics for first-order normal logic programs is proposed in (Zhang and Zhou 2010), based on the following notion of evolution stage.

Definition 2 (Evolution stage for normal programs) *Let Π be a normal program and $\Omega_\Pi = \{Q_1, \ldots, Q_n\}$ the set of all intentional predicates of Π. Consider a structure $\mathcal{M}$ of $\tau(\Pi)$. The* t-th simultaneous evolution stage[2] *of Π based on $\mathcal{M}$, denoted as $\mathcal{M}^t(\Pi)$, is a structure of $\tau(\Pi)$ defined inductively as follows:*

- *$\mathcal{M}^0(\Pi) = \mathcal{M}|\tau_{ext}(\Pi) \cup \mathcal{E}|\tau_{int}(\Pi)$, where $\mathcal{E}$ is the structure in which all interpretations of predicates are empty set;*
- *$\mathcal{M}^{t+1}(\Pi) = \mathcal{M}^t(\Pi) \cup \{Head(r)\eta \mid \eta$ is an assignment, $\mathcal{M}^t(\Pi) \models Pos(r)\eta$ and $\mathcal{M} \models Neg(r)\eta\}$, where $\mathcal{M}^t(\Pi) \models Pos(r)\eta$ means that for all $i, m+1 \leq i \leq n$, $\beta_i\eta \in \mathcal{M}^t(\Pi)$; $\mathcal{M} \models Neg(r)\eta$ means that for all $j, n+1 \leq j \leq k$, $\gamma_j\eta \notin \mathcal{M}$.*

The underlying intuition behind Definition 2 is quite simple. As the initial step, $\mathcal{M}^0(\Pi)$ just takes the extensional part of $\mathcal{M}$ into account and set the intentional part to be empty. Then, for each t, $\mathcal{M}^{t+1}(\Pi)$ is the structure expanded from $\mathcal{M}^t(\Pi)$ by applying the rules in Π, where the negative parts are fixed by $\mathcal{M}$. In other words, $\mathcal{M}^{t+1}(\Pi)$ is expanded from $\mathcal{M}^t(\Pi)$ by adding those heads of rules whose positive part is satisfied by $\mathcal{M}^t(\Pi)$ and whose negative part is satisfied by $\mathcal{M}$.

Definition 3 (Progression semantics for normal programs) *Let Π be a first-order normal program and $\mathcal{M}$ a structure of $\tau(\Pi)$. $\mathcal{M}$ is called a* stable model *(or an* answer set*) of Π iff $\mathcal{M}^\infty(\Pi) = \mathcal{M}$.*

Definition 3 simply says that a structure is a stable model of a program iff it is a fixed point of the progression of the program. As shown in (Zhang and Zhou 2010), for normal programs, the progression semantics coincides with the translational semantics.

Theorem 1 (Theorem 1, (Zhang and Zhou 2010)) *Let Π be a normal program and $\mathcal{M}$ a structure of $\tau(\Pi)$. Then, $\mathcal{M}$ is a model of $SM(\Pi)$ iff $\mathcal{M}^\infty(\Pi) = \mathcal{M}$.*

[2] This is called evaluation stage in (Zhang and Zhou 2010).

Extending Progression Semantics for Disjunctive Programs

In this section, we extend the progression semantics for disjunctive programs. Different from normal rules, the head of a disjunctive rule is generally a set of atoms instead of a singleton. As we mentioned in the introduction section, this will cause problems for defining the progression semantics.

Suppose that we are now at the t-th evolution stage. The only problem is all about how to reach the next one. Similar to the progression for normal programs (i.e. Definition 2), we will have a collection of rules which are applicable. Hence, we will get a collection of heads of rules together with assignments, i.e., a collection of sets of ground atoms.

The basic requirement is that for each set in the collection, at least one element in it needs to be selected. To this end, we recall the following notion of (minimal) hitting set.

Definition 4 (Minimal hitting set) *Let S be a set and $\sigma = \{S_1, \ldots, S_t, \ldots\}$ a collection of sets such that for all i, $S_i \subseteq S$. A subset $H \subseteq S$ is said to be a* hitting set *of σ if for all i, $H \cap S_i \neq \emptyset$. Furthermore, H is said to be a* minimal hitting set *of σ if H is a hitting set of σ and there is no $H' \subset H$ such that H' is also a hitting set of σ.*

Then, we can pick up a minimal hitting set of the collection of sets of ground atoms to continue the progression. However, there might exist many possible minimal hitting sets, which means that there might exist different ways to expand the t-th structure. This shows that, for disjunctive programs, there might exist different strategies for progression/evolution. Hence, we will need a notion of evolution sequence to represent each possible way of progression.

Another issue is that, the minimal hitting set selected above might beyond the scope the original structure. Recall the simple example mentioned in the introduction section, which consists of two rules: $a; b \leftarrow$ and $a; c \leftarrow$. Now suppose that the candidate structure is $\{a\}$. At the first evolution step, both the two rules are applicable. Then, we have a collection of atoms, namely $\{\{a, b\}, \{a, c\}\}$, which has two minimal hitting sets, namely $\{a\}$ and $\{b, c\}$. The former is an appropriate choice and it actually shows that $\{a\}$ is a stable model of the program. However, the latter is not recognized by the original structure (i.e. $\{a\}$). If chosen, then the progression will never converge to the the structure $\{a\}$, which means that this progression is failed.

To address this issue, we require that each progression step must be "guarded" by the original structure in the sense that each minimal hitting set selected must be recognized by the original structure. That is, in every progression step, if we want to expand the structure to the next one, the new content must be already in the original structure.

The key definition of evolution sequence for our progression semantics is defined as follows.

Definition 5 (Evolution sequence) *Let Π be a program and $\mathcal{M}$ a structure of $\tau(\Pi)$. Let $\Omega_\Pi = \{Q_1, \ldots, Q_n\}$ be the set of all intentional predicates of Π. An* evolution sequence *of Π based on $\mathcal{M}$, denoted as $\sigma_\mathcal{M}(\Pi)$, is a sequence $\sigma^0_\mathcal{M}(\Pi), \sigma^1_\mathcal{M}(\Pi), \ldots, \sigma^t_\mathcal{M}(\Pi), \ldots$ of structures of $\tau(\Pi)$ such that*

- $\sigma^0_\mathcal{M}(\Pi) = \mathcal{M}|\tau_{ext}(\Pi) \cup \mathcal{E}|\tau_{int}(\Pi)$;
- $\sigma^{t+1}_\mathcal{M}(\Pi) = \sigma^t_\mathcal{M}(\Pi) \cup \mathcal{H}^t$, *where there exists $\mathcal{H}^t \subseteq \mathcal{M}$ such that it is a minimal hitting set of the collection of the following sets:*

$$Head(r)\eta,$$

where r is a rule in Π and η is an assignment such that $Head(r)\eta \cap \sigma^t_M(\Pi) = \emptyset$, $\sigma^t_\mathcal{M}(\Pi) \models Pos(r)\eta$ and $\mathcal{M} \models Neg(r)\eta$.[3]

The basic idea of Definition 5 follows similarly to the evolution stage for normal programs (see Definition 2). That is, we take the extensional structure of $\mathcal{M}$ as the initial input and expand it step-by-step by applying the rules in Π. Again, the negative parts of rules in Π are fixed by $\mathcal{M}$, which is captured by $\mathcal{M} \models Neg(r)\eta$ in the above definition. $\sigma^t_\mathcal{M}(\Pi) \models Pos(r)\eta$ means that this pair of rule and assignment can be applied at the t-th stage, while $Head(r)\eta \cap \sigma^t_M(\Pi) = \emptyset$ means that none of the elements in the head is derived before.

The main differences between them are threefold. Firstly, as there are many possible progressions for a disjunctive program, we use the evolution sequence to distinguish them. Indeed, $\sigma^t_\mathcal{M}(\Pi)$ in Definition 5 plays a similar role of $\mathcal{M}^t(\Pi)$ in Definition 2. Secondly, the new content (i.e. $\mathcal{H}^t$) added are considered as a whole rather than rule-by-rule in Definition 2. In fact, $\mathcal{H}^t$ is a minimal hitting set of the collection of heads of rules applied at the t-th stage. Finally, we require that $\mathcal{H}^t$ is guarded by $\mathcal{M}$, i.e., $\mathcal{H}^t \subseteq \mathcal{M}$.

Based on the notion of evolution sequence, we are able to define our progression semantics for (first-order) disjunctive programs.

Definition 6 (Progression semantics) *Let Π be a (disjunctive) program and $\mathcal{M}$ a structure of $\tau(\Pi)$. $\mathcal{M}$ is called a* stable model *(or an* answer set*) of Π iff there exists at least one evolution sequence, and for all evolution sequence σ of Π based on $\mathcal{M}$, $\sigma^\infty_\mathcal{M}(\Pi) = \mathcal{M}$.*

Two programs are said to be *equivalent* if they have the same set of stable models.

Again, similar to Definition 3, we require that the progression converges to the original structure. That is, $\mathcal{M}$ is a fixed point of the progression. However, the main difference is that we require this is the case "for all" possible progressions. Restricted to normal programs, this definition coincides with Definition 3 as normal programs only have a unique sequence of evolution/progression.

One may wonder whether this "for all" condition can be someway modified into a "there exists" condition. Most likely, this is not the case. Evident from (Eiter, Gottlob, and Mannila 1997), disjunctive logic programs under the stable model semantics is more expressive than normal programs, providing some general assumptions in the computational complexity theory. Definition 6 provides another explanation where the gap between disjunctive ASP and normal ASP lies.

[3]Note that we allow double negation-as-failure in the bodies. Nevertheless, the meaning of $\mathcal{M} \models Neg(r)\eta$ is clear here, which requires additionally that for all $i, k+1 \leq i \leq l$, $r_i\eta \in \mathcal{M}$.

Example 1 [originated from Example 2 in (Eiter, Gottlob, and Mannila 1997)] Consider the following program Π_0:

$$R(x); G(x); B(x) \leftarrow \quad (2)$$
$$NotColored \leftarrow R(x), R(y), E(x,y) \quad (3)$$
$$NotColored \leftarrow G(x), G(y), E(x,y) \quad (4)$$
$$NotColored \leftarrow B(x), B(y), E(x,y) \quad (5)$$
$$R(x) \leftarrow NotColored \quad (6)$$
$$G(x) \leftarrow NotColored \quad (7)$$
$$B(x) \leftarrow NotColored \quad (8)$$
$$NotColored \leftarrow \mathsf{not}\ NotColored \quad (9)$$

E is the only extensional predicates of Π_0, which, ideally, represents the edges of a graph; R, G and B are three unary intentional predicates, which denote three colors red, green and blue respectively; $NotColored$ is a 0-ary intentional predicate to state that the graph is not 3-colorable. In fact, Π_0 is the program for 3-uncolorability. That is, if a graph, represented by E, is not 3-colorable, then Π_0 has a unique stable model, in which all intentional predicates are interpreted as full relation on the domain. On the other hand, if a graph is 3-colorable, then Π_0 has no stable models.

First of all, if Π has a stable model, it must contain $NotColored$ and the full relation for R, G and B. This is because by rule (9), $NotColored$ must be in the stable model, and then by rules (6)-(8), all R, G and B will be derived.

Now we analyze Π_0 by using the progression semantics. Suppose that a graph, represented by E, is not 3-colorable. Consider the structure $\mathcal{A}$ containing E, $NotColored$ and $R^{\mathcal{A}} = G^{\mathcal{A}} = B^{\mathcal{A}} = Dom(\mathcal{A})$. Then, at the first stage, rule (2) is the only applicable one, and each minimal hitting set actually corresponds to a possible color mapping. In fact, there might exist many different mappings thus many different evolution sequences. Since the graph is not 3-colorable, no matter which evolution sequence is selected, there must exist a rule among (3)-(5) which is applicable at the second stage. Hence, $NotColored$ must be derived. Therefore, all R, G and B will be derived at the third stage according to rules (6)-(8). Hence, $\mathcal{A}$ is a stable model of Π_0 if the graph is not 3-colorable. In addition, it is the unique one.

Suppose that the graph is 3-colorable. For the structure $\mathcal{B}$ containing E, $NotColored$ and $R^{\mathcal{B}} = G^{\mathcal{B}} = B^{\mathcal{B}} = Dom(\mathcal{B})$, we can construct an evolution sequence which does not converges to $\mathcal{B}$ by simply select the minimal hitting set that corresponds to the 3-coloring solution at the first stage. Therefore, Π_0 has no stable models in this case.

In general, we present the following main theorem.

Theorem 2 *Let Π be a program and $\mathcal{M}$ a structure of $\tau(\Pi)$. Then, $\mathcal{M}$ is a model of $SM(\Pi)$ iff there exists and for every evolution sequence σ of Π based on $\mathcal{M}$, $\sigma^{\infty}_{\mathcal{M}}(\Pi) = \mathcal{M}$.*

Proof:(sketch)[4] In order to prove this theorem, we introduce an intermediate semantics and show that it is equivalent to both semantics described above.

Let Π be a program and $\mathcal{M}$ a structure of $\tau(\Pi)$. We say that $\mathcal{M}$ is a *stable model* of Π iff

1. for every assignment η and every rule r of form (1) in Π, if $\mathcal{M} \models Body(r)\eta$, then $\mathcal{M} \models Head(r)\eta$.
2. there does not exist a structure $\mathcal{M}'$ of $\tau(\Pi)$ such that
   (a) $\mathsf{Dom}(\mathcal{M}') = \mathsf{Dom}(\mathcal{M})$,
   (b) for each constant c in $\tau(\Pi)$, $c^{\mathcal{M}'} = c^{\mathcal{M}}$,
   (c) for each $P \in \tau_{ext}(\Pi)$, $P^{\mathcal{M}'} = P^{\mathcal{M}}$,
   (d) for all $Q \in \tau_{int}(\Pi)$, $Q^{\mathcal{M}'} \subseteq Q^{\mathcal{M}}$, and for some $Q \in \tau_{int}(\Pi)$, $Q^{\mathcal{M}'} \subset Q^{\mathcal{M}}$,
   (e) for every assignment η and every rule r of form (1) in Π, if $\mathcal{M}' \models Pos(r)\eta$ and $\mathcal{M} \models Neg(r)\eta$, then $\mathcal{M}' \models Head(r)\eta$.

This semantics coincides with the translational semantics because Condition 1 holds iff $\mathcal{M} \models \widehat{\Pi}$, and Condition 2 does not hold iff $\mathcal{M} \models \exists \Omega^*_{\Pi}((\Omega^*_{\Pi} < \Omega_{\Pi}) \wedge \Pi^*)$.

Now we show that this semantics also coincides with the progression semantics. On one hand, suppose that for every evolution sequence σ of Π based on $\mathcal{M}$, $\sigma^{\infty}_{\mathcal{M}}(\Pi) = \mathcal{M}$. Then, Condition 1 holds. Otherwise, there exists a pair (r, η) violating Condition 1. Hence, $\mathcal{M} \models Body(r)\eta$ but $\mathcal{M} \not\models Head(r)\eta$. Consider an evolution sequence σ. We have $\mathcal{M} = \sigma^{\infty}_{\mathcal{M}}(\Pi)$. Thus, there exists a number t such that $\sigma^{t}_{\mathcal{M}}(\Pi) \models Body(r)\eta$. Then, $\sigma^{t+1}_{\mathcal{M}}(\Pi) = \sigma^{t}_{\mathcal{M}}(\Pi) \cup \mathcal{H}^t$. Hence, $\mathcal{H}^t \models Head(r)\eta$ according to the definition. Since $\mathcal{H}^t \subseteq \sigma^{t+1}_{\mathcal{M}}(\Pi) \subseteq \sigma^{\infty}_{\mathcal{M}}(\Pi) = \mathcal{M}$, $\mathcal{M} \models Head(r)\eta$ as well, a contradiction. In addition, Condition 2 must hold as well. Otherwise, let us assume that there exists such an $\mathcal{M}'$. We construct an evolution sequence σ in the same way as in Definition 5 except that we require additionally $\mathcal{H}^t \subseteq \mathcal{M}'$. This evolution sequence is well defined because $\mathcal{M}'$ satisfies Condition 2, particularly Condition 2(e). Now, by induction on the evolution stage t, it can be shown that for all t, $\sigma^{t}_{\mathcal{M}}(\Pi) \subseteq \mathcal{M}'$. Therefore, $\sigma^{\infty}_{\mathcal{M}}(\Pi) \subseteq \mathcal{M}' \subset \mathcal{M}$, a contradiction.

On the other hand, suppose that a structure $\mathcal{M}$ satisfies both Conditions 1 and 2. Then, it can be shown that for every evolution sequence σ, $\sigma^{t}_{\mathcal{M}}(\Pi) \subseteq \mathcal{M}$ by induction on the evolution stage t by Condition 1. Hence, $\sigma^{\infty}_{\mathcal{M}}(\Pi) \subseteq \mathcal{M}$. Now we prove that for any σ, $\sigma^{\infty}_{\mathcal{M}}(\Pi) \not\subset \mathcal{M}$. Otherwise, we construct a structure $\mathcal{M}'$ of $\tau(\Pi)$ in the following way: $\mathsf{Dom}(\mathcal{M}') = \mathsf{Dom}(\mathcal{M})$, for each constant $c \in \tau(\Pi)$, $c^{\mathcal{M}'} = c^{\mathcal{M}}$, for each extensional predicate $P \in \tau_{ext}(\Pi)$, $P^{\mathcal{M}'} = P^{\mathcal{M}}$, and finally, for each intentional predicate $Q \in \Omega_{\Pi}$, $Q^{\mathcal{M}'} = Q^{\sigma^{\infty}_{\mathcal{M}}(\Pi)}$. It can be checked that $\mathcal{M}'$ satisfies Conditions 2(a)-(e) as well, a contradiction. This shows that for any σ, $\sigma^{\infty}_{\mathcal{M}}(\Pi) = \mathcal{M}$. □

A relate work to our progression semantics is due to Minker and Rajasekar (1990), who proposed a fixed point for disjunctive logic programs and showed that it coincides with the minimal model semantics. Different from ours, their definition is defined on so-called extended Herbrand structures but not on arbitrary structures. Also, its relationship to stable model semantics remains unknown.

[4]The proofs in this paper, if given, are sketched due to a space limit.

From Disjunctive ASP to SMT

In this section, we consider an application of the progressions semantics to relate Disjunctive ASP and Satisfiability Modulo Theories (SMT), which are logical theories augmented with some background theories (Nieuwenhuis, Oliveras, and Tinelli 2006). More precisely, we show that the stable model semantics for (first-order) disjunctive logic programs can be characterized alternatively in second-order logic augmented with linear arithmetics.

Recall Definition 5 for evolution sequence. The sequence of structures is constructed inductively. Then, for every intentional ground atom $P(\overline{a}) \in \sigma^{\infty}_{\mathcal{M}}(\Pi)$, there exists a minimal number k such that $P(\overline{a}) \in \sigma^{k}_{\mathcal{M}}(\Pi)$. We call k the *level* of $P(\overline{a})$ in σ. For those $P(\overline{a}) \notin \sigma^{\infty}_{\mathcal{M}}(\Pi)$, we treat its level as ∞.[5] This motives us to consider a notion of level mapping for ground atoms as follows.

Definition 7 (Level mapping) *Let Π be a program and $\mathcal{M}$ a finite structure of $\tau(\Pi)$. A* level mapping *of $\mathcal{M}$ on Π is a function f from all intentional ground atoms to the natural numbers, i.e. $f : \{P(\overline{a}) \mid P \in \tau_{int}(\Pi), \overline{a} \in Dom(\mathcal{M})\} \mapsto \mathbb{N}$. In addition, we say that f is* complete *if*

$$f(P(\overline{a})) = \infty \textit{ iff } P(\overline{a}) \notin \mathcal{M};$$

we say that f is valid *if*

1. *for all $P(\overline{a}) \notin \mathcal{M}$, $f(P(\overline{a})) = \infty$;*
2. *for every pair (r, η) of rule and assignment, if $\mathcal{M} \models Body(r)\eta$ and for all $Q(\overline{a}) \in Head(r)\eta$, $f(Q(\overline{a})) > max(Pos(r)\eta)$,[6] then there exists $Q(\overline{a}) \in Head(r)\eta$ such that $f(Q(\overline{a})) = max(Pos(r)\eta) + 1$.*
3. *for every $Q(\overline{a})$ such that $f(Q(\overline{a})) \neq \infty$, there exists a pair (r, η) of rule and assignment such that $Q(\overline{a}) \in Head(r)\eta$, $f(Q(\overline{a})) = max(Pos(r)\eta) + 1$ and for all other $R(\overline{b}) \in Head(r)\eta$, $f(R(\overline{b})) > f(Q(\overline{a}))$.*

Intuitively, a level mapping represents the minimal steps to derive all intentional ground atoms in $\mathcal{M}$ by iteratively applying rules in Π. It is complete if consistent with the structure $\mathcal{M}$. That is, all intentional ground atoms in $\mathcal{M}$ can be derived and those not in $\mathcal{M}$ cannot (i.e. its level is ∞). It is valid if corresponding to an evolution sequence. Condition 1 means that all intentional ground atoms not in $\mathcal{M}$ cannot be derived. Condition 2 means that every applicable rule (together with an assignment) should be used. $\mathcal{M} \models Body(r)\eta$ means that the rule r is applicable, and $max(Pos(r)\eta)$ represents its stage. The if-then condition means that if none of the elements in the head is not derived before, then at least one of them must be derived at this stage. Finally, Condition 3 is a reverse condition of Condition 2, meaning that every derivable ground atom must be justified by a particular pair of rule and assignment, which is used to derive this atom. Here, $f(R(\overline{b})) > f(Q(\overline{a}))$ means that the other atoms in the head are derived neither before nor at this stage.

[5] Here, we use the notation ∞ for clarity and more readability. Technically, we can use an arbitrarily large integer instead.

[6] For simplicity, by given a set S of atoms, we use $max(S)$ ($min(S)$ resp.) to denote the maximal (minimal) number of $\{f_P(\overline{t}) \mid P(\overline{t}) \in S\}$. In particular, if S contains no intentional predicates, we treat this maximum as 0.

The following proposition indicates the correspondence between evolution sequences and valid level mappings.

Proposition 1 *An evolution sequence σ yields a valid level mapping f, in which all ground atoms are mapped into its level in σ, i.e., $f(P(\overline{a})) = k$, where k is the level of $P(\overline{a})$ in σ. Conversely, a valid level mapping f can be converted back to an evolution sequence σ by grouping the ground atoms at the same level together as the expanded ground atoms, i.e., $\sigma^{k+1}_{\mathcal{M}}(\Pi) = \sigma^{k}_{\mathcal{M}}(\Pi) \cup \mathcal{H}^k$, where $\mathcal{H}^k = \{P(\overline{a}) \mid f(P(\overline{a})) = k\}$.*

Proof:(sketch) "$\Rightarrow$:" Condition 1 holds obviously since $\mathcal{H}^t \subseteq \mathcal{M}$, thus $\sigma^{\infty}_{\mathcal{M}}(\Pi) \subseteq \mathcal{M}$. For Condition 2, if $\mathcal{M} \models Body(r)\eta$ and for all $Q(\overline{a}) \in Head(r)\eta$, $f(Q(\overline{a})) > max(Pos(r)\eta)$, then (r, η) is applicable at some stage t in σ. Hence, there exists $Q(\overline{a}) \in Head(r)\eta$ selected at this stage, i.e., $Q(\overline{a}) \in \mathcal{H}^t$. Thus, $f(Q(\overline{a})) = max(Pos(r)\eta) + 1$. Condition 2 holds. Finally, for Condition 3, if $f(Q(\overline{a})) \neq \infty$, then $Q(\overline{a})$ must be selected at some stage t in σ with respect to a pair (r, η). Hence, $f(Q(\overline{a})) = max(Pos(r)\eta)+1$. In addition, for all other $R(\overline{b}) \in Head(r)\eta$, $f(R(\overline{b})) > f(Q(\overline{a}))$ since none of the other elements in $Head(r)\eta$ is derived before. This shows that Condition 3 holds as well.

"$\Leftarrow$:" It suffices to show that for all k, $\mathcal{H}^k$ is the minimal hitting set of the collection $\{Head(r)\eta\}$ meeting the requirements. This can be proved by induction on k. It holds obviously when $k = 0$. Consider the t-th stage. Condition 1 implies that $\mathcal{H}^t \subseteq \mathcal{M}$. In addition, if a pair (r, η) is applicable, then Condition 2 applies under the induction hypothesis. Therefore, $Head(r)\eta \cap \mathcal{H}^t \neq \emptyset$. Finally, Condition 3 ensures that $\mathcal{H}^t$ is actually minimal since every $Q(\overline{a})$ in $\mathcal{H}^t$ must be selected with respect to a pair (r, η) at this stage. □

Furthermore, the following proposition shows that an evolution sequence converges to the original structure iff its corresponding level mapping is complete.

Proposition 2 *Let σ be an evolution sequence and f its corresponding level mapping. Then, $\sigma^{\infty}_{\mathcal{M}}(\Pi) = \mathcal{M}$ iff f is complete.*

Based on the above observations, we can reformulate the progression semantics for (first-order) disjunctive programs on finite structures to second-order logic augmented with linear arithmetic, similar to SMT. To this end, we need to introduce a set of function variables[7] to represent the level mapping. More precisely, let $\Omega_{\Pi} = \{Q_1, \ldots, Q_n\}$ be the set of intentional predicates in Π, we introduce a set $F_{\Pi} = \{f_{Q_1}, \ldots, f_{Q_n}\}$ of function variables, where f_{Q_i} has the same arity as Q_i. Ideally, F_{Π} is used to simulate a level mapping in the sense that $f(Q_i(\overline{t})\eta)$ (i.e. the level mapping of a ground atom $Q_i(\overline{t})\eta$) is represented as $f_{Q_i}(\overline{t})\eta$ in the language.

Theorem 3 *Let Π be a program and $\mathcal{M}$ a finite structure of $\tau(\Pi)$. Then, $\mathcal{M}$ is a stable model of Π iff it is a model of the following sentence:*

$$\exists F_{\Pi}(VF(\Pi) \wedge CF(\Pi)) \wedge \forall F_{\Pi}(VF(\Pi) \rightarrow CF(\Pi)), \quad (10)$$

[7] An n-ary function variable in second-order logic can be reformulated as an $n+1$-ary predicate variable in a standard way.

where $VF(\Pi)$ *is the conjunction of the following formulas*

$$\bigwedge_{P\in\Omega_\Pi} \forall\overline{x}(\neg P(\overline{x}) \rightarrow [f_P(\overline{x}) = \infty]), \quad (11)$$

$$\bigwedge_{r\in\Pi} \forall\overline{y}(\widehat{Body(r)} \rightarrow \bigvee_{Q(\overline{x})\in Head(r)} Q(\overline{x}) \wedge [min(Head(r)) \leq max(Pos(r)) + 1]), \quad (12)$$

$$\bigwedge_{Q\in\Omega_\Pi} \forall\overline{x}([f_Q(\overline{x}) \neq \infty] \rightarrow \bigvee_{r\in\Pi, Q(\overline{x})\in Head(r)} \exists\overline{y}(\widehat{Body(r)} \wedge [f_Q(\overline{x}) = max(Pos(r)) + 1] \wedge [f_Q(\overline{x}) < min(Head(r)\backslash Q(\overline{x}))])), \quad (13)$$

and $CF(\Pi)$ *is the following formula*

$$\bigwedge_{P\in\Omega_\Pi} \forall\overline{x}(P(\overline{x}) \leftrightarrow [f_P(\overline{x}) \neq \infty]). \quad (14)$$

Proof:(sketch) This assertion follows from Proposition 1, Proposition 2 and Definition 6 by noticing that a structure $\mathcal{M}$ of the signature $\tau(\Pi) \cup F_\Pi$ is a model of $VF(\Pi)$ iff the level mapping f, obtained by $f(Q(\overline{a})) = f_Q(\overline{a})$, is valid; $\mathcal{M}$ is a model of $CF(\Pi)$ iff f is complete. □

The underlying intuitions behind Theorem 3 are quite clear. Above all, formula (10) precisely encodes the progression semantics (see Definition 6). That is, a structure is a stable model of a program iff "there exists" and "for all" evolution sequences (formulated by valid level mappings according to Proposition 1), the resulting structure coincides with the original one (formulated by complete level mappings). Here, $CF(\Pi)$ means that F_Π represents a complete level mapping, while $VF(\Pi)$ means that F_Π represents a valid level mapping. In particular, Formulas (11)-(13) exactly correspond to Conditions 1-3 in Definition 7 respectively.

By restricting Theorem 3 to the propositional case, we can get a translation from propositional disjunctive ASP to quantified Satisfiability Modulo Theories (SMT), where quantifiers may be used over arithmetical variables. This points out the possibility to develop a new kind of disjunctive ASP solver by using a QBF-like solver with some techniques for handling linear constraints. We believe this is a promising approach and leave this to our future investigations.

Also, it is worth mentioning that Theorem 3, restricted to normal logic programs, yields a translation from normal ASP to first-order SMT on finite structures since normal programs only have a unique evolution sequence (i.e. valid level mapping). More precisely, a finite structure $\mathcal{M}$ is a stable model of a normal program Π if it is a model of the formula

$$\exists F_\Pi(VF(\Pi) \wedge CF(\Pi)),$$

or equivalently, it is a reduct of the formula $VF(\Pi)\wedge CF(\Pi)$ on $\tau(\Pi)$. A similar work is proposed recently (Asuncion et al. 2010), whose host language is classical first-order logic instead of first-order SMT. By further restricting this result to the propositional case, we can get a translation from propositional normal ASP to propositional SMT. This result is actually different from Niemela's translation from normal ASP to difference logic (Niemelä 2008) in the sense that it uses an alternative encoding. We leave the detailed comparisons about related work to our full version.

Conclusions

In this paper, we extended the progression semantics for (first-order) disjunctive logic programs and showed that it coincides with the translational stable model semantics (see Theorem 2). This semantics sheds new insights into disjunctive answer set programming. According to the progression semantics, one can conclude that:

disjunctive answer set programming
= rules + negation-as-failure + disjunctive heads (syntactic)
= progression + fixed point + minimal hitting set (semantic)

In contrast, Datalog corresponds to the first component and normal ASP corresponds to the first two, both from a syntactic and a semantic point of view.

Based on the progression semantics, we presented another equivalent characterization of the stable model semantics (see Theorem 3). We believe that this new characterization is not only of theoretical interests but also useful for developing more sophisticated solvers for disjunctive ASP. We leave this as one of our future investigations. Another work worth pursuing is to further extend the progression semantics for arbitrary sentence-like programs, particularly for handling existential quantifiers.

References

Asuncion, V.; Lin, F.; Zhang, Y.; and Zhou, Y. 2010. Ordered completion for first-order logic programs on finite structures. In *AAAI*, 249–254.

Chen, Y.; Lin, F.; Wang, Y.; and Zhang, M. 2006. First-order loop formulas for normal logic programs. In *KR*, 298–307.

Eiter, T.; Gottlob, G.; and Mannila, H. 1997. Disjunctive datalog. *ACM Trans. Database Syst.* 22(3):364–418.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2011. Stable models and circumscription. *Artif. Intell.* 175(1):236–263.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Comput.* 9(3/4):365–386.

Lin, F., and Zhou, Y. 2011. From answer set logic programming to circumscription via logic of GK. *Artif. Intell.* 175(1):264–277.

Minker, J., and Rajasekar, A. 1990. A fixpoint semantics for disjunctive logic programs. *J. Log. Program.* 9(1):45–74.

Niemelä, I. 2008. Stable models and difference logic. *Ann. Math. Artif. Intell.* 53(1-4):313–329.

Nieuwenhuis, R.; Oliveras, A.; and Tinelli, C. 2006. Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). *J. ACM* 53(6):937–977.

Pearce, D., and Valverde, A. 2004. Towards a first order equilibrium logic for nonmonotonic reasoning. In *JELIA*, 147–160.

Zhang, Y., and Zhou, Y. 2010. On the progression semantics and boundedness of answer set programs. In *KR*, 518–526.