Graphs and colorings for answer set programming with preferences: Preliminary Report

Kathrin Konczak, Torsten Schaub, and Thomas Linke

Institut für Informatik, Universität Potsdam
Postfach 90 03 27, D–14439 Potsdam, Germany
{konczak,torsten,linke}@cs.uni-potsdam.de

Abstract. The integration of preferences into answer set programming constitutes an important practical device for distinguishing certain preferred answer sets from non-preferred ones. To this end, we elaborate upon rule dependency graphs and their colorings for characterizing different preference handling strategies found in the literature. We start from a characterization of (three types of) preferred answer sets in terms of totally colored dependency graphs. In turn, we exemplarily develop an operational characterization of preferred answer sets in terms of operators on partial colorings for one particular strategy. In analogy to the notion of a derivation in proof theory, our operational characterization is expressed as a (non-deterministically formed) sequence of colorings, gradually turning an uncolored graph into a totally colored one.

1 Introduction

Graphs constitute a fundamental tool within computing science. Similarly, in answer set programming, graphs are used for deciding whether answer sets exist. Recently, there is even an increased interest in using graphs as the primary computational model for computing answer sets [4, 13]. In fact, one of the distinguishing features of answer set programming is that it provides non-deterministic programming techniques that usually induce multiple distinct answer sets. For filtering out certain preferred answer sets, a prominent approach is to incorporate preference handling into answer set programming. Up to know, preferences where incorporated into answer set solvers either by meta-interpretation [6] or by pre-compilation front-ends [5]; therefore, preferences were never integrated into the solvers themselves. This is where our contribution comes in. We argue that the aforementioned graph-based approaches provide an appropriate model for integrating preferences into answer set programming and the corresponding solvers. We underpin this claim, first, by showing how three among the most prominent preference handling approaches can be characterized by graph-oriented methods and, second, by showing how this can be realized by means of an operational semantics. This is usable for extending graph-based answer set solvers, such as `noMoRe` [1]. Following [10], our idea is to start from an uncolored rule dependency graph and to employ specific operators that turn a partially colored graph gradually into a totally colored one that represents a preferred answer set. This approach, developed in [10] for standard answer set programming, is strongly inspired by the concept of a derivation, in particular, that of an SLD-derivation [14]. Accordingly, a program has a certain preferred

answer set iff there is a sequence of operations turning the uncolored graph into a totally colored one, expressing the answer set.

2 Background

A *logic program* is a finite set of rules such as $p_0 \leftarrow p_1, \ldots, p_m, not\ p_{m+1}, \ldots, not\ p_n$, where $n \geq m \geq 0$, and each p_i $(0 \leq i \leq n)$ is an *atom*. For such a rule r, we let $head(r)$ denote the *head*, p_0, of r and $body(r)$ the *body*, $\{p_1, \ldots, p_m,\ not\ p_{m+1}, \ldots, not\ p_n\}$, of r. Let $body^+(r) = \{p_1, \ldots, p_m\}$ and $body^-(r) = \{p_{m+1}, \ldots, p_n\}$. For a set of rules Π, we write $head(\Pi) = \{head(r) \mid r \in \Pi\}$ and conversely, for an atom p, we define $rule(p) = \{r \in \Pi \mid head(r) = p\}$. A program is *basic* if $body^-(r) = \emptyset$ for all its rules. The *reduct*, Π^X, of a program Π *relative to* a set X of atoms is defined by $\Pi^X = \{head(r) \leftarrow body^+(r) \mid r \in \Pi, body^-(r) \cap X = \emptyset\}$. A set of atoms X is *closed under* a basic program Π if for any $r \in \Pi$, $head(r) \in X$ if $body^+(r) \subseteq X$. The smallest set of atoms being closed under a basic program Π is denoted by $Cn(\Pi)$. Then, a set X of atoms is an *answer set* of a program Π if $Cn(\Pi^X) = X$. We use $AS(\Pi)$ for denoting the set of all answer sets of Π. In what follows, an important concept is that of the *generating rules* of an answer set. The set $R_\Pi(X)$ of generating rules of a set X of atoms from program Π is given by

$$R_\Pi(X) = \{r \in \Pi \mid body^+(r) \subseteq X, body^-(r) \cap X = \emptyset\}.$$

An *ordered logic program* is a pair $(\Pi, <)$, where Π is a logic program and $< \subseteq \Pi \times \Pi$ is a strict partial order. Given, $r_1, r_2 \in \Pi$, the relation $r_1 < r_2$ expresses that r_2 has *higher priority* than r_1. This informal interpretation can be made precise in different ways. In what follows, we consider three such interpretations: D– [5], B– [3], and W– preference [17]. Given $(\Pi, <)$, all of them use $<$ for selecting preferred answer sets among the standard answer sets of Π. As shown in [17], the three strategies yield an increasing number of preferred answer sets. That is, D is stronger than W, which is stronger than B, which is stronger than no preference. For brevity, we give below only a formal definition of D–preference, and refer the reader to the literature regarding B– and W– preference [3, 17].

Definition 1. *Let $(\Pi, <)$ be an ordered program and let X be an answer set of Π. Then, X is called $<^D$–preserving, if there exists an enumeration $\langle r_i \rangle_{i \in I}$ of $R_\Pi(X)$ such that for every $i, j \in I$ we have that:*

1. *if $r_i < r_j$, then $j < i$,*
2. *$body^+(r_i) \subseteq \{head(r_j) \mid j < i\}$, and*
3. *if $r_i < r'$ and $r' \in \Pi \setminus R_\Pi(X)$, then*
   (a) $body^+(r') \not\subseteq X$ or
   (b) $body^-(r') \cap \{head(r_j) \mid j < i\} \neq \emptyset$.

Condition 1 stipulates that $\langle r_i \rangle_{i \in I}$ is compatible with $<$. Condition 2 makes the property of supportness explicit. Although any standard answer set is generated by a supported sequence of rules, in D–preferences, rules cannot be supported by lower-ranked

rules. Condition 3a separates the handling of unsupported rules from preference handling. Condition 3b guarantees that rules can never be blocked by lower-ranked rules. For W–preference, the previous concept of order preservation is weakened in Condition 1 and 3 for suspending both conditions, whenever the head of a preferred rule is derivable in an alternative way. Roughly speaking, B–preference additionally drops Condition 2; thus decoupling preference handling from the order induced by consecutive rule applications. Define $AS^{\sigma}((\Pi, <))$ as the set of all $<^{\sigma}$–preserving answer sets for $\sigma \in \{D, B, W\}$.

3 Graphs and colorings with preferences

This section lays the graph-theoretical foundations of our approach. A *graph* is a pair (V, E) where V is a set of *vertices* and $E \subseteq V \times V$ a set of (directed) *edges*. A graph (V, E) is *acyclic* if E contains no cycles. For $W \subseteq V$, we denote $E \cap (W \times W)$ by $E|_W$. Also, we abbreviate $G = (V \cap W, E|_W)$ by $G|_W$. A *subgraph* of (V, E) is a graph (W, F) such that $W \subseteq V$ and $F \subseteq E|_W$.

In the sequel, we are interested in labeled graphs reflecting dependencies among rules.

Definition 2. *Let $(\Pi, <)$ be an ordered logic program. The ordered rule dependency graph (DG) $\Gamma_{(\Pi,<)} = (\Pi, E_0, E_1, E_2)$ of $(\Pi, <)$ is a labeled directed graph with*

$$E_0 = \left\{(r, r') \mid r, r' \in \Pi, head(r) \in body^+(r')\right\};$$
$$E_1 = \left\{(r, r') \mid r, r' \in \Pi, head(r) \in body^-(r')\right\};$$
$$E_2 = \left\{(r, r') \mid r, r' \in \Pi, r' < r\right\}.$$

This definition extends the one in [13] by 2-edges for representing preferences among rules. Whenever clear from the context, we write Γ instead of $\Gamma_{(\Pi,<)}$. An i-subgraph (V, E) of Γ is a subgraph of Γ with $E \subseteq E_i$ for $i \in \{0, 1, 2\}$.

For illustration, consider the ordered program $(\Pi_1, <) = (\{r_1, \ldots, r_4\}, <)$, where:

$$\begin{array}{lll} r_1 : p \leftarrow & r_3 : \ f \leftarrow b, not\ f' & \\ r_2 : b \leftarrow p & r_4 : f' \leftarrow p, not\ f & r_3 < r_4 \end{array} \tag{1}$$

Among the two standard answer sets of Π_1, $\{p, b, f\}$, and $\{p, b, f'\}$, the preference $r_3 < r_4$ selects the latter. That is,

$$AS^D((\Pi_1, <)) = \{\{p, b, f'\}\}.^1$$

The *DG* of $(\Pi_1, <)$ is depicted in Figure 1a. For instance, $(\{r_1, r_2, r_4\}, \{(r_1, r_2)\})$ is a 0-subgraph of $\Gamma_{(\Pi_1,<)}$.

We call C a *coloring* of $\Gamma_{(\Pi,<)}$ if C is a mapping $C : \Pi \to \{\oplus, \ominus\}$. Intuitively, the colors $\oplus$ and $\ominus$ indicate whether a rule is supposedly applied or blocked. We sometimes denote the set of all vertices colored with $\oplus$ or $\ominus$ by $C_{\oplus}$ or $C_{\ominus}$, respectively. That is, $C_{\oplus} = \{r \mid C(r) = \oplus\}$ and $C_{\ominus} = \{r \mid C(r) = \ominus\}$. If C is total, $(C_{\oplus}, C_{\ominus})$ is a binary partition of Π. That is, $\Pi = C_{\oplus} \cup C_{\ominus}$ and $C_{\oplus} \cap C_{\ominus} = \emptyset$. Accordingly, we often identify a coloring C with the pair $(C_{\oplus}, C_{\ominus})$. A *partial* coloring C induces a pair $(C_{\oplus}, C_{\ominus})$ of

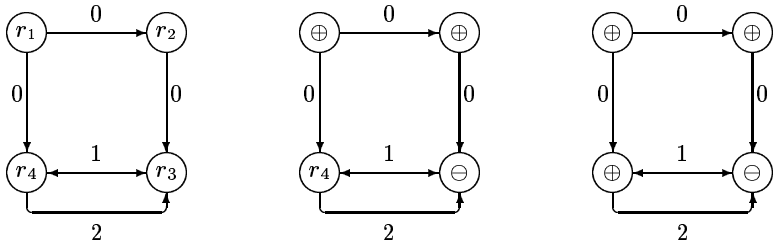

Fig. 1. (a): The *DG* of ordered logic program $(\Pi_1, <)$; **(b)**: The (partially) colored *DG* $(\Gamma_{(\Pi_1,<)}, C_2)$; **(c)** The totally colored *DG* $(\Gamma_{(\Pi_1,<)}, C_3)$.

sets such that $C_\oplus \cup C_\ominus \subseteq \Pi$ and $C_\oplus \cap C_\ominus = \emptyset$. For comparing partial colorings, C and C', we define $C \sqsubseteq C'$, if $C_\oplus \subseteq C'_\oplus$ and $C_\ominus \subseteq C'_\ominus$. The "empty" coloring $(\emptyset, \emptyset)$ is the $\sqsubseteq$-smallest coloring. Accordingly, we define $C \sqcup C'$ as $(C_\oplus \cup C'_\oplus, C_\ominus \cup C'_\ominus)$.

If C is a coloring of Γ, we call the pair (Γ, C) a colored *DG*. For example, "coloring" the *DG* of $(\Pi_1, <)$ with

$$C_2 = (\{r_1, r_2\},\ \{r_3\}) \tag{2}$$

yields the colored graph given in Figure 1b. For simplicity, when coloring, we replace the label of a node by the respective color.

The central question addressed in this paper is how to compute the total colorings of *DG*s that correspond to the preferred answer sets of an underlying program. In fact, the colorings of interest can be distinguished in a straightforward way. Given an ordered logic program $(\Pi, <)$ along with its *DG* Γ. Let $\sigma \in \{D, B, W\}$. Then, for every $<^\sigma$–preserving answer set X of Π, define an *$<^\sigma$–preserving admissible coloring* C of Γ as

$$C = (R_\Pi(X), \Pi \setminus R_\Pi(X)).$$

By way of the respective generating rules, we associate with any program a set of $<^\sigma$–preserving admissible colorings whose members are in one-to-one correspondence with its $<^\sigma$–preserving answer sets. Clearly, any $<^\sigma$–preserving admissible coloring is total; also, we have $X = head(C_\oplus)$. We use $AC^\sigma((\Pi, <))$ for denoting the set of all $<^\sigma$–preserving admissible colorings of a *DG* $\Gamma_{(\Pi,<)}$. For a partial coloring C, we define

$$AC^\sigma_{(\Pi,<)}(C) = \{C' \in AC^\sigma((\Pi, <)) \mid C \sqsubseteq C'\}$$

as the set of all $<^\sigma$–preserving admissible colorings of $\Gamma_{(\Pi,<)}$ compatible with C. Clearly, $C_1 \sqsubseteq C_2$ implies $AC^\sigma_{(\Pi,<)}(C_1) \supseteq AC^\sigma_{(\Pi,<)}(C_2)$. Observe that a partial coloring C is extendible to a $<^\sigma$–preserving admissible one C', that is, $C \sqsubseteq C'$ iff $AC^\sigma_{(\Pi,<)}(C)$ is non-empty. For a total coloring C, $AC^\sigma_{(\Pi,<)}(C)$ is either empty or singleton. Regarding program $(\Pi_1, <)$ and coloring C_2, we get

$$AC^\sigma_{(\Pi_1,<)}(C_2) = AC^\sigma((\Pi_1, <)) = \{(\{r_1, r_2, r_4\}, \{r_3\})\}, \tag{3}$$

for $\sigma \in \{D, B, W\}$. Accordingly, define $AS^\sigma_{(\Pi,<)}(C)$ as the set of all $<^\sigma$–preserving answer sets X of $(\Pi, <)$ compatible with partial coloring C:

$$AS^\sigma_{(\Pi,<)}(C) = \{X \in AS^\sigma((\Pi, <)) \mid C_\oplus \subseteq R_\Pi(X) \text{ and } C_\ominus \cap R_\Pi(X) = \emptyset\}.$$

Note that $head(C_\oplus) \subseteq X$ for any $<^\sigma$–preserving answer set $X \in AS_\Pi(C)$. As regards program $(\Pi_1, <)$ and coloring C_2, we get

$$AS^\sigma_{(\Pi_1,<)}(C_2) = \{\{b, p, f'\}\}.$$

We call a coloring simply *admissible*, if X is a standard answer set of Π. Also, if X is a standard answer set of Π, we omit the superscript σ in the above defined sets.

We need the following concepts for describing a rule's status of applicability.

Definition 3. *Let $\Gamma = (\Pi, E_0, E_1, E_2)$ be the DG of ordered program $(\Pi, <)$ and C be a partial coloring of Γ. For $r \in \Pi$, we define:*

1. *r is supported in (Γ, C), if $body^+(r) \subseteq \{head(r') \mid (r', r) \in E_0, r' \in C_\oplus\}$;*
2. *r is unsupported in (Γ, C), if $\{r' \mid (r', r) \in E_0, head(r') = q\} \subseteq C_\ominus$ for some $q \in body^+(r)$;*
3. *r is blocked (by r') in (Γ, C), if $r' \in C_\oplus$ for some $(r', r) \in E_1$;*
4. *r is unblocked in (Γ, C), if $r' \in C_\ominus$ for all $(r', r) \in E_1$;*
5. *r is maximal in (Γ, C) if $\{r' \mid (r', r) \in E_2\} \subseteq (C_\oplus \cup C_\ominus)$.*

Conditions 1–4 express standard concepts of logic programming adapted to DGs (cf. [10]). The concept expressed in Condition 5 allows for distinguishing rules, all of which more preferred rules have either been found to be applicable or blocked. Such rules are *maximal* insofar as they are not dominated by any preferred rules having an undecided status of applicability. In what follows, we use $S(\Gamma, C), \overline{S}(\Gamma, C), B(\Gamma, C), \overline{B}(\Gamma, C)$, and $M(\Gamma, C)$ for denoting the sets of all supported, unsupported,blocked,unblocked, and maximal rules in (Γ, C). For illustration, consider the sets obtained regarding $(\Gamma_{(\Pi_1,<)}, C_2)$, given in Figure 1b.

$$\begin{array}{ll} S(\Gamma_{(\Pi_1,<)}, C_2) = \{r_1, r_2, r_3, r_4\} & \overline{S}(\Gamma_{(\Pi_1,<)}, C_2) = \emptyset \\ B(\Gamma_{(\Pi_1,<)}, C_2) = \emptyset & \overline{B}(\Gamma_{(\Pi_1,<)}, C_2) = \{r_1, r_2, r_4\} \\ M(\Gamma_{(\Pi_1,<)}, C_2) = \{r_1, r_2, r_4\} & \end{array} \quad (4)$$

Rule r_3 is not maximal in $(\Gamma_{(\Pi_1,<)}, C_2)$ because the higher preferred rule r_4 is uncolored and thus not known to be blocked or applied.

4 Deciding preferred answersetship from colored graphs

We now develop concepts that allow us to decide whether a (total) coloring represents an order preserving admissible coloring by purely graph-theoretical means. For this purpose, we build upon the concept of a *support graph* and a corresponding characterization proposed in [10]. To begin with, we give the definition of a support graph, accounting for the notion of recursive support; it is adapted to ordered programs.

Definition 4. *Let Γ be the DG of ordered logic program $(\Pi, <)$ and C be a partial coloring of Γ. We define a support graph of (Γ, C) as an acyclic 0-subgraph (V, E) of Γ such that $body^+(r) \subseteq \{head(r') \mid (r', r) \in E\}$ for all $r \in V$, $C_\oplus \subseteq V$ and $C_\ominus \cap V = \emptyset$.*

Observe that the order $<$ does not influence the support graph. If (Γ, C) has a support graph, then there is also a *maximal* support graph (V, E) of (Γ, C) such that $V' \subseteq V$ for all support graphs (V', E') of (Γ, C).

We build upon the following characterization of admissible colorings (along with their underlying answer sets) for standard logic programs, taken from [10].

Theorem 1. *Let Γ be the DG of logic program $(\Pi, \emptyset)$ and C be a total coloring of Γ. Then, C is an admissible coloring of Γ iff $C_\oplus = S(\Gamma, C) \cap \overline{B}(\Gamma, C)$ and $(C_\oplus, E)$ is a support graph of (Γ, C) for some $E \subseteq (\Pi \times \Pi)$.*

For illustration, let us consider program $(\Pi_1, \emptyset)$. We obtain the admissible colorings

$$AC((\Pi_1, \emptyset)) = \{(\{r_1, r_2, r_3\}, \{r_4\}), (\{r_1, r_2, r_4\}, \{r_3\})\}$$

representing answer sets

$$AS(\Pi_1) = \{\{p, b, f\}, \{p, b, f'\}\}.$$

For capturing preferences, we propose the concept of a *height function*. To begin with, we develop this concept for D–preferences.

Definition 5. *Let Γ be the DG of ordered logic program $(\Pi, <)$, C be a total coloring of Γ and let (V, E_0, E_1, E_2) be a subgraph of Γ.*

We define a D–height function of (V, E_0, E_1, E_2) as a function $h : V \to I\!N$ such that for all $r \in \Pi$, we have

1. *$h(r') < h(r)$ if $(r', r) \in E_2$,*
2. *if $r \in C_\oplus$ then we have $h(r') < h(r)$ if $(r', r) \in E_0$ and $r' \in C_\oplus$, and*
3. *if $r \in C_\ominus \cap V$ then there exists an $r' \in C_\oplus$ such that $(r', r) \in E_1$ and $h(r') < h(r)$.*

The values attributed by a height function reflect a possible order of rule consideration (not necessarily application). In this respect, Condition 1 stipulates that higher ranked rules must be considered before lower ranked rules; in this way, h respects the preferences from $<$. If (V, E_0) forms a support graph of (Γ, C), then Condition 2 ensures that rules are never supported by rules having a greater h-value. Condition 3 expresses that rules colored with $\ominus$, must be blocked by rules with a smaller h-value (that is, intuitively, already applied rules). It is instructive to observe that every height function induces a partial order on Π extending the given partial order $<$. Furthermore, this induced order is always extendible to a total order of Π respecting an enumeration of the generating rules, given in Definition 1. A more detailed analysis is given in the full paper [11].

Taking the concept of a D–height function together with Theorem 1, we obtain a characterization of $<^D$–preserving admissible colorings.

Theorem 2. *Let $\Gamma = (\Pi, E_0, E_1, E_2)$ be the DG of ordered logic program $(\Pi, <)$ and C be a total coloring of Γ. Then, C is a $<^D$–preserving admissible coloring iff*

1. *$C_\oplus = S(\Gamma, C) \cap \overline{B}(\Gamma, C)$ and*
2. *for some $E_0' \subseteq E_0$, we have*

(a) $(C_\oplus, E_0')$ *is a support graph of* (Γ, C) *and*
(b) there exists a D*–height function of* $(S(\Gamma, C), E_0', E_1|_{S(\Gamma,C)}, E_2|_{S(\Gamma,C)})$.

Conditions 1 and 2a are the ones found in Theorem 1 for standard admissible colorings, while Condition 2b selects the $<^D$–preserving ones by means of a D–height function. For this, only supported rules are taken into account; unsupported rules are inapplicable anyway. Now, the height function ties the arcs E_0' of the (standard) support graph to the ones reflecting blockage $E_1|_{S(\Gamma,C)}$ and preference $E_2|_{S(\Gamma,C)}$. This guarantees that the underlying answer set can only be formed in an order preserving way.

For illustration, consider $(\Pi_1, <)$. For the admissible coloring $(\{r_1, r_2, r_4\}, \{r_3\})$, we detect only the following D–height function h_D:

$$h_D(r_1) = 1,\ h_D(r_2) = 2,\ h_D(r_3) = 4,\ h_D(r_4) = 3. \tag{5}$$

For admissible coloring $(\{r_1, r_2, r_3\}, \{r_4\})$, there is no D–height function because $r_3 < r_4$ and the blockage of r_4 by r_3 lead to a contradiction between Condition 1 and 3 in Definition 5. Hence, only $(\{r_1, r_2, r_4\}, \{r_3\})$ is $<^D$–preserving and $\{p, b, f'\}$ is the only $<^D$–preserving answer set.

For B– and W–preferences, we can define similar height functions in an analogous way.

Definition 6. *Let* Γ *be the DG of ordered logic program* $(\Pi, <)$, C *be a total coloring of* Γ *and let* (V, E_0, E_1, E_2) *be a subgraph of* Γ.

We define a B*–height function of* (V, E_0, E_1, E_2) *as a function* $h : V \to I\!N$ *such that for all* $r \in \Pi$ *we have*

1. $h(r') < h(r)$ *if* $(r', r) \in E_2$,
2. *if* $r \in C_\ominus \cap V$ *one of the following conditions is fulfilled:*
   (a) there exists $r' \in C_\oplus$ *such that* $(r', r) \in E_1$ *and* $h(r') < h(r)$*;*
   (b) $rule(head(r)) \cap C_\oplus \neq \emptyset$

Note that a B–height function does not take into account 0-edges, since B–preference decouples supportedness from preference handling [3]. If $X = head(C_\oplus)$ is a set of atoms, then $rule(head(r)) \cap C_\oplus \neq \emptyset$ states that $head(r) \in X$ for some $r \in \Pi$. Hence, Condition 2b weakens the concept of order preservation given in Condition 3 of Definition 5, whenever the head of a blocked rule is derived by another applied rule. By this weakening, more admissible colorings are $<^B$–preserving than $<^D$–preserving.

The next definition addresses W–preferences.

Definition 7. *Let* Γ *be the DG of ordered logic program* $(\Pi, <)$, C *be a total coloring of* Γ *and let* (V, E_0, E_1, E_2) *be a subgraph of* Γ.

We define a W*–height function of* (V, E_0, E_1, E_2) *as a function* $h : V \to I\!N$ *such that for all* $r \in \Pi$ *we have*

1. $h(r') < h(r)$ *if* $(r', r) \in E_2$,
2. *if* $r \in C_\oplus$ *then is one of the following conditions fulfilled:*
   (a) we have $h(r') < h(r)$ *if* $(r', r) \in E_0$ *and* $r' \in C_\oplus$
   (b) there exists an $r' \in rule(head(r)) \cap C_\oplus$ *such that* $h(r') < h(r)$,

3. *if* $r \in C_\ominus \cap V$ *then one of the following conditions is fulfilled:*
   (a) *there exists* $r' \in C_\oplus$ *such that* $(r', r) \in E_1$ *and* $h(r') < h(r)$*;*
   (b) *there exists an* $r'' \in rule(head(r)) \cap C_\oplus$ *such that* $h(r'') < h(r)$*.*

W–height functions combine supportedness and preference handling similar to D–height functions. In contrast to D–height functions, however, Definition 7 allows for supporting and blocking a rule r by lower ranked rules, if $head(r)$ is derived by some applied rule with a lower h-value than r. Hence, Condition 2b and 3b weaken the concept of order preservation given in Definition 5, but they are not so generous as the conditions for a B–height function given in Definition 6. For this reason, the conditions for the existence of a D–height function are stronger that than for a W–height function, which are stronger conditions than for a B–height function.

For illustration, consider ordered logic program $(\Pi_1, <)$. For admissible coloring $(\{r_1, r_2, r_4\}, \{r_3\})$, the D–height function given in (5) is also a B– as well as a W–height function. Observe that $h(r_1) = 2,\ h(r_2) = 1,\ h(r_3) = 4,\ h(r_4) = 3$ provides an alternative B–height function. No σ–height function is obtained for the second admissible coloring, corresponding to answer set $\{p, b, f\}$, for any $\sigma \in \{D, B, W\}$.

In analogy to to Theorem 2, B– and W–height functions allow us to characterize $<^B$– and $<^W$–preserving admissible colorings.

Theorem 3. *Theorem 2 still holds, when replacing* D *by either* B *or* W*.*

For illustration, consider program $(\Pi_1, <)$. As above, $(\{r_1, r_2, r_3\}, \{r_4\})$ is neither $<^B$– nor $<^W$–preserving since $head(r_4)$ is not derivable in an alternative way. Hence, $(\{r_1, r_2, r_4\}, \{r_3\})$ is a $<^\sigma$–preserving admissible coloring for $\sigma \in \{D, B, W\}$.

Whenever no preferences are given, Theorem 2 and 3 fall back to characterizations of standard admissible colorings:

Corollary 1. *Let* Γ *be the* DG *of ordered logic program* $(\Pi, \emptyset)$ *and* C *be a total coloring of* Γ*. Then,* C *is an admissible coloring iff* C *is a* $<^\sigma$*–preserving admissible coloring for* $\sigma \in \{D, B, W\}$*.*

5 Operational characterization

In this section, we exemplarily provide an operational characterization of $<^D$–preserving answer sets; for brevity, the corresponding characterizations for B– and W–preferences are omitted. The idea is to start with the empty coloring and to successively apply operators that turn a partial coloring C into another one C' such that $C \sqsubseteq C'$. This is done until a total coloring is obtained that corresponds to a $<^D$–preserving answer set.

For this, it is necessary to introduce a new color $\oslash$, which only appears in intermediate partial colorings. That is, a partial coloring is now a partial mapping $C : \Pi \to \{\oplus, \ominus, \oslash\}$. Analogously, we define $C_\oslash = \{r \in \Pi \mid C(r) = \oslash\}$. We color a vertex r with $\oslash$ in a partial coloring C, if r must be colored with $\ominus$ in the final total coloring but there is not yet any justification for coloring r with $\ominus$. As before, C is a total coloring if $C_\oplus \cup C_\ominus = \Pi$ and $C_\oslash = \emptyset$. Furthermore, $C \sqsubseteq C'$ if $C_\oplus \subseteq C'_\oplus$, $C_\ominus \subseteq C'_\ominus$ and $C_\oslash \subseteq C'_\ominus \cup C'_\oslash$. We define $C \sqcup C'$ as $(C_\oplus \cup C'_\oplus, C_\ominus \cup C'_\ominus, (C_\oslash \cup C'_\oslash) \setminus (C_\ominus \cup C'_\ominus))$. We denote the set of all partial colorings of a DG $\Gamma_{(\Pi,<)}$ by $\mathbb{C}_{\Gamma_{(\Pi,<)}}$. Whenever clear

from the context, we simply write $\mathbb{C}$. Otherwise, all concepts from the previous sections directly carry over, since they keep relying on rules belonging to $C_\oplus$ and $C_\ominus$ only.

We concentrate first on operations deterministically extending partial colorings.

Definition 8. *Let Γ be the DG of ordered logic program $(\Pi, <)$ and C be a partial coloring of Γ. Then, define $\mathcal{P}_\Gamma : \mathbb{C} \to \mathbb{C}$ as $\mathcal{P}_\Gamma(C) = C'$ where*

$C'_\oplus = C_\oplus \cup (S(\Gamma, C) \cap \overline{B}(\Gamma, C) \cap M(\Gamma, C))$,
$C'_\ominus = C_\ominus \cup \overline{S}(\Gamma, C) \cup (B(\Gamma, C) \cap M(\Gamma, C))$, *and*
$C'_\oslash = C_\oslash \setminus C'_\ominus$.

For standard logic programs Π, $\mathcal{P}_\Gamma$ is defined in [10] by means of standard colors $C_\oplus$ and $C_\ominus$ only. Definition 8 thus offers an extension of operator $\mathcal{P}_\Gamma$, augmented by a third color for dealing with ordered logic programs. A coloring is extended by maximal rules only, with the exception of unsupported rules (cf. Condition 3a in Definition 1). The idea is to propagate along a D–height function, while excluding unsupported rules and coloring them with $\ominus$.

A partial coloring C is closed under $\mathcal{P}_\Gamma$, if $C = \mathcal{P}_\Gamma(C)$. Note that $C \sqsubseteq \mathcal{P}_\Gamma(C)$. In fact, $\mathcal{P}_\Gamma(C)$ is not guaranteed to be a partial coloring. To see this, observe that $\mathcal{P}_\Gamma((\{a \leftarrow not\ a\}, \emptyset, \emptyset))$ would be $(\{a \leftarrow not\ a\}, \{a \leftarrow not\ a, \emptyset\})$, which is no mapping and thus no partial coloring. Interestingly, $\mathcal{P}_\Gamma$ exists on colorings expressing preferred answer sets (cf. Theorem 4 below). Now, we can define our principal propagation operator in the following way.

Definition 9. *Let Γ be the DG of ordered logic program $(\Pi, <)$ and C a partial coloring of Γ. Then, define $\mathcal{P}^*_\Gamma(C)$ as the $\sqsubseteq$-smallest partial coloring closed under $\mathcal{P}_\Gamma$ and containing C.*

Although $\mathcal{P}^*_\Gamma$ is not always defined, it is on colorings expressing preferred answer sets.

Theorem 4. *Let Γ be the DG of ordered logic program $(\Pi, <)$ and C a partial coloring of Γ. If $AC^D_{(\Pi,<)}(C) \neq \emptyset$, then $\mathcal{P}^*_\Gamma(C)$ exists.*

Essentially, $\mathcal{P}^*_\Gamma(C)$ amounts to computing the deterministic "consequences" of a given partial coloring C. In fact, $\mathcal{P}^*_\Gamma(C)$ is monotonic and preserves preferred answer sets in the following sense.

Theorem 5. *Let Γ be the DG of ordered logic program $(\Pi, <)$ and C be a partial coloring of Γ.*

1. *If $AC^D_{(\Pi,<)}(C') \neq \emptyset$ and $C \sqsubseteq C'$, then $\mathcal{P}^*_\Gamma(C) \sqsubseteq \mathcal{P}^*_\Gamma(C')$;*
2. *$AC^D_{(\Pi,<)}(C) = AC^D_{(\Pi,<)}(\mathcal{P}^*_\Gamma(C))$.*

In [10], it is shown for standard programs that $\mathcal{P}^*_\Gamma$ amounts to Fitting's operator [7]. Therefore, Definition 9 can be viewed as an extension of Fitting's operator to ordered programs. The next operation draws upon the maximal support graph of colored DGs.

Definition 10. *Let Γ be the DG of ordered logic program $(\Pi, <)$ and C be a partial coloring of Γ. Furthermore, let (V, E) be a maximal support graph of (Γ, C) for some $E \subseteq (\Pi \times \Pi)$. Then, define $\mathcal{U}_\Gamma : \mathbb{C} \to \mathbb{C}$ as*

$$\mathcal{U}_\Gamma(C) = (C_\oplus, \Pi \setminus V, C_\oslash \cap V).$$

A 2-ary version of $\mathcal{U}_\Gamma$ was proposed in [10] for standard programs. This operator allows for coloring rules with $\ominus$ whenever it is clear from the given partial coloring that they will remain unsupported. Observe that $C_\ominus \subseteq \Pi \setminus V$ and $C_\oslash \cap V = C_\oslash \setminus (\Pi \setminus V)$. As with $\mathcal{P}^*_\Gamma$, operator $\mathcal{U}_\Gamma(C)$ is an extension of C. Unlike $\mathcal{P}^*_\Gamma$, however, $\mathcal{U}_\Gamma$ allows for coloring nodes unconnected with the already colored part of the graph. Although $\mathcal{U}_\Gamma$ is not defined in general, it is on colorings guaranteeing the existence of support graphs.

Theorem 6. *Let Γ be the DG of ordered logic program $(\Pi, <)$ and C be a partial coloring of Γ. If (Γ, C) has a support graph, then $\mathcal{U}_\Gamma(C)$ exists.*

We show in the full paper [11] that $\mathcal{U}_\Gamma$ is reflexive, idempotent, monotonic, and preferred answer set preserving. That is, for partial colorings C and C' of Γ such that $AC^D_\Pi(C) \neq \emptyset$ and $AC^D_\Pi(C') \neq \emptyset$, we have $C \sqsubseteq \mathcal{U}_\Gamma(C)$, $\mathcal{U}_\Gamma(C) = \mathcal{U}_\Gamma(\mathcal{U}_\Gamma(C))$, and if $C \sqsubseteq C'$, then $\mathcal{U}_\Gamma(C) \sqsubseteq \mathcal{U}_\Gamma(C')$. Moreover, we have $AC^D_\Pi(C) = AC^D_\Pi(\mathcal{U}_\Gamma(C))$. Note that unlike $\mathcal{P}_\Gamma$, $\mathcal{U}_\Gamma$ leaves the support graph of (Γ, C) unaffected.

Now we develop a strategy for choice operations based on supported of rules.

Definition 11. *Let $\Gamma = (\Pi, E_0, E_1, E_2)$ be the DG of ordered logic program $(\Pi, <)$ and C be a partial coloring of Γ. For $r \in (\Pi \cap M(\Gamma, C)) \setminus (C_\oplus \cup C_\ominus)$, we define the following operators $\mathcal{D}^\circ_\Gamma : \mathbb{C} \to \mathbb{C}$ for $\circ \in \{\oplus, \ominus\}$:*

1. $\mathcal{D}^\oplus_\Gamma(C) = (C_\oplus \cup \{r\}, C_\ominus, C_\oslash)$, *if $r \in S(\Gamma, C)$;*
2. $\mathcal{D}^\ominus_\Gamma(C) = (C_\oplus, C_\ominus \cup \{r\}, ((C_\oslash \cup R_p) \setminus \{r\})$, *if $r \notin S(\Gamma, C) \cup \overline{S}(\Gamma, C)$, where non-empty $R_p = rule(p) \setminus C_\ominus$ for some $p \in body^+(r)$.*

The $\mathcal{D}^\oplus_\Gamma$ operator colors a maximal, supported rule r with $\oplus$, that is, r is taken to be applied. $\mathcal{D}^\ominus_\Gamma$ colors a maximal, up to now neither supported nor unsupported rule with $\ominus$. That is, r must be unsupported and thus belong to $C_\ominus$ in the final total coloring. For guaranteeing this, it must be ensured that for some atom $p \in body^+(r)$ all rules r' with $head(r') = p$ are eventually inapplicable. This is accomplished by coloring all rules in R_p with $\oslash$. In this way, p is excluded from all preferred answer sets obtained from C. Observe that blocked rules are never colored $\ominus$ by $\mathcal{D}^\ominus_\Gamma$ in order to guarantee Condition 3 in Definition 5. Similarly, unsupported rules are taken care of by propagation operations. For illustration, consider $(\Pi_6, <)$, where:

$$\begin{array}{ll} r_1 : a \leftarrow & r_1 < r_2 \\ r_2 : b \leftarrow not\ \ \mathrm{a} & \end{array} \tag{6}$$

We obtain $\mathcal{P}_\Gamma((\emptyset, \emptyset, \emptyset)) = (\emptyset, \emptyset, \emptyset)$. If $\mathcal{D}^\ominus_\Gamma$ would color maximal, (supported,) blocked rules, then $\mathcal{D}^\ominus_\Gamma((\emptyset, \emptyset, \emptyset)) = (\emptyset, \{r_2\}, \emptyset)$. But then, $\mathcal{P}_\Gamma$ would not detect the miscoloring $\mathcal{P}_\Gamma((\emptyset, \{r_2\}, \emptyset)) = (\{r_1\}, \{r_2\}, \emptyset)$ which is an admissible coloring but no $<^D$–preserving one. That is, there is no corresponding $<^D$–preserving answer set.

The above operator takes preference into account in two ways. First, it restricts the choice of r to rules belonging to $M(\Gamma, C)$. Second, it eliminates the coloration of blocked rules and delegates it to the deterministic operator $\mathcal{P}_\Gamma$. While both measures clearly restrict the possible number of overall choices, a further non-determinism is created in Condition 2 in Definition 11 through the choice of $p \in body^+(r)$. Interestingly,

the choice of p has further repercussions since it amounts to excluding all rules r with $head(r) = p$ from $C_{\oplus}$. That is, whenever such a choice is made further propagation follows, which leads to a more constrained situation. We are currently investigating the effect of this in an experimental study.

Combining our deterministic operators with the choice operator yields an operational characterization of order preserving admissible colorings. For this, for a partial coloring C we define $(\mathcal{PU})^*_\Gamma(C)$ as the $\sqsubseteq$- smallest partial coloring containing C and being closed under $\mathcal{P}_\Gamma$ and $\mathcal{U}_\Gamma$.

Theorem 7. *Let Γ be the DG of ordered logic program $(\Pi, <)$ and let C be a total coloring of Γ. Then, C is a $<^D$–preserving admissible coloring of Γ iff there exists a sequence $(C^i)_{0 \leq i \leq n}$ with the following properties:*

1. $C^0 = (\mathcal{PU})^*_\Gamma((\emptyset, \emptyset, \emptyset))$
2. $C^{i+1} = (\mathcal{PU})^*_\Gamma(\mathcal{D}^\circ_\Gamma(C^i))$ *for some* $\circ \in \{\oplus, \ominus\}$ *and* $0 \leq i < n$;
3. $C^n = C$.

The formation of sequences is driven by the coloration of maximal rules. Operators $\mathcal{P}_\Gamma, \mathcal{U}_\Gamma$ and $\mathcal{D}^\circ_\Gamma$ color along a D–height function, where lower valued rules are colored first. That is, the sequence starts by coloring most preferred rules and ends with the lowest ones.

For illustration, consider the coloring sequence in Figure 2, obtained for $<^D$–preserving answer set $\{b, p, f'\}$ of program $(\Pi_1, <)$. First, maximal rules r_1 and r_2 are

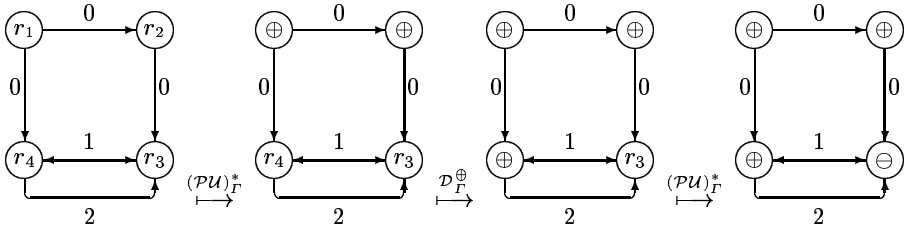

Fig. 2. A coloring sequence.

colored by $(\mathcal{PU})^*_\Gamma$. From the remaining uncolored rules, only r_4 is maximal and (since being supported) taken to be $\oplus$ by $\mathcal{D}^\oplus_\Gamma$. This leads to coloring r_3 by $(\mathcal{PU})^*_\Gamma$ since it is maximal and blocked. The resulting admissible coloring $(\{r_1, r_2, r_4\}, \{r_3\})$ is $<^D$–preserving and reflects the $<^D$–preserving answer set $\{p, b, f'\}$. Note that coloring a maximal, supported and blocked rule r in the absence of a blocker of r would lead to illegal total colorings, which are admissible but not order preserving. For example, coloring r_4 with $\ominus$ instead of $\oplus$ after $(\mathcal{PU})^*_\Gamma$ would lead to coloring r_3 with $\oplus$ by $\mathcal{P}_\Gamma$. The resulting total coloring $(\{r_1, r_2, r_3\}, \{r_4\})$ is admissible, but not $<^D$– preserving.

To illustrate the usage of $\oslash$, consider the following program $(\Pi_7, <)$, where

$$\begin{array}{ll} r_1 : a \leftarrow c & \\ r_2 : b \leftarrow not\ c & r_2 < r_1 \\ r_3 : c \leftarrow not\ b & r_3 < r_2 \end{array} \qquad (7)$$

$(\mathcal{PU})^*_\Gamma$ cannot color any rule. Only r_1 is maximal and available for our choice operator. By $r \notin S(\Gamma_{(\Pi_7,<)}, (\emptyset,\emptyset,\emptyset))$, we color r_1 by $\mathcal{D}^{\ominus}_\Gamma$, which leads to coloring $(\emptyset, \{r_1\}, \{r_3\}) = (\mathcal{PU})^*_\Gamma((\emptyset, \{r_1\}, \{r_3\}))$. Applying $\mathcal{D}^{\oplus}_\Gamma$ to maximal rule r_2 and applying $(\mathcal{PU})^*_\Gamma$ lead to total coloring $(\{r_2\}, \{r_1, r_3\}, \emptyset)$, which is $<^D$–preserving and corresponds to the only existing $<^D$–preserving answer set $\{b\}$. The successful coloring sequence is given in Figure 3. Note that the *DG* contains the 2-edge (r_1, r_3) since $<$ is transitive.

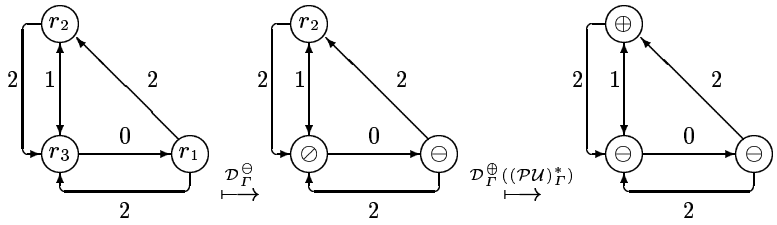

Fig. 3. A (successful) coloring sequence.

In the full paper [11], other sequences of partial colorings leading to $<^D$–preserving admissible colorings are considered. For example, starting with the empty coloring $C^0 = (\emptyset,\emptyset,\emptyset)$ and obtaining C^{i+1} by $\mathcal{P}^*_\Gamma(\mathcal{D}^\circ_\Gamma(C^i))$, thus disposing of $\mathcal{U}_\Gamma$. In this way, the coloration of unsupported rules is entirely accomplished by $\mathcal{D}^{\ominus}_\Gamma$. Although this avoids using (deterministic) operator $\mathcal{U}_\Gamma$, it delegates the treatment of unsupported rules to a non-deterministic operator, which seems not advisable from a computational point of view. Furthermore, we discuss an alternative support-driven operational characterization using an incremental version of $\mathcal{U}_\Gamma$.

6 Discussion, related work, and conclusions

Many approaches to adding preferences to answer set programming can be found in the literature [16, 2, 8, 19, 9, 3, 5, 18]. Among them, we have chosen the three approaches, interpreting preferences as inducing a selection function among the answer sets of the underlying program [3, 5, 17]. Up to now, the latter approaches have either been implemented by meta-interpretation [6] or by pre-compilation front-ends [5]. The advantage of both approaches is that one can harness existing answer sets solvers without any need for modification. On the other hand, it remains unclear whether the "selection of answer sets" cannot be realized more efficiently within a solver by restricting its search space. For instance, "weight-based" approaches, as pursed in the `dlv` [12] and `smodels` [15] systems, can be implemented rather efficiently through branch-and-bound techniques. Such a quantitative approach is unfortunately inapplicable in our setting.

For addressing this problem, we have put forward the usage of graphs and colorings as an appropriate computational model. Preferences are simply taken as a third type of edges in a graph, reflecting an additional dependency among rules. In particular, we have demonstrated that this approach allows us to capture all three "selection

function" approaches to preferences in a uniform setting by means of the concept of a height function. To a turn, we have exemplarily developed an operational characterization for one of these strategies. For this purpose, we have extended a recently proposed operational framework for graph-based computation of answer sets [10]. Apart from the extension of colorings by a third "transitory" coloring $\oslash$ (comparable to `dlv`'s "`must_be_true`"), we have extended the deterministic and non-deterministic operations by preference handling. This is done through the restriction of propagation and choice operations to those rules that are not dominated by any preferred rules whose application status is indeterminate (viz. $M(\Gamma, C)$). We have prototypically implemented different operational variants, using different operators; the resulting Prolog implementation is available at

`http://www.cs.uni-potsdam.de/~konczak/system/GCplp`.

An integration into the `noMoRe` system is envisaged in the near future.

Acknowledgements

The authors were partially supported by the German Science Foundation (DFG) under grant FOR 375/1 and SCHA 550/6, TP C and they were partially funded by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2001-37004 WASP project.

References

1. C. Anger, K. Konczak, and T. Linke. `noMoRe`: Non-monotonic reasoning with logic programs. In S. Flesca et al., editors, *Proceedings of the Eighth European Conference on Logics in Artificial Intelligence (JELIA'02)*, pages 521–524. Springer, 2002.
2. G. Brewka. Well-Founded Semantics for Extended Logic Programs with Dynamic Preferences. *Journal of Artificial Intelligence Research*, 4:19-36, 1996.
3. G. Brewka and T. Eiter. Preferred Answer Sets for Extended Logic Programs. *Artificial Intelligence*, 109(1-2):297-356, 1999.
4. G. Brignoli, S. Costantini, O. D'Antona, and A. Provetti. Characterizing and computing stable models of logic programs: the non-stratified case. In C. Baral and H. Mohanty, editors, *Proceedings of the Conference on Information Technology, Bhubaneswar, India*, pages 197–201. AAAI Press, 1999.
5. J. Delgrande, T. Schaub and H. Tompits. A Framework for Compiling Preferences in Logic Programs. *Theory and Practice of Logic Programming*, 3(2):129-187, March 2003.
6. T. Eiter, W. Faber, N. Leone, G. Pfeifer. Computing preferred answer sets by meta-interpretation in answer set programming. *Theory and Practice of Logic Programming*, 3(4-5):463-498, 2003.
7. M. Fitting. Fixpoint semantics for logic programming a survey. *Theoretical Computer Science*, 278(1-2):25–51, 2002.
8. M. Gelfond and T. Son. Reasoning with Prioritized Defaults. *Third International Workshop on Logic Programming and Knowledge Representation*, volume 1471 of *lecture Notes in Computer Science*, pages 164-223. Springer-Verlag, 1997.
9. B. Grosof. Prioritized Conflict Handling for Logic Programs. In J. Maluszynsk, editor, *Logic Programming: Proceedings of the 1997 International Symposium*, pages 197-211. The MIT Press, 1997.

10. K. Konczak, T. Linke and T. Schaub. Graphs and colorings for answer set programming: Abridged Report, 2003. To appear.
11. K. Konczak, T. Linke and T. Schaub. Graphs and colorings for answer set programming with preferences, 2003. In preparation.
12. N. Leone, W. Faber, G. Pfeifer, T. Eiter, G. Gottlob, C. Koch, C. Mateis, S. Perri and F. Scarcello. The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic*, 2003. To appear.
13. T. Linke. Graph theoretical characterization and computation of answer sets. In B. Nebel, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 641–645. Morgan Kaufmann Publishers, 2001.
14. J. Lloyd. *Foundations of Logic Programming*. Springer, 1987.
15. I. Niemelä, P. Simons and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2):181-234, 2002.
16. C. Sakama and K. Inoue. Prioritized Logic Programming and its Application to Commonsense Reasoning. *Artificial Intelligence*, 123(1-2):185-222, 2000.
17. T. Schaub and K. Wang. A semantic framework for preference handling in answer set programming. *Theory and Practice of Logic Programming*, 3(4-5):569-607, 2003.
18. K. Wang, L. Zhou and F. Lin. Alternating fixpoint theory for logic programs with priority. In *Proceedings of the First International Conference on Computational Logic*, volume 1861 of *Lecture Notes in Computer Science*, pages 164-178. Springer-Verlag, 2000.
19. Y. Zhang. Two results for prioritized logic programming. In *Theory and Practice of Logic Programming*, 3(2):223-242, 2003.