Graphs and colorings for answer set programming: Abridged Report

Kathrin Konczak, Thomas Linke, and Torsten Schaub

Institut für Informatik, Universität Potsdam, Postfach 90 03 27, D–14439 Potsdam

Abstract. We investigate rule dependency graphs and their colorings for characterizing the computation of answer sets of logic programs. We start from a characterization of answer sets in terms of totally colored dependency graphs. To a turn, we develop a series of operational characterizations of answer sets in terms of operators on partial colorings. In analogy to the notion of a derivation in proof theory, our operational characterizations are expressed as (non-deterministically formed) sequences of colorings, turning an uncolored graph into a totally colored one. This results in an operational framework in which different combinations of operators result in different formal properties. Among others, we identify the basic strategy employed by the `noMoRe` system and justify its algorithmic approach. Also, we distinguish Fitting's and well-founded semantics.

1 Introduction

Graphs constitute a fundamental tool within computing science. Similarly, in *answer set programming* [8] graphs are used for deciding whether answer sets exist [6]. We take the application of graphs further and elaborate upon using graphs as the underlying computational model for computing answer sets. To this end, we build upon the theoretical foundations introduced in [9, 1]. Accordingly, we are interested in characterizing answer sets by means of their set of generating rules. For determining, whether a rule belongs to this set, we must verify that each positive body atom is derivable and that no negative body atom is derivable. In fact, an atom is derivable, if the set of generating rules includes a rule, having the atom as its head; or conversely, an atom is not derivable, if there is no rule among the generating rules that has the atom as its head. Consequently, the formation of the set of generating rules boils down to resolving positive and negative dependencies among rules. For capturing these dependencies, we take advantage of the concept of a *rule dependency graph*, wherein each node represents a rule of the underlying program and two types of edges stand for the aforementioned positive and negative rule dependencies, respectively. For expressing the applicability status of rules, that is, whether a rule belongs to a set of generating rules or not, we *color* the respective nodes in the graph. In this way, an answer set can be expressed by a total coloring of the rule dependency graph. Of course, in what follows, we are mainly interested in the inverse, that is, when does a graph coloring correspond to an answer set of the underlying program; and, in particular, how can we compute such a total coloring. To this end, we start by identifying graph structures that allow for characterizing answer sets in terms of totally colored dependency graphs. We then build upon these characterizations for developing an operational framework for answer set formation. The 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 an answer set. This approach is strongly inspired by the concept of a derivation, in particular, that of an SLD-derivation [15]. Accordingly, a program has a certain answer set iff there is a sequence of operations turning the uncolored graph into a totally colored one, expressing the answer set.

All proofs can be found in the full version of this abridged report [12].

2 Rules, programs, graphs, and colorings

A *logic program* is a finite set of rules such as $p_0 \leftarrow p_1, \dots, p_m, \mathit{not}\ p_{m+1}, \dots, \mathit{not}\ p_n$, 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 $\mathit{head}(r)$ denote the *head*, p_0, of r and $\mathit{body}(r)$ the *body*, $\{p_1, \dots, p_m,\ \mathit{not}\ p_{m+1}, \dots, \mathit{not}\ p_n\}$, of r. Let $\mathit{body}^+(r) = \{p_1, \dots, p_m\}$ and $\mathit{body}^-(r) = \{p_{m+1}, \dots, p_n\}$. A program is *basic* if $\mathit{body}^-(r) = \emptyset$ for all its rules. The *reduct*, Π^X, of a program Π *relative to* a set X of atoms is defined by $\Pi^X = \{\mathit{head}(r) \leftarrow \mathit{body}^+(r) \mid r \in \Pi, \mathit{body}^-(r) \cap X = \emptyset\}$. A set of atoms X is *closed under* a basic program Π if for any $r \in \Pi$, $\mathit{head}(r) \in X$ if $\mathit{body}^+(r) \subseteq X$. The smallest set of atoms being closed under a basic program Π is denoted by $\mathit{Cn}(\Pi)$. Then, a set X of atoms is an *answer set* of a program Π if $\mathit{Cn}(\Pi^X) = X$. We use $\mathit{AS}(\Pi)$ 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

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

Next, we lay 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 graphs reflecting dependencies among rules.

Definition 1. *Let Π be a logic program. The rule dependency graph (RDG) $\Gamma_\Pi = (\Pi, E_0, E_1)$ of Π is a labeled directed graph with*

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

We omit the subscript Π from Γ_Π whenever the underlying program is clear from the context. An i-subgraph (V, E) of Γ is a subgraph of Γ with $E \subseteq E_i$ for $i \in \{0, 1\}$.

For illustration, consider the logic program $\Pi_1 = \{r_1, \dots, r_6\}$, where

$$\begin{array}{lll} r_1 : p \leftarrow & r_3 :\ f \leftarrow b, \mathit{not}\ f' & r_5 : b \leftarrow m \\ r_2 : b \leftarrow p & r_4 : f' \leftarrow p, \mathit{not}\ f & r_6 : x \leftarrow f, f', \mathit{not}\ x \end{array} \qquad (1)$$

The *RDG* of Π_1 is depicted graphically in Figure 1a. The *RDG* Γ_{Π_1} has among others

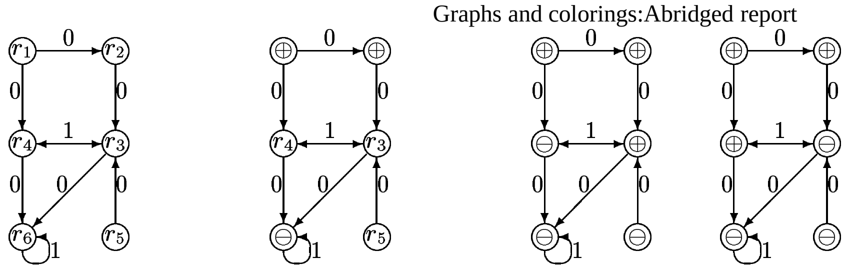

Fig. 1. (a) The *RDG* of logic program Π_1; **(b)** The (partially) colored *RDG* (Γ_{Π_1}, C_2); **(c+d)** The totally colored *RDG*s (Γ_{Π_1}, C_{4a}) and (Γ_{Π_1}, C_{4b}).

0-subgraph $(\{r_1,\dots,r_4\},\{(r_1,r_2)\})$ and 1-subgraph $(\{r_5,r_6\},\{(r_6,r_6)\})$.

We call C a *coloring* of Γ_Π if C is a mapping $C : \Pi \to \{\oplus,\ominus\}$. We denote the set of all partial colorings of a *RDG* Γ_Π by $\mathbb{C}_{\Gamma_\Pi}$. For readability, we often omit the index Γ_Π. Intuitively, the colors $\oplus$ and $\ominus$ indicate whether a rule is supposedly applied or blocked. We define $C_\oplus = \{r \mid C(r) = \oplus\}$ and $C_\ominus = \{r \mid C(r) = \ominus\}$ for obtaining all vertices colored by C with $\oplus$ or $\ominus$. 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 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_Π, we call the pair (Γ_Π, C) a *colored RDG*. For example, "coloring" the *RDG* of Π_1 with

$$C_2 = (\{r_1, r_2\},\ \{r_6\}) \tag{2}$$

yields the colored graph 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 *RDG*s that correspond to the answer sets of an underlying program. In fact, the colorings of interest can be distinguished in a straightforward way. Given a logic program Π along with its *RDG* Γ. Then, for every answer set X of Π, define an *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 admissible colorings whose members are in one-to-one correspondence with its answer sets. Clearly, any admissible coloring is total; also, we have $X = \mathit{head}(C_\oplus)$. We use $AC(\Pi)$ for denoting the set of all admissible colorings of a *RDG* Γ_Π. For a partial coloring C, we define $AC_\Pi(C)$ as the set of all admissible colorings of Γ_Π compatible with C. Formally, given the *RDG* Γ of a logic program Π and a partial coloring C of Γ, define

$$AC_\Pi(C) = \{C' \in AC(\Pi) \mid C \sqsubseteq C'\}.$$

Clearly, $C_1 \sqsubseteq C_2$ implies $AC_\Pi(C_1) \supseteq AC_\Pi(C_2)$. Observe that a partial coloring C is extendible to an admissible one C', that is, $C \sqsubseteq C'$ iff $AC_\Pi(C)$ is non-empty. For a total coloring C, $AC_\Pi(C)$ is either empty or singleton. Regarding program Π_1 and coloring C_2, we get

$$AC_{\Pi_1}(C_2) = AC(\Pi_1) = \{(\{r_1, r_2, r_3\}, \{r_4, r_5, r_6\}), (\{r_1, r_2, r_4\}, \{r_3, r_5, r_6\})\}$$

as shown in Figure 1c+d. Accordingly, define $AS_\Pi(C)$ as the set of all answer sets X of Π compatible with partial coloring C:

$$AS_\Pi(C) = \{X \in AS(\Pi) \mid C_\oplus \subseteq R_\Pi(X) \text{ and } C_\ominus \cap R_\Pi(X) = \emptyset\}.$$

Note that $head(C_\oplus) \subseteq X$ for any answer set $X \in AS_\Pi(C)$. As regards program Π_1 and coloring C_2, we get

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

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

Definition 2. *Let $\Gamma = (\Pi, E_0, E_1)$ be the RDG of logic program Π 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 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 $(r', r) \in E_1$.*

In what follows, we use $S(\Gamma, C)$, $\overline{S}(\Gamma, C)$, $B(\Gamma, C)$, and $\overline{B}(\Gamma, C)$ for denoting the sets of all supported, unsupported, blocked, and unblocked rules in (Γ, C). For illustration, consider the sets obtained regarding the colored *RDG* (Γ_{Π_1}, C_2) in Figure 1b.

$$\begin{array}{ll} S(\Gamma_{\Pi_1}, C_2) = \{r_1, r_2, r_3, r_4\} & \overline{S}(\Gamma_{\Pi_1}, C_2) = \{r_5\} \\ B(\Gamma_{\Pi_1}, C_2) = \emptyset & \overline{B}(\Gamma_{\Pi_1}, C_2) = \{r_1, r_2, r_5, r_6\} \end{array} \tag{3}$$

The next results are important for understanding the idea of our approach.

Theorem 1. *Let Γ be the RDG of logic program Π and C be a partial coloring of Γ. Then, we have for every $X \in AS_\Pi(C)$ that*

1. *$S(\Gamma, C) \cap \overline{B}(\Gamma, C) \subseteq R_\Pi(X)$;*
2. *$\overline{S}(\Gamma, C) \cup B(\Gamma, C) \subseteq \Pi \setminus R_\Pi(X)$.*

If C is admissible, we have for $\{X\} = AS_\Pi(C)$ that

3. *$S(\Gamma, C) \cap \overline{B}(\Gamma, C) = R_\Pi(X)$;*
4. *$\overline{S}(\Gamma, C) \cup B(\Gamma, C) = \Pi \setminus R_\Pi(X)$.*

Equation 3 and 4 are equivalent since C is total. Reconsider the partially colored *RDG* (Γ_{Π_1}, C_2) in Figure 1b. For every $X \in AS_{\Pi_1}(C_2) = \{\{b, p, f\}, \{b, p, f'\}\}$, we have

$$\begin{array}{ll} S(\Gamma_{\Pi_1}, C_2) \cap \overline{B}(\Gamma_{\Pi_1}, C_2) = \{r_1, r_2\} & \subseteq R_{\Pi_1}(X); \\ \overline{S}(\Gamma_{\Pi_1}, C_2) \cup B(\Gamma_{\Pi_1}, C_2) = \{r_5\} & \subseteq \Pi \setminus R_{\Pi_1}(X). \end{array}$$

3 Deciding answersetship from colored graphs

The result in Theorem 1 started from an existing answer set induced from a given coloring. We now develop concepts that allow us to decide whether a (total) coloring represents an answer set by purely graph-theoretical means. To begin with, we define a graph structure accounting for the notion of recursive support.

Definition 3. *Let Γ be the RDG of logic program Π 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 $C_\ominus \cap V = \emptyset$.*

Every uncolored *RDG* (with $C = (\emptyset, \emptyset)$) has a unique support graph possessing a largest set of vertices. We refer to such support graphs as *maximal* ones; all of them share the same set of vertices. For example, the maximal support graph of $(\Gamma_{\Pi_1}, (\emptyset, \emptyset))$, given in Figure 1a, excludes r_5, since it cannot be supported (recursively); otherwise, it contains, except for (r_5, r_3), all 0-edges of Γ_{Π_1}. The maximal support graph of the colored *RDG* (Γ_{Π_1}, C_2), given in Figure 1b, is $(\{r_1, r_2, r_3, r_4\}, \{(r_1, r_2), (r_1, r_4), (r_2, r_3)\})$. It includes all positively colored and exclude all negatively colored nodes in (Γ_{Π_1}, C_2).

Given a program $\{q, p \leftarrow q\}$ a “bad” coloring, like $C = (\{p \leftarrow q\}, \{q\})$, may deny the existence of a support graph of (Γ, C). As above, we distinguish maximal support graphs of colored graphs through their maximal set of vertices. For colored graphs, we have the following conditions guaranteeing the existence of (maximal) support graphs.

Theorem 2. *Let Γ be the RDG of logic program Π and C be a partial coloring of Γ. If $AC_\Pi(C) \neq \emptyset$, then there is a (maximal) support graph of (Γ, C).*

Clearly, the existence of a support graph implies that of a maximal one. Note furthermore that support graphs of totally colored graphs are necessarily maximal.

Corollary 1. *Let Γ be the RDG of logic program Π and C be an admissible coloring of Γ. Then, $(C_\oplus, E)$ is a support graph of (Γ, C) for some $E \subseteq (\Pi \times \Pi)$.*

Taking the last result together with Property 3 or 4 in Theorem 1, we obtain a sufficient characterization of admissible colorings (along with their underlying answer sets).

Theorem 3. *Let Γ be the RDG of logic program Π and let C be a total coloring of Γ. Then, the following statements are equivalent.*

1. *C is an admissible coloring of Γ;*
2. *$C_\oplus = \underline{S}(\Gamma, C) \cap \overline{B}(\Gamma, C)$ and there is a support graph of (Γ, C);*
3. *$C_\ominus = \overline{S}(\Gamma, C) \cup \underline{B}(\Gamma, C)$ and there is a support graph of (Γ, C).*

For illustration, let us consider the two admissible colorings of *RDG* Γ_{Π_1}, corresponding to the two answer sets of program Π_1:

$$C_{4a} = (\{r_1, r_2, r_3\},\ \{r_4, r_5, r_6\}) \quad \text{and} \quad C_{4b} = (\{r_1, r_2, r_4\},\ \{r_3, r_5, r_6\}). \quad (4)$$

The resulting colored *RDG*s are depicted in Figure 1c+d. Let us detail the case of C_{4a}:

$$\underline{S}(\Gamma_{\Pi_1}, C_{4a}) \cap \overline{B}(\Gamma_{\Pi_1}, C_{4a}) = \{r_1, r_2, r_3\} = (C_{4a})_\oplus;$$
$$\overline{S}(\Gamma_{\Pi_1}, C_{4a}) \cup \underline{B}(\Gamma_{\Pi_1}, C_{4a}) = \{r_4, r_5, r_6\} = (C_{4a})_\ominus.$$

The maximal support graph of (Γ_{Π_1}, C_{4a}) is given by $((C_{4a})_\oplus, \{(r_1, r_2), (r_2, r_3)\})$.

In the full paper [12], we show how our graph-theoretical approach allows for capturing the original concepts like Cn and Π^X. Also, we introduce the concept of a *blockage graph* by means of 1-subgraphs for capturing blockage relations.

4 Operational characterizations

The goal of this section is to provide operational characterizations of answer sets. The idea is to start with the empty coloring $(\emptyset, \emptyset)$ and to successively apply operators that turn a partial coloring C into another one C' such that $C \sqsubseteq C'$, if possible. This is done until an admissible coloring, encompassing an answer set, is obtained.

We concentrate first on operations deterministically extending partial colorings.

Definition 4. *Let Γ be the RDG of logic program Π and C be a partial coloring of Γ. Then, define $\mathcal{P}_\Gamma : \mathbb{C} \to \mathbb{C}$ as*

$$\mathcal{P}_\Gamma(C) = C \sqcup (S(\Gamma, C) \cap \overline{B}(\Gamma, C),\ \overline{S}(\Gamma, C) \cup B(\Gamma, C))\ .$$

A partial coloring C is closed under $\mathcal{P}_\Gamma$, if $C = \mathcal{P}_\Gamma(C)$. Note that $\mathcal{P}_\Gamma(C)$ does not always exist. To see this, observe that $\mathcal{P}_\Gamma((\{a \leftarrow not\ a\}, \emptyset))$ would be $(\{a \leftarrow not\ a\}, \{a \leftarrow not\ a\})$, which is no mapping and thus no partial coloring.

Interestingly, $\mathcal{P}_\Gamma$ exists on colorings expressing answer sets.

Theorem 4. *Let Γ be the RDG of logic program Π and C a partial coloring of Γ.*
If $AC_\Pi(C) \neq \emptyset$, then $\mathcal{P}_\Gamma(C)$ exists.

Note that $\mathcal{P}_\Gamma(C)$ may exist although $AC_\Pi(C) = \emptyset$. To see this, consider $\Pi = \{a \leftarrow , c \leftarrow a, not\ c\}$. Clearly, $AC_\Pi(C) = \emptyset$. However, $\mathcal{P}_\Gamma((\emptyset, \emptyset)) = (\{r_1\}, \emptyset)$ exists.

Now, we can define our principal propagation operator in the following way.

Definition 5. *Let Γ be the RDG of logic program Π and C a partial coloring of Γ.*
Then, define $\mathcal{P}^*_\Gamma : \mathbb{C} \to \mathbb{C}$ where $\mathcal{P}^*_\Gamma(C)$ is the $\sqsubseteq$-smallest partial coloring containing C and being closed under $\mathcal{P}_\Gamma$.

Like $\mathcal{P}_\Gamma(C)$, $\mathcal{P}^*_\Gamma(C)$ is not necessarily defined. This situation is made precise next.

Theorem 5. *Let Γ be the RDG of logic program Π and C a partial coloring of Γ.*
If $AC_\Pi(C) \neq \emptyset$, then $\mathcal{P}^*_\Gamma(C)$ exists.

In fact, the non-existence of $\mathcal{P}^*_\Gamma$ is an important feature since an undefined application of $\mathcal{P}^*_\Gamma$ amounts to a backtracking situation at the implementation level. Note that $\mathcal{P}^*_\Gamma((\emptyset, \emptyset))$ always exists, even though we may have $AC_\Pi((\emptyset, \emptyset)) = \emptyset$ (because of $AS(\Pi) = \emptyset$).

For illustration, consider program Π_1. We get:

$$\begin{aligned} \mathcal{P}_\Gamma((\emptyset, \emptyset)) &= (\{r_1\}, \{r_5\}) \\ \mathcal{P}_\Gamma((\{r_1\}, \{r_5\})) &= (\{r_1, r_2\}, \{r_5\}) \\ \mathcal{P}_\Gamma((\{r_1, r_2\}, \{r_5\})) &= (\{r_1, r_2\}, \{r_5\}) \text{ and so } \mathcal{P}^*_\Gamma((\emptyset, \emptyset)) = (\{r_1, r_2\}, \{r_5\})\ . \end{aligned}$$

Let us now elaborate upon the formal properties of $\mathcal{P}_\Gamma$ and $\mathcal{P}^*_\Gamma$. First, we observe that both are reflexive, that is, $C \sqsubseteq \mathcal{P}_\Gamma(C)$ and $C \sqsubseteq \mathcal{P}^*_\Gamma(C)$ provided they exist. As shown in the full paper [12], both operators are monotonic: For partial colorings C, C' of Γ such that $AC_\Pi(C') \neq \emptyset$, we have: If $C \sqsubseteq C'$, then $\mathcal{P}_\Gamma(C) \sqsubseteq \mathcal{P}_\Gamma(C')$; analogously for $\mathcal{P}^*_\Gamma$. Consequently, we have $C \sqsubseteq \mathcal{P}_\Gamma(C) \sqsubseteq \mathcal{P}_\Gamma(\mathcal{P}_\Gamma(C))$. Moreover, $\mathcal{P}_\Gamma$ and $\mathcal{P}^*_\Gamma$ are answer set preserving: $AC_\Pi(C) = AC_\Pi(\mathcal{P}_\Gamma(C)) = AC_\Pi(\mathcal{P}^*_\Gamma(C))$.

In fact, $\mathcal{P}_\Gamma$ can be used for deciding answersetship in the following way.

Corollary 2. *Let Γ be the RDG of logic program Π and let C be a total coloring of Γ. Then, C is an admissible coloring of Γ iff $\mathcal{P}_\Gamma(C) = C$ and (Γ, C) has a support graph.*

For relating $\mathcal{P}^*_\Gamma$ to the well-known Fitting operator [7], we need the following.

Definition 6. *Let Γ be the RDG of logic program Π and let C be a partial coloring of Γ. Define $X_C = \{head(r) \mid r \in C_\oplus\}$ and $Y_C = \{q \mid$ for all $r \in \Pi$, if $head(r) = q$, then $r \in C_\ominus\}$.*

The pair (X_C, Y_C) is a 3-valued interpretation of Π. By letting the pair mapping $\Phi_\Pi(X, Y)$ be Fitting's operator [7], we have the following result.

Theorem 6. *Let Γ be the RDG of logic program Π.*
If $C = \mathcal{P}^*_\Gamma((\emptyset, \emptyset))$, then $\Phi^\omega_\Pi(\emptyset, \emptyset) = (X_C, Y_C)$.

A more detailed analysis of this relationship is given in the full paper [12].

The next operation draws upon the maximal support graph of colored *RDG*s.

Definition 7. *Let Γ be the RDG of logic program Π 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

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

This operator allows for coloring rules with $\ominus$ whenever it is clear from the given partial coloring that they will remain unsupported.[1] Observe that $\Pi \setminus V = C_\ominus \cup (\Pi \setminus V)$. Like $\mathcal{P}^*_\Gamma$, $\mathcal{U}_\Gamma(C)$ 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. For Π_1, for instance, we obtain $\mathcal{U}_\Gamma((\emptyset, \emptyset)) = (\emptyset, \{r_5\})$. While this information on r_5 can also be supplied by $\mathcal{P}_\Gamma$, it is not obtainable for "self-supporting 0-loops", as in $\Pi = \{p \leftarrow q, q \leftarrow p\}$. In this case, we obtain $\mathcal{U}_\Gamma((\emptyset, \emptyset)) = (\emptyset, \{p \leftarrow q, q \leftarrow p\})$, which is not obtainable through $\mathcal{P}_\Gamma$.

Operator $\mathcal{U}_\Gamma$ is defined on colorings guaranteeing the existence of support graphs.

Corollary 3. *Let Γ be the RDG of logic program Π 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 [12] that $\mathcal{U}_\Gamma$ is reflexive, idempotent, monotonic, and answer set preserving. That is, for partial colorings C and C' of Γ such that $AC_\Pi(C) \neq \emptyset$ and $AC_\Pi(C') \neq \emptyset$, 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 $AC_\Pi(C) = AC_\Pi(\mathcal{U}_\Gamma(C))$. Note that unlike $\mathcal{P}_\Gamma$, operator $\mathcal{U}_\Gamma$ leaves the support graph of (Γ, C) unaffected.

Because $\mathcal{U}_\Gamma$ implicitly enforces the existence of a support graph, our operators furnish yet another characterization of answer sets.

[1] The relation to unfounded sets is detailed in the full paper [12].

Corollary 4. *Let Γ be the RDG of logic program Π and let C be a total coloring of Γ. Then, C is an admissible coloring of Γ iff $C = \mathcal{P}_\Gamma(C)$ and $C = \mathcal{U}_\Gamma(C)$.*

Note that the last condition cannot guarantee that all supported unblocked rules belong to $C_\oplus$. For instance, $(\emptyset, \{a \leftarrow\})$ has an empty support graph; hence $(\emptyset, \{a \leftarrow\}) = \mathcal{U}_\Gamma((\emptyset, \{a \leftarrow\}))$. That is, the trivially supported fact $a \leftarrow$ remains in $C_\ominus$. In our setting, such a miscoloring is detected by $\mathcal{P}_\Gamma$. That is, $\mathcal{P}_\Gamma((\emptyset, \{a \leftarrow\}))$ is no partial coloring.

Finally, we can express well-founded semantics [18] with our operators. For this, given a partial coloring C, 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 RDG of logic program Π.
If $C = (\mathcal{PU})^*_\Gamma((\emptyset, \emptyset))$, then (X_C, Y_C) is the well-founded model of Π.*

A more detailed analysis of this relationship is given in the full paper [12].

We continue by providing a very general operational characterization that possesses a maximum degree of freedom. To this end, we observe that Corollary 4 can serve as a straightforward *check* for deciding whether a given total coloring constitutes an answer set. A corresponding *guess* can be provided through an operator capturing a non-deterministic (don't know) choice.

Definition 8. *Let Γ be the RDG of logic program Π and C be a partial coloring of Γ. For $\circ \in \{\oplus, \ominus\}$, define $\mathcal{C}^\circ_\Gamma : \mathbb{C} \to \mathbb{C}$ as*

1. $\mathcal{C}^\oplus_\Gamma(C) = (C_\oplus \cup \{r\}, C_\ominus)$ *for some* $r \in \Pi \setminus (C_\oplus \cup C_\ominus)$;
2. $\mathcal{C}^\ominus_\Gamma(C) = (C_\oplus, C_\ominus \cup \{r\})$ *for some* $r \in \Pi \setminus (C_\oplus \cup C_\ominus)$.

We use $\mathcal{C}^\circ_\Gamma$ if the distinction between $\mathcal{C}^\oplus_\Gamma(C)$ and $\mathcal{C}^\ominus_\Gamma(C)$ is of no importance. Strictly speaking, $\mathcal{C}^\circ_\Gamma$ is also parametrized with r; we leave this implicit.

Combining the previous guess and check operators yields our first operational characterization of admissible colorings (along with its underlying answer sets).

Theorem 8. *Let Γ be the RDG of logic program Π and let C be a total coloring of Γ. Then, C is an admissible coloring of Γ iff there exists a sequence $(C^i)_{0 \le i \le n}$ where*

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

We refer to such sequences also as *coloring sequences*. Note that all sequences satisfying conditions 1-5 of Theorem 8 are *successful* insofar as their last element corresponds to an existing answer set. If a program has no answer set, then no such sequence exists.

Although this guess and check approach is of no great implementational value, it supplies us with a skeleton for the coloring process that we refine in the sequel. In particular, it stresses the basic fact that we possess complete freedom in forming a coloring sequence as long as we can guarantee that the resulting coloring is a fixed point of $\mathcal{P}_\Gamma$ and $\mathcal{U}_\Gamma$. It is worth mentioning that this simple approach is inapplicable when fixing $\circ$ to either $\oplus$ or $\ominus$ (see full paper [12]). We observe the following properties.

Theorem 9. *Given the prerequisites in Theorem 8, let $(C^i)_{0\leq i\leq n}$ be a sequence satisfying conditions 1-5 in Theorem 8. Then, we have the following properties for $0 \leq i \leq n$.*

1. *C^i is a partial coloring;*
2. *$C^i \sqsubseteq C^{i+1}$;*
3. *$AC_\Pi(C^i) \supseteq AC_\Pi(C^{i+1})$;*
4. *$AC_\Pi(C^i) \neq \emptyset$;*
5. *(Γ, C^i) has a (maximal) support graph.*

All these properties represent invariants of the consecutive colorings. While the first three properties are provided by operator $\mathcal{C}^\circ_\Gamma$ in choosing among uncolored rules only, the last two properties are actually enforced by the "check" on the final coloring C^n expressed by conditions 3–5. In fact, sequences only enjoying conditions 1 and 2 in Theorem 8, fail to satisfy Property 4 and 5. In practical terms, this means that computations of successful sequences may be led numerous times on the "garden path".

As well-known, the number of choices can be significantly reduced by applying deterministic operators.

Theorem 10. *Let Γ be the* RDG *of logic program Π and let C be a total coloring of Γ. Then, C is an admissible coloring of Γ iff there exists a sequence $(C^i)_{0\leq i\leq n}$ where*

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

The continuous applications of $\mathcal{P}_\Gamma$ and $\mathcal{U}_\Gamma$ extend colorings after each choice. Furthermore, this proceeding guarantees that each partial coloring C^i is closed under $\mathcal{P}_\Gamma$ and $\mathcal{U}_\Gamma$. It is clear in view of Theorem 8 that any number of iterations of $\mathcal{P}_\Gamma$ and $\mathcal{U}_\Gamma$ can be executed after $\mathcal{C}^\circ_\Gamma$ as long as $(\mathcal{PU})^*_\Gamma$ is the final operation leading to C^n in Theorem 10.

For illustration, consider the coloring sequence in Figure 2, obtained for answer set $\{b, p, f'\}$ of program Π_1. The decisive operation in this sequence is the application of $\mathcal{C}^\oplus_\Gamma$ leading to $C(r_3) = \oplus$. The same final result is obtained when choosing $\mathcal{C}^\ominus_\Gamma$ such

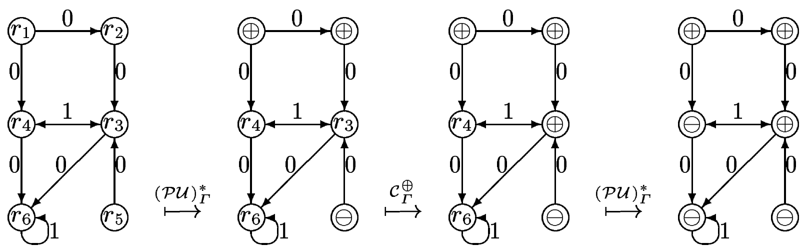

Fig. 2. A coloring sequence.

that $C(r_4) = \ominus$. So, several coloring sequences may lead to the same answer set.

The usage of continuous propagations leads to further invariant properties.

Theorem 11. *Given the prerequisites in Theorem 10, let $(C^i)_{0\leq i\leq n}$ be a sequence satisfying conditions 1-3 in Theorem 10. Then, we have properties 1–5 in Theorem 9 and*

6. $C_{\oplus}^{i+1} \supseteq S(\Gamma, C^i) \cap \overline{B}(\Gamma, C^i)$;
7. $C_{\ominus}^{i+1} \supseteq \overline{S}(\Gamma, C^i) \cup B(\Gamma, C^i)$.

Taking Property 6 and 7 together with 5 from Theorem 9, we see that propagation gradually enforces exactly the attributes on partial colorings, expressed in Theorem 3.

Given that we obtain only two additional properties, one may wonder whether exhaustive propagation truly pays off. In fact, its value becomes apparent when looking at the properties of prefix sequences, not necessarily leading to a successful end.

Theorem 12. *Given the prerequisites in Theorem 10, let $(C^j)_{0 \le j \le m}$ be a sequence satisfying Condition 1 and 2 in Theorem 10.*

Then, we have properties 1–3, 5 in Theorem 9 and 6–7 in Theorem 11.

Using exhaustive propagations, we observe that except for Property 4 all properties of successful sequences, are shared by (possibly unsuccessful) prefix sequences. In the full paper [12], we prove that propagation leads to shorter and fewer (prefix) sequences.

What else may cut down the number of choices? Looking at the graph structures underlying an admissible coloring, we observe that support graphs possess a non-local, since recursive, structure, while blockage exhibits a rather local structure, based on arc-wise constraints. Consequently, it seems advisable to prefer choices maintaining support structures over those maintaining blockage relations, since the former have more global repercussions than the latter. To this end, we develop in what follows a strategy that is based on a choice operation restricted to supported rules.

Definition 9. *Let Γ be the RDG of logic program Π and C be a partial coloring of Γ.*

For $\circ \in \{\oplus, \ominus\}$, define $\mathcal{D}_{\Gamma}^{\circ} : \mathbb{C} \to \mathbb{C}$ as

1. $\mathcal{D}_{\Gamma}^{\oplus}(C) = (C_{\oplus} \cup \{r\}, C_{\ominus})$ *for some* $r \in S(\Gamma, C) \setminus (C_{\oplus} \cup C_{\ominus})$;
2. $\mathcal{D}_{\Gamma}^{\ominus}(C) = (C_{\oplus}, C_{\ominus} \cup \{r\})$ *for some* $r \in S(\Gamma, C) \setminus (C_{\oplus} \cup C_{\ominus})$.

The number of rules colorable by $\mathcal{D}_{\Gamma}^{\circ}$ is normally smaller than that by $\mathcal{C}_{\Gamma}^{\circ}$. Depending on how the non-determinism of $\mathcal{D}_{\Gamma}^{\circ}$ is dealt with algorithmically, this may either lead to a reduced depth of the search tree or a reduced branching factor.

In a successful coloring sequence $(C^i)_{0 \le i \le n}$, all rules in $C_{\oplus}^n$ belong to an encompassing support graph. Furthermore, using $\mathcal{D}_{\Gamma}^{\oplus}(C)$ (and $\mathcal{P}_{\Gamma}^{*}$) the supportness of each set $C_{\oplus}^i$ is made invariant. Consequently, such a proceeding allows for establishing the existence of support graphs "on the fly" and offers a much simpler approach to the task(s) previously accomplished by $\mathcal{U}_{\Gamma}$. In fact, one may completely dispose of operator $\mathcal{U}_{\Gamma}$ and color in a final step all uncolored rules with $\ominus$.

Definition 10. *Let Γ be the RDG of logic program Π and C a partial coloring of Γ. Then, define $\mathcal{N}_{\Gamma} : \mathbb{C} \to \mathbb{C}$ as $\mathcal{N}_{\Gamma}(C) = (C_{\oplus}, \Pi \setminus C_{\oplus})$.*

Roughly speaking, the idea is then to "actively" color only supported rules and rules blocked by supported rules; all remaining rules are then unsupported and "thrown" into $C_{\ominus}$ in a final step.

Theorem 13. *Let Γ be the RDG of logic program Π and let C be a total coloring of Γ.*

Then, C is an admissible coloring of Γ iff there exists a sequence $(C^i)_{0 \le i \le n}$ where

1. $C^0 = (\emptyset, \emptyset)$;
2. $C^{i+1} = \mathcal{D}^{\circ}_{\Gamma}(C^i)$ *where* $\circ \in \{\oplus, \ominus\}$ *and* $0 \leq i < n-1$;
3. $C^n = \mathcal{N}_{\Gamma}(C^{n-1})$;
4. $C^n = \mathcal{P}_{\Gamma}(C^n)$;
5. $C^n = C$.

We note that there is a little price to pay for turning $\mathcal{U}_{\Gamma}$ into $\mathcal{N}_{\Gamma}$, expressed in Condition 4. Without it, one could use $\mathcal{N}_{\Gamma}$ to obtain a total coloring by coloring rules with $\ominus$ in an arbitrary way. We obtain the following properties for this type of sequences.

Theorem 14. *Given the prerequisites in Theorem 13, let* $(C^i)_{0 \leq i \leq n}$ *be a sequence satisfying conditions 1-5 in Theorem 13. Then, we have properties 1–5 in Theorem 9 and*

8. $(C^i_{\oplus}, E)$ *is a support graph of* (Γ, C^i) *for some* $E \subseteq \Pi \times \Pi$.

Unlike the coloring sequences only enjoying Condition 5 in Theorem 9, the sequences formed by means of $\mathcal{D}^{\circ}_{\Gamma}$ guarantee that each $C^i_{\oplus}$ forms an independent support graph.

In fact, there is some overlap among operator $\mathcal{D}^{\ominus}_{\Gamma}$ and $\mathcal{N}_{\Gamma}$. To see this, consider $\Pi = \{a \leftarrow\ , b \leftarrow not\ a\}$. Initially, we must apply $\mathcal{D}^{\oplus}_{\Gamma}$ to obtain $(\{a \leftarrow\}, \emptyset)$ from $(\emptyset, \emptyset)$. Then, however, we may either apply $\mathcal{D}^{\ominus}_{\Gamma}$ or $\mathcal{N}_{\Gamma}$ for obtaining admissible coloring $(\{a\}, \{b \leftarrow not\ a\})$. Interestingly, this overlap can be eliminated by adding propagation operator $\mathcal{P}^*_{\Gamma}$. This results in the basic strategy used in the `noMoRe` system [1].

Theorem 15. *Let* Γ *be the* RDG *of logic program* Π *and let* C *be a total coloring of* Γ. *Then,* C *is an admissible coloring of* Γ *iff there exists a sequence* $(C^i)_{0 \leq i \leq n}$ *where*

1. $C^0 = \mathcal{P}^*_{\Gamma}((\emptyset, \emptyset))$;
2. $C^{i+1} = \mathcal{P}^*_{\Gamma}(\mathcal{D}^{\circ}_{\Gamma}(C^i))$ *where* $\circ \in \{\oplus, \ominus\}$ *and* $0 \leq i < n-1$;
3. $C^n = \mathcal{N}_{\Gamma}(C^{n-1})$;
4. $C^n = \mathcal{P}_{\Gamma}(C^n)$;
5. $C^n = C$.

Indeed, the strategy of `noMoRe` applies operator $\mathcal{D}^{\circ}_{\Gamma}$ as long as there are supported rules. Once no more uncolored supported rules exist, operator $\mathcal{N}_{\Gamma}$ is called. Finally, $\mathcal{P}_{\Gamma}$ is applied but, in practice, only to those rules colored previously by $\mathcal{N}_{\Gamma}$. At first sight, this approach may seem to correspond to a subclass of the coloring sequences described above, in the sense that `noMoRe` enforces a maximum number of transitions described in Condition 2 above. To see that this is not the case, we observe the following property.

Theorem 16. *Given the prerequisites in Theorem 15, let* $(C^i)_{0 \leq i \leq n}$ *be a sequence satisfying conditions 1-5 in Theorem 15. Then, we have* $(\mathcal{N}_{\Gamma}(C^{n-1})_{\ominus} \setminus C^{n-1}_{\ominus}) \subseteq \overline{S}(\Gamma, C)$.

That is, no matter which (supported) rules are colored $\ominus$ by $\mathcal{D}^{\ominus}_{\Gamma}$, operator $\mathcal{N}_{\Gamma}$ only applies to unsupported ones. It is thus no restriction to enforce the consecutive application of $\mathcal{P}^*_{\Gamma}$ and $\mathcal{D}^{\circ}_{\Gamma}$ until no more supported rules are available. In fact, it is the interplay of the two last operators that guarantees this property. For instance, looking at $\Pi = \{a, b \leftarrow not\ a\}$, we see that we directly obtain the final total coloring because $(\{a\}, \{b \leftarrow not\ a\}) = \mathcal{P}^*_{\Gamma}(\mathcal{D}^{\oplus}_{\Gamma}((\emptyset, \emptyset)))$, without any appeal to $\mathcal{N}_{\Gamma}$. Rather it is $\mathcal{P}^*_{\Gamma}$ that detects that $b \leftarrow not\ a$ is blocked. Generally speaking, $\mathcal{D}^{\oplus}_{\Gamma}$ consecutively chooses the generating rules of an answer set, finally gathered in $C_\oplus = S(\Gamma, C) \cap \overline{B}(\Gamma, C)$. Clearly, every rule in $B(\Gamma, C)$ is blocked by some rule in $C_\oplus$. So whenever a rule r is added by $\mathcal{D}_\Gamma^\oplus$ to $C_\oplus$, operator $\mathcal{P}_\Gamma^*$ adds all rules blocked by r to $C_\ominus$. In this way, $\mathcal{P}_\Gamma^*$ and $\mathcal{D}_\Gamma^\oplus$ gradually color all rules in $S(\Gamma, C) \cap \overline{B}(\Gamma, C)$ and $B(\Gamma, C)$, so that all remaining uncolored rules, subsequently treated by $\mathcal{N}_\Gamma$, must belong to $\overline{S}(\Gamma, C)$. We obtain the following properties.

Theorem 17. *Given the prerequisites in Theorem 15, let $(C^i)_{0 \leq i \leq n}$ be a sequence satisfying conditions 1-5 in Theorem 15. Then, we have properties 1–5 in Theorem 9, 6–7 in Theorem 11, and 8 in Theorem 14.*

In the full paper [12], we discuss an alternative support-driven operational characterization using an incremantal version of $\mathcal{U}_\Gamma$ instead of $\mathcal{N}_\Gamma$. As well, we elaborate upon unicoloring strategies, using only one of the above choice operators instead of both.

5 Discussion, related work, and conclusions

Among the many graph-based approaches in the literature, we find some dealing with stratification [2], existence of answer sets [6, 3], or the actual characterization of answer sets or well-founded semantics [4, 3, 9, 11]. Our own approach has its roots in earlier work on default logic [13, 14, 17]. The usage of rule-oriented dependency graphs is common to [4, 3, 9]. In fact, the coloration of such graphs for characterizing answer sets was independently developed in [3] and [9]. While we borrow the term of an admissible coloring from the former, the work reported in Section 3 builds upon the latter and revises its definitions by appeal to the concept of a support graph. [2]

Our major goal is however to provide an operational framework for answer set formation that allows us to bridge the gap between formal yet static characterizations of answer sets and algorithms for computing them. For instance, in the seminal paper [16] describing the `smodels` approach, answer sets are given in terms of so-called *full-sets* and their computation is directly expressed in terms of procedural algorithms. Our operational semantics aims at offering an intermediate stage that facilitates the formal elaboration of computational approaches. Our approach is strongly inspired by the concept of a derivation, in particular, that of an SLD-derivation [15]. This attributes our coloring sequences the flavor of a derivation in a family of calculi, whose respective set of inference rules correspond to the selection of operators.

Although we leave out implementational issues, some remarks relating our approach, and thus the resulting `noMoRe` system [1], to the ones underlying `dlv` [5] and `smodels` [16] are in order. A principal difference manifests itself in how choices are performed. While the two latter's choice is based on atoms occurring (negatively) in the underlying program, our choices are based on its rules. An advantage of our approach is that we can guarantee the support of rules on the fly. Unlike this, support checking is a recurring operation in the `smodels` system, similar to operator $\mathcal{U}_\Gamma$. On

[2] The definition of *RDG*s differs from that of "block graphs" in [9], whose practically motivated restrictions are superfluous from a theoretical perspective. Also, we abandon the latter term in order to give the same status to support and blockage relations.

the other hand, this approach ensures that the `smodels` algorithm runs in linear space complexity, while a graph-based approach needs quadratic space in the worst case. This "investment" pays off once one is able to exploit the additional structural information offered by a graph. First steps in this direction are made in [10], where graph compressions are described that allow for conflating entire subgraphs into single nodes. Propagation is more or less done similarly in all three approaches. `smodels` relies on computing well-founded semantics, whereas `dlv` uses Fitting's operator plus some backpropagation mechanisms.

To sum up, we build upon the basic graph-theoretical characterizations in [9, 11] for developing an operational framework for non-deterministic answer set formation. The general idea is to start from an uncolored *RDG* and to employ specific operators that turn a partially colored graph gradually in a totally colored one, representing an answer set. To this end, we have developed a variety of deterministic and non-deterministic operators. Different coloring sequences (enjoying different formal properties) are obtained by selecting different combinations of operators. Among others, we have identified the particular strategy of the `noMoRe` system as well as operations yielding Fitting's and well-founded semantics. Taken together, the last results show that `noMoRe`'s principal propagation operation amounts to applying Fitting's operator. Notably, the explicit detection of 0-loops is avoided by employing a support-driven choice operation. The `noMoRe` system is available at `http://www.cs.uni-potsdam.de/∼linke/nomore`.

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. K. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, 1987.
3. 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.
4. Y. Dimopoulos and A. Torres. Graph theoretical structures in logic programs and default theories. *Theoretical Computer Science*, 170:209–244, 1996.
5. T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A deductive system for nonmonotonic reasoning. In J. Dix et al., editors, *Proceedings of the Fourth International Conference on Logic Programming and Non-Monotonic Reasoning*, pages 363–374. Springer, 1997.

6. F. Fages. Consistency of clark's completion and the existence of stable models. *Journal of Methods of Logic in Computer Science*, 1:51–60, 1994.
7. M. Fitting. Fixpoint semantics for logic programming a survey. *Theoretical Computer Science*, 278(1-2):25–51, 2002.
8. M. Gelfond and V. Lifschitz. Classical negation in logic programs and deductive databases. *New Generation Computing*, 9:365–385, 1991.
9. 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.
10. T. Linke. Using nested logic programs for answer set programming. 2003. Submitted.
11. T. Linke, C. Anger, and K. Konczak. More on `noMoRe`. In S. Flesca et al., editors, *Proceedings of the Eighth European Conference on Logics in Artificial Intelligence (JELIA'02)*, pages 468–480, 2002.
12. T. Linke, K. Konczak, and T. Schaub. Graphs and colorings for answer set programming. Technical Report KRR-TR-03-01, University of Potsdam, June 2003. Available at `http://www.cs.uni-potsdam.de/~torsten/Papers/KRR-TR-03-01.ps`.
13. T. Linke and T. Schaub. An approach to query-answering in Reiter's default logic and the underlying existence of extensions problem. In J. Dix et al., editors, *Proceedings of the Sixth European Workshop on Logics in Artificial Intelligence*, pages 233–247. Springer, 1998.
14. T. Linke and T. Schaub. Alternative foundations for Reiter's default logic. *Artificial Intelligence*, 124(1):31–86, 2000.
15. J. Lloyd. *Foundations of Logic Programming*. Springer, 1987.
16. I. Niemelä and P. Simons. Efficient implementation of the well-founded and stable model semantics. In M. Maher, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 289–303. The MIT Press, 1996.
17. C. Papadimitriou and M. Sideri. Default theories that always have extensions. *Artificial Intelligence*, 69:347–357, 1994.
18. A. van Gelder, K. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.