NoMoRe: A System for Non-monotonic Reasoning with Logic Programs under Answer Set Semantics

Christian Anger, Kathrin Konczak, and Thomas Linke

Universität Potsdam, Institut für Informatik, Am Neuen Palais 10, D-14469 Potsdam, \{canger, konczak, linke\}@cs.uni-potsdam.de

1 Introduction

The noMoRe system (first prototype) implements answer set semantics for propositional normal logic programs. It uses an alternative implementation paradigm to compute answer sets by computing non-standard graph colorings of labeled directed graphs associated with logic programs. Therefore noMoRe is an interesting experimental tool for scientists working with logic programs on a theoretical or practical basis. Furthermore, we have included a tool for visualization of those graphs corresponding to programs.

2 General Information

The noMoRe-system is implemented in the programming language Prolog; it has been developed under the ECLiPSe Constraint Logic Programming System \[1\] and it was also successfully tested with SWI-Prolog \[11\]. The source code, test cases and documentation are available at http://www.cs.uni-potsdam.de/~linke/nomore. In order to use the system, ECLiPSe- or SWI-Prolog is needed \[1,11\]. Both Prolog systems are freely available for scientific use. Clearly, noMoRe works under each platform under which one of the above Prolog systems is available. The total number of lines of code is only about 2700, i.e. noMoRe is very transparent and it nicely reflects the underlying theory.

3 Description of the System

The experimental prototype of the noMoRe system implements nonmonotonic reasoning with propositional normal logic programs under answer set semantics \[5\]. Originally, answer set semantics was defined for extended logic programs \[2,5\] as a generalization of the stable model semantics \[4\] of normal logic programs. We consider rules \( r \) of the form

\[
p \leftarrow q_1, \ldots, q_n, \text{not } s_1, \ldots, \text{not } s_k
\]

Extended logic programs are logic programs with classical negation.
where \( p_i \) \((0 \leq i \leq n)\) and \( s_j \) \((0 \leq j \leq k)\) are ground atoms, \( \text{head}(r) = p \), \( \text{body}^+(r) = \{q_1, \ldots, q_n\} \), \( \text{body}^-(r) = \{s_1, \ldots, s_k\} \) and \( \text{body}(r) = \text{body}^+(r) \cup \text{body}^-(r) \). Intuitively, the head \( p \) of a rule \( p \leftarrow q_1, \ldots, q_n, \neg s_1, \ldots, \neg s_k \) is in some answer set \( A \) if \( q_1, \ldots, q_n \) are in \( A \) and none of \( s_1, \ldots, s_k \) is in \( A \). Look at the following normal logic program

\[
P = \{ a \leftarrow b, \neg e. \ b \leftarrow d. \ c \leftarrow b. \ d \leftarrow. \ e \leftarrow d, \neg f. \ f \leftarrow a. \} \quad (2)
\]

Let us call the rules of program (2) \( r_a, r_b, r_c, r_d, r_e, \) and \( r_f \), respectively. Then \( P \) has two different answer sets \( A_1 = \{d, b, c, a, f\} \) and \( A_2 = \{d, b, c, e\} \). It is easy to see that the application of \( r_f \) blocks the application of \( r_e \) wrt \( A_1 \), because if \( r_f \) contributes to \( A_1 \), then \( f \in A_1 \) and thus \( r_e \) cannot be applied. Analogously, \( r_e \) blocks \( r_a \) wrt answer set \( A_2 \).

### 3.1 Syntax

The syntax accepted by noMoRe is Prolog-like (without variables). For example, program (2) is represented through the following rules:

\[
\begin{align*}
  &a : - b, \neg e. \\
  &b : - d. \\
  &c : - b. \\
  &d. \\
  &e : - d, \neg f. \\
  &f : - a.
\end{align*}
\]

NoMoRe also accepts ground formulas which are treated as propositional atoms.

### 3.2 Block Graphs and A-Colorings

NoMoRe implements a novel paradigm to compute answer sets by computing non-standard graph colorings of the so-called block graph \([6]\) associated with a given program \( P \). A set of rules \( S \) of the form (1) is grounded iff there exists an enumeration \( \langle r_i \rangle_{i \in I} \) of \( S \) such that for all \( i \in I \) we have that \( \text{body}^+(r_i) \subseteq \text{head}((r_1, \ldots, r_{i-1})^2 \\). With this terminology, the block graph of \( P \) is defined as follows:

**Definition 1.** ([6]) Let \( P \) be a logic program and let \( P' \subseteq P \) be maximal grounded.\(^3\) The block graph \( \Gamma_P = (V_P, A_P^0 \cup A_P^1) \) of \( P \) is a directed graph with vertices \( V_P = P \) and two different kinds of arcs defined as follows

\[
\begin{align*}
  &A_P^0 = \{(r', r) \mid r', r \in P' \text{ and head}(r') \in \text{body}^+(r)\} \\
  &A_P^1 = \{(r', r) \mid r', r \in P' \text{ and head}(r') \in \text{body}^-(r)\}.
\end{align*}
\]

Figure [1] shows the block graph of program (2). Observe, that the rules of \( P \) are the nodes of \( \Gamma_P \). Since groundedness (by definition) ignores negative bodies, there exists a unique maximal grounded set \( P' \subseteq P \) for each program \( P \), that is, \( \Gamma_P \) is well-defined. Definition [1] captures the conditions under which a rule \( r' \) blocks another rule \( r \) (e.g. \( (r', r) \in A^1 \)). We also gather all groundedness information

\(^2\) The definition of the head of a rule is generalized to sets of rules in the usual way.

\(^3\) A maximal grounded set \( P' \) is a grounded set that is maximal wrt set inclusion.