# The *nomore++* Approach to Answer Set Solving

Christian Anger, Martin Gebser, Thomas Linke, André Neumann, and Torsten Schaub

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

**Abstract.** We present a new answer set solver called *nomore++*, along with its underlying theoretical foundations. A distinguishing feature is that it treats heads and bodies equitably as computational objects. Apart from its operational foundations, we show how it improves on previous work through its new lookahead and its support-driven strategy and underpin our claims by selected experimental results.

## 1 Introduction

A large part of the success of Answer Set Programming (ASP) is owed to the early availability of efficient solvers, like *smodels* [1] and *dlv* [2]. Since then, many other systems, sometimes following different approaches, have emerged, among them *assat* [3], *cmodels* [4], and *noMoRe* [5].

We present a new ASP solver, called *nomore++*, along with its underlying theoretical foundations. *nomore++* pursues a hybrid approach in combining features from literal-based approaches, like *smodels* and *dlv*, with the rule-based approach of its predecessor *noMoRe*. To this end, it treats heads and bodies equitably as computational objects. We argue that this approach allows for more effective (in terms of search space pruning) choices than obtainable when dealing with either heads or bodies only. As a particular consequence of this, we demonstrate that the resulting lookahead operation allows for propagating more than previous approaches. Also, we detail a special strategy, keeping the invariant property of being "unfounded-free" and empirically show that it outperforms *smodels* on relevant benchmarks. In fact, due to space limitations, we mainly compare our approach to that of *smodels*. Our choice is motivated by the fact that both systems primarily address normal logic programs.[1] *dlv* addresses the more expressive class of disjunctive logic programs. Thus, many of its distinguishing features are oriented to this extension. Also, *smodels* and *nomore++* share the same concept of "choice points", on which parts of our experiments rely upon.

The paper is organized as follows. After some preliminary definitions, we start with a strictly operational specification of *nomore++*. In fact, its configurable operator-based design is a salient feature of *nomore++*. Another major feature is its graph-based implementation. For simplicity, however, we give its specification in logic programming terminology and then describe how easily this is mapped into graph operations. We then concentrate on two specific features: First, we introduce *nomore++*'s lookahead operation and prove that, in terms of propagation, it is more powerful than the ones encountered in *smodels* and *noMoRe*. Second, we present *nomore++*'s support-driven

[1] Unlike *smodels*, *nomore++* cannot (yet) handle cardinality and weight constraints.

strategy along with further implementation details. Finally, we provide selected experimental results backing up our claims.

## 2 Background

A *logic program* is a finite set of rules of the form

$$p_0 \leftarrow p_1, \ldots, p_m, not\ p_{m+1}, \ldots, not\ p_n, \tag{1}$$

where $n \geq m \geq 0$, and each $p_i$ $(0 \leq i \leq n)$ is an *atom* in some alphabet $\mathcal{A}$. A literal is an atom $p$ or its negation $not\ p$. For $r$ as in (1), let $head(r) = p_0$ be the *head* of $r$ and $body(r) = \{p_1, \ldots, p_m, not\ p_{m+1}, \ldots, not\ p_n\}$ the *body* of $r$. Given a set $X$ of literals, let $X^+ = \{p \in \mathcal{A} \mid p \in X\}$ and $X^- = \{p \in \mathcal{A} \mid not\ p \in X\}$. For $r$, we then get $body(r)^+ = \{p_1, \ldots, p_m\}$ and $body(r)^- = \{p_{m+1}, \ldots, p_n\}$.

A program $\Pi$ is called *basic* if $body(r)^- = \emptyset$ for all $r \in \Pi$. The *reduct*, $\Pi^X$, of $\Pi$ 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 $X$ of atoms is closed under a basic program $\Pi$ if for any $r \in \Pi$, $head(r) \in X$ if $body(r)^+ \subseteq X$. $Cn(\Pi)$ denotes the smallest set of atoms closed under basic program $\Pi$. A set $X$ of atoms is an *answer set* of a program $\Pi$ if $Cn(\Pi^X) = X$.

As an example, consider program $\Pi_1$ comprising rules:

$$\begin{array}{ll} r_1 : a \leftarrow not\ b & r_3 : c \leftarrow not\ d \\ r_2 : b \leftarrow not\ a & r_4 : d \leftarrow not\ c \end{array} \tag{2}$$

We get four answer sets, viz. $\{a, c\}$, $\{a, d\}$, $\{b, c\}$, and $\{b, d\}$.

For a program $\Pi$, we write $head(\Pi) = \{head(r) \mid r \in \Pi\}$ and $body(\Pi) = \{body(r) \mid r \in \Pi\}$. Given that heads and bodies are our primary objects of interest, we further extend this notation: For $h \in head(\Pi)$, define $body(h) = \{body(r) \mid r \in \Pi, head(r) = h\}$.

For being able to define assignments in terms of bodies and heads of rules (in contrast to arbitrary atoms), we restrict ourselves, without loss of generality, to programs $\Pi$ satisfying $\{p \mid r \in \Pi, p \in body(r)^+ \cup body(r)^-\} \subseteq head(\Pi)$. That is, every body atom must occur as the head of some rule. Any program can be transformed into such a format, starting from the observation that all atoms in $(\mathcal{A} \setminus head(\Pi))$ are false.

## 3 Operational specification

To begin with, we give an operational specification of *nomore++* that is based on our extended concept of assignments. The purpose of this is to provide a firm understanding of the basic operations, which may serve as a basis for formal comparisons with techniques used by *smodels* or *dlv*.

We consider assignments that map heads and bodies in a program $\Pi$ into $\{\oplus, \ominus\}$, indicating whether a head or body is true or false, respectively. Formally, a (partial)

assignment is a partial mapping $A : head(\Pi) \cup body(\Pi) \rightarrow \{\oplus, \ominus\}$. For simplicity, we often represent such assignments as pairs $(A^{\oplus}, A^{\ominus})$, where $A^{\oplus} = \{x \mid A(x) = \oplus\}$ and $A^{\ominus} = \{x \mid A(x) = \ominus\}$. Whenever $A^{\oplus} \cap A^{\ominus} \neq \emptyset$, then $A$ is undefined as it is no mapping. We represent an undefined assignment by $(head(\Pi) \cup body(\Pi), head(\Pi) \cup body(\Pi))$. For comparing assignments $A$ and $B$, we define $A \sqsubseteq B$, if $A^{\oplus} \subseteq B^{\oplus}$ and $A^{\ominus} \subseteq B^{\ominus}$. Also, we define $A \sqcup B$ as $(A^{\oplus} \cup B^{\oplus}, A^{\ominus} \cup B^{\ominus})$.

We distinguish two sorts of forward propagation in *nomore++*. Head-oriented propagation assigning $\oplus$ to a head if one of its associated bodies belongs to $A^{\oplus}$ and assigning $\ominus$ whenever all of its bodies are in $A^{\ominus}$. This is captured by $T_{\Pi}(A)$ and $\overline{T}_{\Pi}(A)$ in Definition 1. Body-oriented propagation is based on the concepts of *support* and *blockage*: A body is supported if all its positive literals belong to $A^{\oplus}$ and it is unsupported if one of its positive literals is in $A^{\ominus}$. This is reflected in the definitions of $S_{\Pi}(A)$ and $\overline{S}_{\Pi}(A)$ below. Analogously, but with roles partly interchanged, $B_{\Pi}(A)$ and $\overline{B}_{\Pi}(A)$ define whether a body is *blocked* or *unblocked*, respectively.[2]

**Definition 1.** *Let $\Pi$ be a logic program and let $A$ be a partial assignment of $head(\Pi) \cup body(\Pi)$.*

*We define*

1. $T_{\Pi}(A) = \{h \in head(\Pi) \mid body(h) \cap A^{\oplus} \neq \emptyset\}$;
2. $\overline{T}_{\Pi}(A) = \{h \in head(\Pi) \mid body(h) \subseteq A^{\ominus}\}$;
3. $S_{\Pi}(A) = \{b \in body(\Pi) \mid b^{+} \subseteq A^{\oplus}\}$;
4. $\overline{S}_{\Pi}(A) = \{b \in body(\Pi) \mid b^{+} \cap A^{\ominus} \neq \emptyset\}$;
5. $B_{\Pi}(A) = \{b \in body(\Pi) \mid b^{-} \cap A^{\oplus} \neq \emptyset\}$;
6. $\overline{B}_{\Pi}(A) = \{b \in body(\Pi) \mid b^{-} \subseteq A^{\ominus}\}$.

We omit the subscript $\Pi$ whenever it is clear from the context. In what follows, we also adopt this convention for similar concepts without further notice.

With the above sets at hand, we can now specify *nomore++*'s forward propagation operator $\mathcal{P}$.

**Definition 2.** *Let $\Pi$ be a logic program and let $A$ be a partial assignment of $head(\Pi) \cup body(\Pi)$.*

*We define*

$$\mathcal{P}_{\Pi}(A) = A \sqcup (T(A) \cup (S(A) \cap \overline{B}(A)),\ \overline{T}(A) \cup \overline{S}(A) \cup B(A)) \ .$$

A head is assigned $\oplus$ if it belongs to $T(A)$, while a body must be supported as well as unblocked, namely, belong to $(S(A) \cap \overline{B}(A))$ in order to be assigned $\oplus$. Conversely, a body is marked $\ominus$, whenever it is unsupported or blocked; a head is $\ominus$, if it is not true, that is, in $\overline{T}(A)$. As we detail in the full paper, $\mathcal{P}$ amounts to Fitting's operator [6].

For example, let's apply $\mathcal{P}$ to $A_0 = (\{body(r_1)\}, \emptyset)$ on $\Pi_1$:

$$\begin{array}{ll} \mathcal{P}(A_0) = A_1 = (\{a, body(r_1)\}, \emptyset) & \text{by } T(A_0) \\ \mathcal{P}(A_1) = A_2 = (\{a, body(r_1)\}, \{body(r_2)\}) & \text{by } B(A_1) \\ \mathcal{P}(A_2) = A_3 = (\{a, body(r_1)\}, \{b, body(r_2)\}) & \text{by } \overline{T}(A_2) \end{array}$$

[2] We systematically use over-lining for indicating sets with antonymous contents. For example, $S$ and $\overline{S}$ stand for the sets of supported and *un*supported bodies.

Note that $A_3$ is closed under $\mathcal{P}$, that is, $\mathcal{P}(A_3) = A_3$.

For describing the saturated result of the combined application of operators, we need the following definition. Let $\mathcal{O}$ be an (often singleton) collection of operators and let $A$ be a partial assignment. Then, we denote by $\mathcal{O}^*$ the $\sqsubseteq$-smallest partial assignment containing $A$ and being closed under all operators in $\mathcal{O}$. In the above example, we have $\mathcal{P}^*(A_0) = A_3$.

For defining backward propagation, we have to look for the inverse of $\mathcal{P}$. For example, consider the definition of $T(A)$ and suppose $h \in head(\Pi) \cap A^{\oplus}$ whereas $body(h) \cap A^{\oplus} = \emptyset$. Hence, $h$ was not "produced" in $T(A)$. Yet there must be some body $b \in body(h)$ that is eventually assigned $\oplus$, otherwise $h$ cannot be true. However, this body can only be determined if all other bodies are already in $A^{\ominus}$. This leads us to the definition of $T^{\flat}_{\Pi}(A)$. Analogously, we can derive the following sets; see full paper for details.[3]

**Definition 3.** *Let $\Pi$ be a logic program and let $A$ be a partial assignment of $head(\Pi) \cup body(\Pi)$.*

*We define*

1. $T^{\flat}_{\Pi}(A) = \{b \mid b \in body(h), h \in head(\Pi) \cap A^{\oplus},\ body(h) \setminus \{b\} \subseteq A^{\ominus}\}$;
2. $\overline{T}^{\flat}_{\Pi}(A) = \{b \mid b \in body(h), h \in head(\Pi) \cap A^{\ominus}\}$;
3. $S^{\flat}_{\Pi}(A) = \{h \mid h \in b^{+}, b \in body(\Pi) \cap A^{\oplus}\}$;
4. $\overline{S}^{\flat}_{\Pi}(A) = \{h \mid h \in b^{+}, b \in body(\Pi) \cap A^{\ominus} \cap \overline{B}(A),\ b^{+} \setminus \{h\} \subseteq A^{\oplus}\}$;
5. $B^{\flat}_{\Pi}(A) = \{h \mid h \in b^{-}, b \in body(\Pi) \cap A^{\ominus} \cap S(A),\ b^{-} \setminus \{h\} \subseteq A^{\ominus}\}$;
6. $\overline{B}^{\flat}_{\Pi}(A) = \{h \mid h \in b^{-}, b \in body(\Pi) \cap A^{\oplus}\}$.

Combining the previous sets yields the following backward propagation operator $\mathcal{B}$.

**Definition 4.** *Let $\Pi$ be a logic program and let $A$ be a partial assignment of $head(\Pi) \cup body(\Pi)$.*

*We define*

$$\mathcal{B}_{\Pi}(A) = A \sqcup (T^{\flat}(A) \cup S^{\flat}(A) \cup B^{\flat}(A),\ \overline{T}^{\flat}(A) \cup \overline{S}^{\flat}(A) \cup \overline{B}^{\flat}(A))\ .$$

Adding the rule $b \leftarrow c$ to program $\Pi_1$ still gives $\mathcal{P}(A_3) = A_3$. However, due to the fact that head $b \in A_3^{\ominus}$ and thus body $\{c\} \in \overline{T}^{\flat}(A_3)$, we additionally get $\mathcal{B}^*(A_3) = A_3 \sqcup (\{d, \{not\ c\}\}, \{\{c\}, c, \{not\ d\}\})$; hence, $c$ must be false and $d$ must be true.

The next definition elucidates the notion of an *unfounded set* [7] in our context. Given an assignment $A$, the greatest unfounded set of heads and bodies, $U_{\Pi}(A)$, is defined in terms of all still potentially derivable atoms in $\overline{U}_{\Pi}(A)$.

**Definition 5.** *Let $\Pi$ be a logic program and let $A$ be a partial assignment of $head(\Pi) \cup body(\Pi)$.*

*We define*

$$U_{\Pi}(A) = \{b \in body(\Pi) \mid b^{+} \not\subseteq \overline{U}_{\Pi}(A)\} \cup \{h \in head(\Pi) \mid h \notin \overline{U}_{\Pi}(A)\}$$

*where* $\overline{U}_{\Pi}(A) = Cn((\Pi \setminus \{r \in \Pi \mid body(r) \in A^{\ominus}\})^{\emptyset})$.

[3] We use the superscript $^{\flat}$ to indicate sets used in backward propagation.

The set $\overline{U}(A)$ of potentially derivable atoms is formed by removing all rules whose body belongs to $A^{\ominus}$. The resulting subprogram is reduced with respect to the empty set so that we can compute its (possible) consequences by means of the $Cn$ operator. The counterpart of $\overline{U}(A)$ in *smodels*, known as *atmost*, amounts to

$$Cn((\Pi \setminus \{r \mid body(r)^+ \cap A^{\ominus} \neq \emptyset\})^{A^{\oplus} \cap head(\Pi)}) .$$

As bodies are not explicitly represented in *smodels*' assignments, we have to refer to atoms here.

Finally, we have the following operator $\mathcal{U}$ for falsifying all elements belonging to the greatest unfounded set (with respect to a given assignment).

**Definition 6.** *Let $\Pi$ be a logic program and let $A$ be a partial assignment of $head(\Pi) \cup body(\Pi)$.*

*We define*

$$\mathcal{U}_{\Pi}(A) = A \sqcup (\emptyset,\ U(A)) .$$

Consider program $\Pi_2$, obtained from $\Pi_1$ by adding rules

$$r_5 : e \leftarrow not\ a, not\ c, \quad r_6 : e \leftarrow f, not\ b, \quad r_7 : f \leftarrow e, \tag{3}$$

and assignment $A = (\emptyset, \{body(r_5)\})$. This is a common situation in *nomore++*, whenever bodies are taken to be choices. We then have $\overline{U}(A) = Cn((\Pi_2 \setminus \{r_5\})^{\emptyset}) = Cn(\{a \leftarrow,\ b \leftarrow,\ c \leftarrow,\ d \leftarrow,\ e \leftarrow f,\ f \leftarrow e\}) = \{a, b, c, d\}$, and thus we obtain $\mathcal{U}(A) = (\emptyset, \{body(r_5), e, body(r_6), f, body(r_7)\})$. As we detail in the full paper, the assignment $(\mathcal{PU})^*((\emptyset, \emptyset))$ amounts to the well-founded semantics of $\Pi$ [7].

Let us compare the previous to propagation in *smodels*. Basically, it is based on two functions, called *atleast* and *atmost*. While *atleast* computes deterministic consequences by forward and backward propagation, *atmost* detects unfounded sets. Together they allow for computing the well-founded semantics [7]. As done in [8], we represent *smodels*' assignments as sets of literals, where $not\ a$ means that $a$ is false. For brevity, we have to refer the reader for further formal details to [8]. We mention however that an inconsistent assignment is represented by *atleast* through the set of all literals and by *atmost* through the empty set.

**Theorem 1.** *Let $\Pi$ be a logic program. Let $A$ be a partial assignment of $head(\Pi) \cup body(\Pi)$ and let $X$ be a partial assignment of $head(\Pi)$ such that $(X^+, X^-) = (A^{\oplus}, A^{\ominus})$.* [4]

*Then, we have the following results.*

1. *If $atleast_{\Pi}(X) \subseteq atmost_{\Pi}(X)$, then*
   *(a) if $Y = atleast_{\Pi}(X)$ and $B = (\mathcal{PB})^*(A)$, then*

$$(Y^+, Y^-) = (B^{\oplus} \cap head(\Pi), B^{\ominus} \cap head(\Pi)).$$

   *(b) if $Y = atmost_{\Pi}(X)$ and $B = \mathcal{U}(\mathcal{P}(A))$, then*

$$(Y^+, Y^-) = (B^{\oplus} \cap head(\Pi), B^{\ominus} \cap head(\Pi)).$$

[4] Note that $(A^{\oplus} \cap body(\Pi), A^{\ominus} \cap body(\Pi)) = (\emptyset, \emptyset)$.

2. *If* $atmost_{\Pi}(X) \subset atleast_{\Pi}(X)$*, then*
   $(\mathcal{PBU})^*(A)$ *is undefined and vice versa.*

The last result shows that *nomore++*'s basic propagation operations $\mathcal{P}$, $\mathcal{B}$, and $\mathcal{U}$ are as powerful as those of *smodels*. The reason $\mathcal{P}$ is applied once before $\mathcal{U}$ in *(1b)* is that initially $A$ assigns no values to bodies in order to be comparable to *smodels*.

The first differences are encountered when it comes to making choices. While *smodels*' choices are restricted to literals, *nomore++* generally allows for assigning values to literals as well as bodies. This leads us to *nomore++*'s choice operator $\mathcal{C}$.

**Definition 7.** *Let* $\Pi$ *be a logic program and let* $A$ *be a partial assignment of* $head(\Pi) \cup body(\Pi)$.

*We define*

1. $\mathcal{C}^{\oplus}_{\Pi}(A) = (A^{\oplus} \cup \{x\}, A^{\ominus})$ *for some* $x \in (head(\Pi) \cup body(\Pi)) \setminus (A^{\oplus} \cup A^{\ominus})$;
2. $\mathcal{C}^{\ominus}_{\Pi}(A) = (A^{\oplus}, A^{\ominus} \cup \{x\})$ *for some* $x \in (head(\Pi) \cup body(\Pi)) \setminus (A^{\oplus} \cup A^{\ominus})$.

Note that the chosen object $x$ can be a head or a body.

The possibility of choosing among heads and bodies provides us with great flexibility. Notably, some choices have a higher information gain than others. On the one hand, setting a head to $\ominus$ yields more information than choosing some body to be $\ominus$. Negating some head $h$ by $\ominus$ implies that all bodies in $body(h)$ are false (via $\mathcal{B}$). Conversely, choosing a body to be $\ominus$ has generally no direct effect on the body's heads because there may be alternative rules (i.e. other bodies) sharing the same heads. Also, we normally gain no information on the constituent literals of the body. On the other hand, assigning $\oplus$ to bodies is superior to assigning $\oplus$ to heads. When choosing $\oplus$ for some head, we are generally unable to determine a corresponding body that justifies this choice and would then be assigned $\oplus$, too. Unlike this, choosing a body to be $\oplus$ allows us to infer the corresponding heads (by $\mathcal{P}$). Moreover, assigning $\oplus$ to a body $b$ implies that every literal in $b$ is true (by $\mathcal{B}$). The observation that assigning $\ominus$ to heads and $\oplus$ to bodies, respectively, subsumes the opposite assignments also fortifies *nomore++*'s lookahead strategy, detailed in Section 4.

Following [9], we characterize the process of answer set formation by a sequence of assignments.

**Theorem 2.** *Let* $\Pi$ *be a logic program and let* $A$ *be a total assignment of* $head(\Pi) \cup body(\Pi)$.

*Then,* $A^{\oplus} \cap head(\Pi)$ *is an answer set of* $\Pi$ *iff there exists a sequence* $(A^i)_{0 \leq i \leq n}$ *of assignments with the following properties:*

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

The purpose of the intersection $A^{\oplus} \cap head(\Pi)$ is to filter out the atoms forming an answer set. Different strategies can be shown to be sound and complete. For instance, the above result still holds after eliminating $\mathcal{B}$. For simplicity, we refer to these strategies by $(\mathcal{PBU})^*\mathcal{C}$ or $(\mathcal{PU})^*\mathcal{C}$, respectively.

## 4 Lookahead

We have seen that *nomore++*'s basic propagation is as powerful as that of *smodels*. An effective way of strengthening propagation is to use *lookahead*. [5] Apart from specifying *nomore++*'s lookahead, we demonstrate below that a *hybrid* lookahead strategy, incorporating heads and bodies, allows for stronger propagation than a *uniform* one using only either heads or bodies. Uniform lookahead is for instance used in *smodels* on literals and *noMoRe* on rules (comparable to bodies). However, we do not want to put more computational effort into hybrid lookahead than needed in the uniform case. The solution is simple: Assigning $\ominus$ to heads and $\oplus$ to bodies within lookahead is, in combination with propagation, powerful enough to compensate for the omitted assignments.

First of all, we operationally define our lookahead operator $\mathcal{L}$ as follows.

**Definition 8.** *Let $\Pi$ be a logic program and let $A$ be a partial assignment of $head(\Pi) \cup body(\Pi)$. Furthermore, let $\mathcal{O}$ be a collection of operators.*

*For $x \in (head(\Pi) \cup body(\Pi)) \setminus (A^{\oplus} \cup A^{\ominus})$, we define*

$$\ell_{\Pi}^{\oplus,\mathcal{O}}(A,x) = \begin{cases} (A^{\oplus}, A^{\ominus} \cup \{x\}) & \text{if } \mathcal{O}^*((A^{\oplus} \cup \{x\}, A^{\ominus})) \text{ is undefined} \\ A & \text{otherwise} \end{cases}$$

$$\ell_{\Pi}^{\ominus,\mathcal{O}}(A,x) = \begin{cases} (A^{\oplus} \cup \{x\}, A^{\ominus}) & \text{if } \mathcal{O}^*((A^{\oplus}, A^{\ominus} \cup \{x\})) \text{ is undefined} \\ A & \text{otherwise} \end{cases}$$

*For $X \subseteq head(\Pi) \cup body(\Pi)$, we define*

$$\mathcal{L}_{\Pi}^{\oplus,\mathcal{O}}(A,X) = \bigsqcup_{x \in X \setminus (A^{\oplus} \cup A^{\ominus})} \ell_{\Pi}^{\oplus,\mathcal{O}}(A,x)$$

$$\mathcal{L}_{\Pi}^{\ominus,\mathcal{O}}(A,X) = \bigsqcup_{x \in X \setminus (A^{\oplus} \cup A^{\ominus})} \ell_{\Pi}^{\ominus,\mathcal{O}}(A,x)$$

$$\mathcal{L}_{\Pi}^{\mathcal{O}}(A,X) = \mathcal{L}_{\Pi}^{\oplus,\mathcal{O}}(A,X) \sqcup \mathcal{L}_{\Pi}^{\ominus,\mathcal{O}}(A,X)$$

Lookahead works in a conflict-driven way, assigning a value whenever the opposite assignment leads to a conflict. The most powerful hybrid lookahead operator (relative to some operators $\mathcal{O}$) is $\mathcal{L}^{\mathcal{O}}(A, head(\Pi) \cup body(\Pi))$ as it includes all unassigned heads and bodies. However, taking up the above idea of restricting $\mathcal{L}$ to assigning $\ominus$ to heads and $\oplus$ to bodies only, yields an equally expressive operation that relies on significantly fewer applications of elementary lookahead by $\ell$ (see *3.* in Theorem 3). We show that explicitly assigning $\oplus$ to heads and $\ominus$ to bodies within $\ell$ is unnecessary as long as all conflicts are produced and their sources properly eliminated. In fact, the two sorts of assignments can be dealt with implicitly by $\mathcal{O}^*$, provided that $\mathcal{P}$ belongs to $\mathcal{O}$ and all operators in $\mathcal{O}$ are monotonic (like, for instance, $\mathcal{P}$, $\mathcal{B}$, and $\mathcal{U}$).

**Theorem 3.** *Let $\Pi$ be a logic program. Let $A$ be a partial assignment of $head(\Pi) \cup body(\Pi)$ and let*

$$B = \mathcal{P}(\mathcal{L}^{\oplus,\mathcal{O}}(A, body(\Pi))) \sqcup \mathcal{L}^{\ominus,\mathcal{O}}(A, head(\Pi)) \; .$$

*Then, for every collection $\mathcal{O}$ of $\sqsubseteq$-monotonic operators such that $\mathcal{P} \in \mathcal{O}$, we have*

[5] Often lookahead is additionally used for gathering heuristic values for choice operations. As with *dlv* and *smodels*, this information is exploited by *nomore++* as well.

$$\Pi_3^n = \Pi_1 \cup \left\{ \begin{array}{lll} r_5 : x \leftarrow not\ a, not\ b & r_6 : x \leftarrow not\ c, not\ d & \\ r_7 : y \leftarrow not\ a, not\ b & r_8 : y \leftarrow not\ c, not\ d & r_9 : y \leftarrow not\ x, not\ y \\ r_{10} : x \leftarrow i_1, \ldots, i_n & r_{2m+9} : i_m \leftarrow not\ j_m & r_{2m+10} : j_m \leftarrow not\ i_m \end{array} \right\}$$

$$\Pi_4^n = \Pi_1 \cup \left\{ \begin{array}{lll} r_5 : e \leftarrow not\ f & r_9 : \ y \leftarrow not\ c & r_{12+1} : i_1 \leftarrow x, y, z, not\ i_1 \\ r_6 : f \leftarrow not\ e & r_{10} : y \leftarrow not\ d & r_{12+n} : i_n \leftarrow x, y, z, not\ i_n \\ r_7 : x \leftarrow not\ a & r_{11} : z \leftarrow not\ e & r_{11+n+2m} : i_m \leftarrow not\ j_m \\ r_8 : x \leftarrow not\ b & r_{12} : z \leftarrow not\ f & r_{12+n+2m} : j_m \leftarrow not\ i_m \end{array} \right\}$$

**Fig. 1.** Lookahead examples $\Pi_3^n$ and $\Pi_4^n$, where $m = 1, \ldots, n$ and $\Pi_1 = \{r_1, \ldots, r_4\}$ from (2).

1. $\mathcal{L}^{\mathcal{O}}(A, head(\Pi)) \sqsubseteq B$;
2. $\mathcal{L}^{\mathcal{O}}(A, body(\Pi)) \sqsubseteq \mathcal{P}(B)$;
3. $\mathcal{L}^{\mathcal{O}}(A, head(\Pi) \cup body(\Pi)) \sqsubseteq \mathcal{P}(B)$.

Condition $\mathcal{P} \in \mathcal{O}$ stipulates that propagation (within lookahead) must be at least as powerful as Fitting's operator. Unlike this, the occurrences of $\mathcal{P}$ in $B$, *2.*, and *3.* are only of formal nature and needed for synchronizing heads and bodies. In practice, lookahead is interleaved with $\mathcal{P}$ anyway, since it is integrated into propagation, viz. $(\mathcal{PBUL})^*$. More importantly, *nomore++*'s restricted hybrid lookahead strategy, assigning $\ominus$ to heads and $\oplus$ to bodies only, faces approximately the same computational efforts as encountered in the uniform case since $2 * min\{|head(\Pi)|, |body(\Pi)|\} \leq |head(\Pi)| + |body(\Pi)| \leq 2 * max\{|head(\Pi)|, |body(\Pi)|\}$.[6]

Finally, let us demonstrate that *nomore++*'s hybrid lookahead strategy is in fact *strictly* more powerful than uniform ones. Consider Programs $\Pi_3^n$ and $\Pi_4^n$, given in Figure 1. In both programs, atoms $i_m$ and $j_m$ introduce choices, where $i_m$ is true whenever $j_m$ is false and vice versa. However, in the answer sets of $\Pi_3^n$ and $\Pi_4^n$, every $i_m$ is true and every $j_m$ false, and residual choices produce multiple answer sets. As we demonstrate in the following, a body-based lookahead decides every $i_m$ in $\Pi_3^n$ and a head-based one does the same in $\Pi_4^n$, whereas the opposite variants do not. Hence, only a hybrid lookahead effectively solves both programs, whereas an approach with uniform lookahead relies on "lucky" choices in case of either $\Pi_3^n$ or $\Pi_4^n$, respectively. When choices are "unlucky", uniform approaches may even face exponentially more choices than a hybrid lookahead, as our experiments in Section 6 demonstrate.

For readability, let us fix $\mathcal{O}$ to $(\mathcal{PBU})$ and drop the superscript. Our initial assignment is $A = (\emptyset, \emptyset)$. In Program $\Pi_3^n$, the bodies of rules $r_5, \ldots, r_9$ cannot become true. However, head-based lookahead cannot recognize this. Hence no atom can be decided and we obtain $(\mathcal{L}_{\Pi_3^n})^*(A, head(\Pi_3^n)) = (\emptyset, \emptyset) = A$. Body-based lookahead is superior here. It recognizes that the bodies of $r_5, \ldots, r_9$ have to be assigned $\ominus$. In particular, $body(r_9) = \{not\ x, not\ y\}$ must be assigned $\ominus$. Having this information, assigning $\oplus$ to any of the bodies $body(r_{2m+10}) = \{not\ i_m\}$ leads to a conflict. Thus, we get that $body(r_{10}) = \{i_1, \ldots, i_n\}$ must be true. Otherwise, operator $\mathcal{P}$ infers literals $not\ x$ and

[6] Note that $|body(\Pi)| \leq |\Pi|$ as several rules can share one body; in uniform cases, factor 2 accounts for assigning *both* truth values, $\oplus$ and $\ominus$, one after the other.

$not\ y$, thus contradicting that $body(r_9) = \{not\ x, not\ y\}$ is assigned $\ominus$. Consequently, we infer that all bodies $body(r_{2m+10}) = \{not\ i_m\}$ must be assigned $\ominus$. By application of $\mathcal{P}^*$, we can now decide all atoms $i_m$ to be true and all atoms $j_m$ to be false.

Next, consider $\Pi_4^n$, and let $B = (\mathcal{L}_{\Pi_4^n})^*(A, body(\Pi_4^n))$. Although all bodies $body(r_{12+m}) = \{x, y, z, not\ i_m\}$ belong to $B^{\ominus}$, body-based lookahead does not recognize that atoms $x$, $y$, and $z$ must be true due to rules $r_1, \ldots, r_{12}$. Here, head-based lookahead is superior. For $C = (\mathcal{L}^{\ominus}_{\Pi_4^n})^*(A, head(\Pi_4^n))$, we have $x \in C^{\oplus}$, $y \in C^{\oplus}$, and $z \in C^{\oplus}$. Furthermore, each $i_m$ is in $C^{\oplus}$ because, when trying to assign $\ominus$, $\mathcal{P}$ infers that $body(r_{12+m}) = \{x, y, z, not\ i_m\}$ is true, thus contradicting $i_m$ being false. All in all, head-based lookahead decides all atoms $i_m$, whereas a body-based approach only recognizes that certain bodies must not be true but is unable to determine a falsifying literal.

## 5 Design and implementation

The primary data structure in *nomore++* consists of a graph representing dependencies among heads and bodies.

**Definition 9.** *Let $\Pi$ be a logic program.*

*The body-head dependency graph $\Gamma_\Pi$ of $\Pi$ is a directed graph $(body(\Pi) \cup head(\Pi), E_0 \cup E_1 \cup E_2)$ with labeled arcs*

1. $E_0 = \{(h, b) \mid h \in head(\Pi), b \in body(\Pi), h \in b^+\}$;
2. $E_1 = \{(h, b) \mid h \in head(\Pi), b \in body(\Pi), h \in b^-\}$;
3. $E_2 = \{(b, h) \mid h \in head(\Pi), b \in body(\Pi), h \leftarrow b \in \Pi\}$.

With this, the concepts given in Definitions 1 and 3 can be directly mapped into graph concepts. To see this, observe that $\{h \mid (h, b) \in E_0\} = b^+$, $\{h \mid (h, b) \in E_1\} = b^-$, and $\{b \mid (b, h) \in E_2\} = body(h)$. For example, the graph-based counterpart of $T_\Pi(A)$ becomes $T_\Gamma(A) = \{h \mid pred^2_\Gamma(h) \cap A^{\oplus} \neq \emptyset\}$ where $pred^2_\Gamma(v) = \{v' \mid (v', v) \in E_2\}$. Hence, all operator specifications, except for that of $\mathcal{U}$, can be used unchanged for defining graph-based propagation. The same applies to the characterization of answer sets in Theorem 2.

The graph-based approach gives rise to further operators exploiting the graph structure. Among them, *nomore++* uses a preprocessing operator $\mathcal{I}$ assigning $\ominus$ to all bodies of $\Pi$ involved in a "self-blocking" situation.

**Definition 10.** *Let $\Gamma$ be the body-head dependency graph of logic program $\Pi$ and let $A$ be a partial assignment of $head(\Pi) \cup body(\Pi)$.*

*We define*

$$\mathcal{I}_\Gamma(A) = A \sqcup (\emptyset, \{b \in body(\Pi) \mid (h, b) \in E_1, (b, h) \in E_2 \text{ for some } h \in head(\Pi)\})\ .$$

This takes care of integrity constraints as well as rules $r$ where $head(r) \in body(r)^-$. More graph-specific issues are discussed in [10, 9].

The primary strategy of *nomore++* is to compute answer sets in a "support-driven" way. To this end, we rely on the graph structure for maintaining this as an invariant

property. To be precise, in logic programming terminology, [7] we call a set of rules $R$ *unfounded-free*, if

$$\{p \mid p \in body(r)^+, r \in R\} \subseteq Cn(R^\emptyset)\ . \tag{4}$$

Intuitively, unfounded-freeness guarantees that no rule in $R$ is justified by an unfounded set.[8] As a matter of fact, this allows us to ameliorate the implementation of operator $\mathcal{U}$. To see this, note that $\mathcal{U}$ leaves unfounded-free parts of assignments unchanged; because given a set $R$ satisfying (4) along with an assignment $A$ such that $body(R) \cap A^\ominus = \emptyset$, we have $U_R(A) = \emptyset$ (cf. Definition 5). Hence, such parts of an assignment may be ignored when applying $\mathcal{U}$. In our setting, we are therefore interested in operators guaranteeing that

$$R_\Pi(A) = \{r \in \Pi \mid body(r) \in A^\oplus\} \tag{5}$$

remains unfounded-free. The next result makes this precise for operator $\mathcal{P}$.

**Theorem 4.** *Let $\Pi$ be a logic program and let $A$ be a partial assignment of $head(\Pi) \cup body(\Pi)$ such that $head(\Pi) \cap A^\oplus \subseteq T(A)$.*

*If $R(A)$ is unfounded-free and $\mathcal{P}(A)$ is defined, then $R(\mathcal{P}(A))$ is unfounded-free.*

The requirement of $head(\Pi) \cap A^\oplus \subseteq T(A)$ stipulates that every assignment of $\oplus$ to a head is justified by a body in $A^\oplus$. This is needed since $R(A)$ is defined in (5) in terms of bodies.

As an example, consider $\Pi_2$ in (3) and assignment $A = (\{body(r_5), e\}, \emptyset)$. The rules in $R(A) = \{r_5\}$ are unfounded-free, that is, $body(r_5)^+ = \emptyset \subseteq Cn(\{e \leftarrow\}) = \{e\}$, and $e \in T(A)$. We obtain $\mathcal{P}^*(A) = (\{body(r_5), e, body(r_7), f\}, \emptyset)$, inducing again an unfounded-free set $R(\mathcal{P}^*(A)) = \{r_5, r_7\}$ since $body(r_5)^+ \cup body(r_7)^+ = \{e\} \subseteq Cn(\{e \leftarrow, f \leftarrow e\}) = \{e, f\}$.

Choice operator $\mathcal{C}^\oplus$ cannot guarantee unfounded-freeness as it assigns $\oplus$ to arbitrary heads and bodies. Hence, as an alternative, *nomore++* provides the following one.

**Definition 11.** *Let $\Pi$ be a logic program and let $A$ be a partial assignment of $head(\Pi) \cup body(\Pi)$.*

*We define*

1. $\mathcal{D}^\oplus_\Pi(A) = (A^\oplus \cup \{x\}, A^\ominus)$ *for some* $x \in (body(\Pi) \cap S(A)) \setminus (A^\oplus \cup A^\ominus)$;
2. $\mathcal{D}^\ominus_\Pi(A) = (A^\oplus, A^\ominus \cup \{x\})$ *for some* $x \in (body(\Pi) \cap S(A)) \setminus (A^\oplus \cup A^\ominus)$.

This operator differs from $\mathcal{C}$ in restricting its choices to *supported bodies*, as this guarantees unfounded-freeness.

**Theorem 5.** *Let $\Pi$ be a logic program and let $A$ be a partial assignment of $head(\Pi) \cup body(\Pi)$ such that $head(\Pi) \cap A^\oplus \subseteq T(A)$.*

*If $R(A)$ is unfounded-free, then $R(\mathcal{D}^\circ(A))$ is unfounded-free for $\circ \in \{\oplus, \ominus\}$.*

[7] See full paper for a graph-based characterization.

[8] In [11], a related notion of *unfounded-free interpretations* is used for characterizing answer sets of disjunctive programs. In our context, unfounded-freeness is used for describing partial assignments guaranteeing that none of their $\oplus$-assigned atoms becomes unfounded, no matter how such an assignment is extended.

Note that for heads there is no operator similar to $\mathcal{D}$ maintaining unfounded-freeness. A head having a true body, i.e. $body(h) \cap A^{\oplus} \neq \emptyset$, is already decided through $\mathcal{P}$. Therefore, it cannot be assigned $\ominus$ and thus is not a reasonable choice. On the other hand, if we concentrate on heads having a body that is supported but not already unblocked, i.e. there is a body $b \in (body(h) \cap S(A)) \setminus (B(A) \cup \overline{B}(A))$, generally, we cannot ensure that $b$ is still not blocked at some later step. That is, a head chosen to be true could become unfounded later on.

Unlike $\mathcal{P}$ and $\mathcal{D}$, backward propagation ($\mathcal{B}$) cannot maintain unfounded-freeness. To see this, suppose a body $b$ is assigned $\ominus$ by $\mathcal{D}$ and all but one, $h$, of its negative body literals have already been assigned $\ominus$, that is, $b^- \setminus A^{\ominus} = \{h\}$. Similarly, all but one, $b'$, of the bodies "deriving" $h$ are in $A^{\ominus}$, that is, $body(h) \setminus A^{\ominus} = \{b'\}$. Then, $h \in B^{\flat}(A)$, $b' \in T^{\flat}(\mathcal{B}(A))$, and backward propagation by $\mathcal{B}^*$ assigns $\oplus$ to $b'$, although $b'$ may not be unfounded-free. That is why we introduce at the implementation level a *weak* counterpart of $\oplus$, denoted by $\otimes$, indicating that some head or body must eventually be assigned $\oplus$ but its unfounded-freeness has not yet been established. In the implementation, only $\mathcal{P}$ and $\mathcal{D}$ assign $\oplus$, while operators like $\mathcal{B}$ and $\mathcal{C}$ can only assign $\otimes$ (or $\ominus$).[9] Any head or body in $A^{\otimes}$ can be turned into $\oplus$ by $\mathcal{P}$ without causing an undefined situation. So, by distinguishing two types of "true", we guarantee unfounded-freeness for objects assigned $\oplus$.

As mentioned above, this invariance allows us to greatly improve the implementation of operator $\mathcal{U}$. Rather than repeatedly re-establishing unfounded-freeness from scratch, the scope of $\overline{U}(A)$ (in Definition 5) can be restricted to $(head(\Pi) \cup body(\Pi)) \setminus (A^{\oplus} \cup A^{\ominus})$, while taking the support of $A^{\oplus}$ for granted. In other words, $\overline{U}(A)$ (and so $U(A)$) can be restricted to heads and bodies being either unassigned or assigned $\otimes$. For brevity, we refrain from giving a formal definition and refer to this enhancement of $\mathcal{U}$ for further reference by $\mathcal{V}$. Also, our graph-based representation allows us to implement this in a "lazy fashion": The nodes in the graph are initially marked as supported, if (roughly) they belong to $\overline{U}((\emptyset, \emptyset))$. Whenever such a node is assigned $\ominus$, this information is propagated to the connecting nodes in order to find unfounded ones.

All in all, enforcing unfounded-freeness as an invariant on $A^{\oplus}$ allows for an incremental implementation of operator $\mathcal{U}$. In fact, these restrictions do not sacrifice completeness. As detailed in the full version, Theorem 2 is still valid after replacing operators $\mathcal{C}$ and $\mathcal{U}$ by $\mathcal{D}$ and $\mathcal{V}$, respectively, subject to the extension of assignments by $\otimes$. Furthermore, any assignment $A$ produced by strategy $(\mathcal{PBV})^*\mathcal{D}$ satisfies the precondition, $head(\Pi) \cap A^{\oplus} \subseteq T(A)$, of Theorems 4 and 5.

Finally, let us mention that *nomore++* is implemented in C++ and uses *lparse* as parser. *nomore++* facilitates the easy use of different sets of operators. For instance, if called with command line option `-op 'D:(PBV)*'`, it uses operator $\mathcal{D}$ for choices and $(\mathcal{PBV})^*$ for propagation. One can also determine which set $\mathcal{O}$ of operators to use for the lookahead operator $\mathcal{L}$ via command line option `-laop`. The system is freely available at [12].

[9] Please note that $\mathcal{P}$ retains $\otimes$ when propagating from $\otimes$. Also, a body $b$ cannot be chosen by $\mathcal{D}$ if some $h \in b^+$ is in $A^{\otimes}$.

| $HC_n$ | *smodels* | *nomore++* $_{(PBVL_b)^*D}$ | *nomore++* $_{(PBVL_{bh})^*D}$ | *smodels* | *nomore++* $_{(PBVL_b)^*D}$ | *nomore++* $_{(PBVL_{bh})^*D}$ |
|---|---|---|---|---|---|---|
| 3 | 1 (0.00) | 1 (0.00) | 1 (0.00) | 1 (0.00) | 1 (0.00) | 1 (0.00) |
| 4 | 2 (0.01) | 2 (0.01) | 2 (0.00) | 5 (0.00) | 5 (0.00) | 5 (0.00) |
| 5 | 3 (0.00) | 3 (0.00) | 3 (0.01) | 26 (0.00) | 23 (0.02) | 23 (0.02) |
| 6 | 4 (0.01) | 4 (0.01) | 4 (0.01) | 305 (0.02) | 119 (0.11) | 119 (0.11) |
| 7 | 30 (0.01) | 5 (0.02) | 5 (0.02) | 4814 (0.38) | 719 (0.83) | 719 (0.85) |
| 8 | 8 (0.00) | 6 (0.03) | 6 (0.03) | 86364 (7.29) | 5039 (7.40) | 5039 (7.60) |
| 9 | 48 (0.01) | 7 (0.05) | 7 (0.05) | 1864470 (177.91) | 40319 (73.94) | 40319 (76.09) |
| 10 | 1107 (0.18) | 8 (0.08) | 8 (0.08) | n/a | 362879 (818.73) | 362879 (842.57) |
| 11 | 18118 (2.88) | 9 (0.13) | 9 (0.12) | n/a | n/a | n/a |
| 12 | 398306 (65.29) | 10 (0.19) | 10 (0.20) | n/a | n/a | n/a |
| 13 | n/a | 11 (0.29) | 11 (0.30) | n/a | n/a | n/a |

**Table 1.** Experiments for $HC_n$ computing (a) one answer set; (b) all answer sets

## 6 Selected experimental results

Because of space limitations, we confine our listed experiments to selected benchmarks illustrating the major features of *nomore++*. A complete evaluation, including further ASP solvers, like *dlv*, *assat*, and *cmodels*, can be found at the ASP benchmarking site [13]. All tests were run on an AMD Athlon 1.4GHz PC with 512MB RAM. As in the context of [13], a memory limit of 256MB, as well as a time limit of 900s was enforced. All results are given in terms of number of choices and seconds (in parentheses), reflecting the average of 10 runs.

Let us note that, due to the fairly early development state of *nomore++*, its base speed is still inferior to more mature ASP solvers, like *smodels* or *dlv*. This can for instance be seen in the results of the "Same Generation" benchmark, where *smodels* outperforms *nomore++* roughly by a factor of two (cf. [13]). [10] Despite this, the selected experiments demonstrate the computational value of crucial features of *nomore++* and provide an indication of the prospect of the overall approach.

In all test series, we ran *smodels* with its (head-based) lookahead. For a complement, we also give tests for *nomore++* with body-based lookahead $\mathcal{L}^{(\mathcal{PBV})}(A, body(\Pi))$, abridged $\mathcal{L}_b$. The tests with *nomore++*'s hybrid lookahead rely on $\mathcal{L}^{\oplus,(\mathcal{PBV})}(A, body(\Pi)) \sqcup \mathcal{L}^{\ominus,(\mathcal{PBV})}(A, head(\Pi))$, abbreviated by $\mathcal{L}_{hb}$. For illustrating *nomore++*'s support-driven strategy, we give in Table 1 results obtained on classical Hamiltonian cycle problems on complete graphs ($HC_n$), both for the first and for all answer sets. While *nomore++* does not make any wrong choices leading to a linear performance in Table 1(a), *smodels* makes an exponential number of choices, even for finding the first answer set. *nomore++*'s support-driven strategy enforces that rules are chained in the appropriate way. This is nicely reflected in $HC_n$ examples, where solutions are characterized by unfounded-free sets of rules. We note that, on $HC_n$ ex-

[10] Other apt benchmarks are "Factoring" and "Schur Numbers" (cf. [13]); in both cases *smodels* still outperforms *nomore++* by an order of magnitude.

| $\Pi_3^n$ | *smodels* | *nomore++* $_{(PBVL_b)^*D}$ | *nomore++* $_{(PBVL_{bh})^*D}$ | $\Pi_4^n$ | *smodels* | *nomore++* $_{(PBVL_b)^*D}$ | *nomore++* $_{(PBVL_{bh})^*D}$ |
|---|---|---|---|---|---|---|---|
| 1 | 5 (0.00) | 3 (0.00) | 3 (0.00) | 1 | 3 (0.00) | 5 (0.00) | 3 (0.01) |
| 2 | 15 (0.00) | 4 (0.00) | 4 (0.00) | 2 | 6 (0.00) | 11 (0.01) | 6 (0.01) |
| 3 | 38 (0.00) | 5 (0.01) | 5 (0.01) | 3 | 9 (0.00) | 43 (0.01) | 9 (0.01) |
| 4 | 137 (0.00) | 6 (0.00) | 6 (0.00) | 4 | 12 (0.00) | 120 (0.03) | 12 (0.01) |
| 5 | 460 (0.01) | 7 (0.01) | 7 (0.01) | 5 | 15 (0.00) | 269 (0.07) | 15 (0.01) |
| 6 | 1447 (0.02) | 8 (0.01) | 8 (0.01) | 6 | 18 (0.00) | 1158 (0.27) | 18 (0.01) |
| 7 | 4738 (0.06) | 9 (0.00) | 9 (0.01) | 7 | 21 (0.00) | 5285 (1.15) | 21 (0.02) |
| 8 | 14725 (0.19) | 10 (0.00) | 10 (0.01) | 8 | 24 (0.01) | 15222 (3.27) | 24 (0.02) |
| 9 | 46230 (0.58) | 11 (0.01) | 11 (0.01) | 9 | 27 (0.00) | 51377 (10.88) | 27 (0.02) |
| 10 | 143283 (1.82) | 12 (0.01) | 12 (0.01) | 10 | 30 (0.00) | 602312 (118.75) | 30 (0.03) |
| 11 | 440234 (5.70) | 13 (0.01) | 13 (0.01) | 11 | 33 (0.00) | 3284697 (645.62) | 33 (0.03) |
| 12 | 1354823 (17.85) | 14 (0.01) | 14 (0.01) | 12 | 36 (0.01) | n/a | 36 (0.03) |
| 13 | 4147650 (55.63) | 15 (0.01) | 15 (0.01) | 13 | 39 (0.01) | n/a | 39 (0.04) |
| 14 | 12667755 (173.21) | 16 (0.01) | 16 (0.01) | 14 | 42 (0.01) | n/a | 42 (0.04) |
| 15 | 38647666 (538.24) | 17 (0.01) | 17 (0.02) | 15 | 45 (0.01) | n/a | 45 (0.04) |
| 16 | n/a | 18 (0.01) | 18 (0.01) | 16 | 48 (0.01) | n/a | 48 (0.04) |

**Table 2.** Experiments computing one answer set for (a) $\Pi_3^n$; (b) $\Pi_4^n$

amples, *dlv* performs much better regarding time (cf. [13]); the different concept of "choice points" makes them incomparable in this respect.

The results in Table 2(a) and (b) aim at supporting *nomore++*'s hybrid lookahead; they are obtained on (extensions of the) lookahead examples $\Pi_3^n$ and $\Pi_4^n$ from Figure 1. The exact programs as well as additional measurements, like those for computing all answer sets, can be found at [12]. We see that a hybrid approach is superior to both kinds of uniform lookahead. *smodels* employs a head-based lookahead, leading to a good performance on examples $\Pi_4^n$, yet a bad one on $\Pi_3^n$. The converse is true when restricting *nomore++* to lookahead on bodies only. *nomore++* with hybrid lookahead performs choice-point-optimal on both types of examples. Also, a comparison of the two *nomore++* variants shows that a hybrid lookahead does not lead to any computational overhead. Note that these examples are designed to show the effect of lookahead. Depending on heuristics, a better performance may be obtainable without lookahead. Note that *dlv* performs worse than either *smodels* or *nomore++* on some of the $\Pi_4^n$ and $\Pi_3^n$ benchmarks. [11]

## 7 Discussion

We have presented a new ASP solver, along with its underlying theory, design and some experimental results. Its distinguishing features are (i) the extended concept of an

[11] *dlv* handles examples of form $\Pi_4^n$ only up to $n = 8$ (computing one answer set) and $\Pi_3^n$ examples up to $n = 11$ (computing all answer sets; see [12] for details). Also, *nomore++* outperforms *dlv* on some other relevant benchmarks, such as "Schur Numbers", by one order of magnitude (see [13]).

assignment, including atoms as well as bodies, (ii) the more powerful lookahead operation, and (iii) its support-driven strategy. We draw from previous work on the *noMoRe* system [5], whose methodology for answer set computation is based on "coloring" the rule dependency graph (RDG) of a program. It therefore pursues a rule-based approach, which amounts to restricting the domain of assignments to $body(\Pi)$. The functionality of *noMoRe* was described in [9] by graph-theoretical operators similar to $\mathcal{P}$, $\mathcal{U}$, and $\mathcal{C}$. *nomore++*'s operators for backward propagation ($\mathcal{B}$) and lookahead ($\mathcal{L}$) have been presented here for the first time. In general, operator-based specifications facilitate formal comparisons between techniques used by different ASP solvers. Operators capturing propagation in `dlv` are given in [14]. Pruning operators based on Fitting's [6] and well-founded semantics [7] are investigated in [15]. The full paper contains a detailed comparison of these operators.

*dlv* and *smodels* pursue a purely literal-based approach, which boils down to restricting the domain of assignments to $head(\Pi)$. Interestingly, *smodels*' implementation relies on a rule-head dependency graph, in which rules and atoms are connected via pointers. This data structure is more redundant than the body-head dependency graph, since the number of unique bodies in a program is always less or equal to the number of rules.[12] Moreover, *smodels* does not take the concept of support into account. Contrastingly, *dlv* uses a partly support-driven strategy for selecting choices (so-called *possibly-true literals*). Also, *dlv* uses a truth value "must be true", which is similar to $\otimes$. Interestingly, its choice operator assigns either $\oplus$ or $\otimes$, depending on the support status of the chosen literal. Our empirical studies show definite prospect of *nomore++* but also reveal an unfledged state of development. A major concern in future work will be to close the engineering gap to the mature ASP solvers *smodels* and *dlv*.

## References

1. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence **138** (2002) 181–234
2. Leone, N., Faber, W., Pfeifer, G., Eiter, T., Gottlob, G., Koch, C., Mateis, C., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Transactions on Computational Logic (2005) To appear.
3. Lin, F., Zhao, Y.: Assat: computing answer sets of a logic program by sat solvers. Artificial Intelligence **157** (2004) 115–137
4. Lierler, Y., Maratea, M.: Cmodels-2: Sat-based answer sets solver enhanced to non-tight programs. In Lifschitz, V., Niemelä, I., eds.: Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'04). Volume 2923 of Lecture Notes in Computer Science., Springer-Verlag (2004) 346–350
5. Anger, C., Konczak, K., Linke, T.: `noMoRe`: A system for non-monotonic reasoning under answer set semantics. In Eiter, T., Faber, W., Truszczyński, M., eds.: Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning, Springer-Verlag (2001) 406–410
6. Fitting, M.: Fixpoint semantics for logic programming: A survey. Theoretical Computer Science **278** (2002) 25–51

[12] Measuring over 241 ground programs in [13], the ratio of the number of distinct bodies over the number of rules is $0.41$.

7. van Gelder, A., Ross, K., Schlipf, J.: The well-founded semantics for general logic programs. Journal of the ACM **38** (1991) 620–650
8. Niemelä, I., Simons, P.: Efficient implementation of the well-founded and stable model semantics. In Maher, M., ed.: Proceedings of the Joint International Conference and Symposium on Logic Programming, The MIT Press (1996) 289–303
9. Konczak, K., Linke, T., Schaub, T.: Graphs and colorings for answer set programming: Abridged report. In Vos, M.D., Provetti, A., eds.: Proceedings of the Second International Workshop on Answer Set Programming (ASP'03). Volume 78., CEUR Workshop Proceedings (2003) 137–150
10. Linke, T., Sarsakov, V.: Suitable graphs for answer set programming. In Baader, F., Voronkov, A., eds.: Proceedings of the Eleventh International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'04). Volume 3452 of Lecture Notes in Computer Science., Springer-Verlag (2005) 154–168
11. Leone, N., Rullo, P., Scarcello, F.: Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation. Information and Computation **135** (1997) 69–112
12. (`http://www.cs.uni-potsdam.de/nomore`)
13. (`http://asparagus.cs.uni-potsdam.de`)
14. Faber., W.: Enhancing Efficiency and Expressiveness in Answer Set Programming Systems. Dissertation, Technische Universität Wien (2002)
15. Calimeri, F., Faber, W., Leone, N., Pfeifer, G.: Pruning operators for answer set programming systems. In Benferhat, S., Giunchiglia, E., eds.: Proceedings of the ninth International Workshop on Non-Monotonic Reasoning (NMR'04). (2002) 200–209