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 computational strategy of maintaining unfounded-freeness. We 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 of logic programs' rules 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 more effective propagation than previous approaches. Finally, we detail a computational strategy of maintaining "unfounded-freeness".

We empirically show that, thanks to its hybrid approach, *nomore++* outperforms *smodels* on relevant benchmarks. In fact, 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* and many of its distinguishing features are devised for dealing with the more expressive class of disjunctive logic programs. Also, *smodels* and *nomore++* share the same concept of "choice points", on which parts of our experiments rely upon.

The paper is organised 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++*. 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 strategy of keeping assignments unfounded-free. Finally, we provide selected experimental results backing up our claims.

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

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 $body(r) = \{p_1, \ldots, p_m, not\ p_{m+1}, \ldots, not\ p_n\}$ be 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 $body(r)$, we then get $body(r)^+ = \{p_1, \ldots, p_m\}$ and $body(r)^- = \{p_{m+1}, \ldots, p_n\}$.

A logic program Π is called *basic* if $body(r)^- = \emptyset$ for all $r \in \Pi$. The *reduct*, Π^X, of Π relative to a set X of atoms is defined by

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

A set X of atoms is closed under a basic program Π 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 Π. A set X of atoms is an *answer set* of a logic program Π if $Cn(\Pi^X) = X$.

As an example, consider program Π_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 Π, we write $head(\Pi) = \{head(r) \mid r \in \Pi\}$ and $body(\Pi) = \{body(r) \mid r \in \Pi\}$. We further extend this notation: For $h \in head(\Pi)$, define $body(h) = \{body(r) \mid r \in \Pi, head(r) = h\}$.

Without loss of generality, we restrict ourselves to programs Π 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 Π can be transformed into such a format, exploiting the fact that no atom in $(\mathcal{A} \setminus head(\Pi))$ is contained in any answer set of Π.

3 Operational specification

We provide in this section a detailed operational specification of *nomore++*. The firm understanding of *nomore++*'s propagation mechanisms serves as a basis for formal comparisons with techniques used by *smodels* or *dlv*. We indicate how the operations applied by *nomore++* are related to well-known propagation principles, in particular, showing that our basic propagation operations are as powerful as those of *smodels* (cf. Theorem 1). Beyond this, the hybrid approach of *nomore++* allows for more flexible choices, in particular, leading to a more powerful lookahead, as we detail in Section 4.

We consider assignments that map heads and bodies of a program Π into $\{\oplus, \ominus\}$, indicating whether a head or body is true or false, respectively. Such assignments are extended in comparison to those used in literal-based solvers, such as *smodels* and *dlv*, or rule-based solvers, such as *noMoRe*. Formally, a (partial) assignment is a partial

mapping $A : head(\Pi) \cup body(\Pi) \rightarrow \{\oplus, \ominus\}$. For simplicity, we often represent such an assignment A as a pair $(A^{\oplus}, A^{\ominus})$, 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 $(A^{\oplus} \cup B^{\oplus}, A^{\ominus} \cup B^{\ominus})$.

Forward propagation in *nomore++* can be divided into two sorts. Head-oriented propagation assigns $\oplus$ to a head if one of its associated bodies belongs to $A^{\oplus}$, it assigns $\ominus$ whenever all of a head's bodies are in $A^{\ominus}$. This kind of propagation is captured by sets $T_{\Pi}(A)$ 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 $A^{\oplus}$, it is *unsupported* if one of its positive literals is in $A^{\ominus}$. This is reflected by sets $S_{\Pi}(A)$ and $\overline{S}_{\Pi}(A)$ below. Analogously, but with roles partly interchanged, sets $B_{\Pi}(A)$ and $\overline{B}_{\Pi}(A)$ define whether a body is *blocked* or *unblocked*, respectively.[2]

Definition 1. *Let Π 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 Π whenever it is clear from the context. In what follows, we also adopt this convention for similar concepts without further notice.

Based on the above sets, we define forward propagation operator $\mathcal{P}$ as follows.

Definition 2. *Let Π 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 $T(A)$; a body must be supported as well as unblocked, namely, belong to $S(A) \cap \overline{B}(A)$. Conversely, a body is assigned $\ominus$ whenever it is unsupported or blocked, i.e. in $\overline{S}(A) \cup B(A)$; a head must be in $\overline{T}(A)$.

For example, let us apply $\mathcal{P}$ to $A_0 = (\{body(r_1)\}, \emptyset)$ on Π_1:

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

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

For describing the saturated result of operators' application, we need the following definition. Let $\mathcal{O}$ be a collection (possibly a singleton) of operators and let A be a partial

[2] We systematically use over-lining for indicating sets with antonymous contents.

assignment. Then, we denote by $\mathcal{O}^*(A)$ the $\sqsubseteq$-smallest partial assignment containing A and being closed under all operators in $\mathcal{O}$. In the above example, we get $\mathcal{P}^*(A_0) = A_3$.

Backward propagation can be viewed as an inversion of $\mathcal{P}$. For example, consider the definition of $T(A)$ and suppose $h \in \mathit{head}(\Pi) \cap A^{\oplus}$ whereas $\mathit{body}(h) \cap A^{\oplus} = \emptyset$, that is, no body of any rule with head h has been assigned $\oplus$ so far. Hence, h is not "produced" by $T(A)$. Yet there must be some body $b \in \mathit{body}(h)$ 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 $T^{\flat}_{\Pi}(A)$.[3] Analogously, we can derive the following sets.[4]

Definition 3. *Let Π be a logic program and let A be a partial assignment of $\mathit{head}(\Pi) \cup \mathit{body}(\Pi)$. We define*

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

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

Definition 4. *Let Π be a logic program and let A be a partial assignment of $\mathit{head}(\Pi) \cup \mathit{body}(\Pi)$. 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 Π_1 still gives $\mathcal{P}(A_3) = A_3$. Due to the fact that $b \in A_3^{\ominus}$, iterated application of $\mathcal{B}$ additionally yields:

$$\begin{array}{lllll}
\mathcal{B}(A_3) = & A_4 & = A_3 \sqcup (\emptyset, \{\{c\}\}) & \text{by} & \overline{T}^{\flat}(A_3) \\
\mathcal{B}(A_4) = & A_5 & = A_3 \sqcup (\emptyset, \{\{c\}, c\}) & \text{by} & \overline{S}^{\flat}(A_4) \\
\mathcal{B}(A_5) = & A_6 & = A_3 \sqcup (\emptyset, \{\{c\}, c, \mathit{body}(r_3)\}) & \text{by} & \overline{T}^{\flat}(A_5) \\
\mathcal{B}(A_6) = & A_7 & = A_3 \sqcup (\{d\}, \{\{c\}, c, \mathit{body}(r_3)\}) & \text{by} & B^{\flat}(A_6) \\
\mathcal{B}(A_7) = \mathcal{B}^*(A_3) & & = A_3 \sqcup (\{d, \mathit{body}(r_4)\}, \{\{c\}, c, \mathit{body}(r_3)\}) & \text{by} & T^{\flat}(A_7)
\end{array}$$

The next definition elucidates the notion of an unfounded set [6] in our context. Given an assignment A, the *greatest unfounded set* of heads and bodies, $U_{\Pi}(A)$, is defined in terms of the still potentially derivable atoms in $\overline{U}_{\Pi}(A)$.

Definition 5. *Let Π be a logic program and let A be a partial assignment of $\mathit{head}(\Pi) \cup \mathit{body}(\Pi)$. We define*

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

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

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

[4] The relation between $\mathcal{P}$ and $\mathcal{B}$ will be detailed in the full paper.

The set $\overline{U}(A)$ of potentially derivable atoms is formed by removing all rules whose bodies belong to $A^{\ominus}$. The resulting subprogram is reduced with respect to the empty set so that we can compute its (potential) consequences by means of the Cn operator, as defined for basic programs in Section 2.

The following operator $\mathcal{U}$ falsifies all elements in a greatest unfounded set.

Definition 6. *Let Π 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 Π_2, obtained from Π_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 $A = (\emptyset, \{body(r_5)\})$.[5] 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 a program's well-founded semantics [6].

Let us compare the introduced operators to propagation in *smodels*, which is based on two functions, called *atleast* and *atmost*. Function *atleast* computes deterministic consequences by forward and backward propagation, Function *atmost* is the counterpart of $\overline{U}(A)$ and amounts to $Cn((\Pi \setminus \{r \mid body(r)^+ \cap A^{\ominus} \neq \emptyset\})^{A^{\oplus} \cap head(\Pi)})$. In [1], *smodels*' assignments are represented as sets of literals. Although we refrain from giving a formal definition, we however mention that *atleast* bounds the set of true literals from "below" and that *atmost* bounds the set of true atoms from "above".

Theorem 1. *Let Π be a logic program. Let X be a partial assignment of $head(\Pi)$ and let A be a partial assignment of $head(\Pi) \cup body(\Pi)$ such that $(A^{\oplus}, A^{\ominus}) = (X^+, X^-)$.*[6] *Then, we have the following results.*

1. *Let $Y = atleast(\Pi, X)$ and $B = (\mathcal{PB})^*(A)$.*
   If $Y^+ \cap Y^- = \emptyset$ and $B^{\oplus} \cap B^{\ominus} = \emptyset$, then $(Y^+, Y^-) = (B^{\oplus} \cap head(\Pi), B^{\ominus} \cap head(\Pi))$; otherwise, $Y^+ \cap Y^- \neq \emptyset$ and $B^{\oplus} \cap B^{\ominus} \neq \emptyset$.
2. *Let $Y = X \sqcup (\emptyset, head(\Pi) \setminus atmost(\Pi, X))$ and $B = \mathcal{U}(\mathcal{P}(A))$.*
   If $Y^+ \cap Y^- = \emptyset$ and $B^{\oplus} \cap B^{\ominus} = \emptyset$, then $(Y^+, Y^-) = (B^{\oplus} \cap head(\Pi), B^{\ominus} \cap head(\Pi))$; otherwise, $Y^+ \cap Y^- \neq \emptyset$ and $B^{\oplus} \cap B^{\ominus} \neq \emptyset$.

The above results show that *nomore++*'s basic propagation operations $\mathcal{P}$, $\mathcal{B}$, and $\mathcal{U}$ are as powerful as those of *smodels*. The reason why $\mathcal{P}$ is applied once in *2.* is that initially A assigns no values to bodies in order to be comparable to *smodels*' assignment X.

Concluding with basic propagation, we mention that $\mathcal{P}$ corresponds to Fitting's operator [7], $(\mathcal{PB})$ coincides to unit propagation on a program's completion [8], $(\mathcal{PU})$ amounts to propagation via well-founded semantics [6], and $(\mathcal{PBU})$ matches *smodels*' propagation, that is, well-founded semantics enhanced by backward propagation.

[5] The situation that a body is in $A^{\ominus}$ without belonging to $\overline{S}(A) \cup B(A)$ is common in *nomore++*, as bodies can be taken as choices.

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

The first differences to well-known approaches are encountered at choices. In *smodels* and *dlv*, choices are restricted to heads; *noMoRe* chooses on rules (comparable to bodies) only. Unlike this, *nomore++* generally allows for choosing to assign values to heads as well as bodies, and we define *nomore++*'s choice operator $\mathcal{C}$ as follows.

Definition 7. *Let Π be a logic program, let A be a partial assignment of $head(\Pi) \cup body(\Pi)$, and let $X \subseteq head(\Pi) \cup body(\Pi)$. We define*

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

The set X delineates the set of possible choices. In general, the chosen object $x \in X$ can be any unassigned head or 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 we choose $\oplus$ for a body, we infer that its heads must be assigned $\oplus$ as well (via $\mathcal{P}$). Moreover, assigning $\oplus$ to a body b implies that every literal in b is true (via $\mathcal{B}$). Unlike this, choosing $\oplus$ for some head does generally not allow to determine a corresponding body that justifies this choice and would then be assigned $\oplus$, too. 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 characterise the process of answer set formation by a sequence of assignments.

Theorem 2. *Let Π be a logic program, let A be a total assignment of $head(\Pi) \cup body(\Pi)$, and let $X = head(\Pi) \cup body(\Pi)$. Then, $A^{\oplus} \cap head(\Pi)$ is an answer set of Π 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. $A^{i+1} = (\mathcal{PBU})^*(\mathcal{C}^\circ(A^i, X))$ *for some* $\circ \in \{\oplus, \ominus\}$ *and* $0 \leq i < n$;
3. $A^n = A$.

The intersection $A^{\oplus} \cap head(\Pi)$ accomplishes a projection to heads and thus to the atoms forming an answer set. Many 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. We also drop superscripts $^{\oplus}$ and $^{\ominus}$ from $\mathcal{C}$ when referring to either case.) As with computational strategies, alternative choices, expressed by X, are possible. For example, Theorem 2 also holds for $X = head(\Pi)$ or $X = body(\Pi)$, respectively, mimicking a literal-based approach such as *smodels*' one or a rule-based approach as the one of *noMoRe*. A further restriction of choices is discussed in Section 5.

Although we cannot provide the details here, it is noteworthy to mention that allowing $X = head(\Pi) \cup body(\Pi)$ as choices leads to an exponentially stronger proof

system (in terms of proof complexity [10], i.e. minimal proofs for unsatisfiability) in comparison to either $X = head(\Pi)$ or $X = body(\Pi)$. The comparison between different proof systems and proof complexity results will be key issues in the full paper.

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*.[7] 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 in *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 Π 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:

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

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

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

Observe that, according to the above definition, elementary lookahead ℓ can only be applied to an unassigned head or body x. For such an x, ℓ tests whether assigning and propagating a value leads to a conflict. If so, the opposite value is assigned. We stipulate x to be unassigned because the intended purpose of lookahead is gaining information from imminent conflicts when basic propagation is stuck, hence the name "lookahead".

Our lookahead operator $\mathcal{L}$ can be parametrised in several ways. First, one can decide on a set $X \subseteq head(\Pi) \cup body(\Pi)$ to apply ℓ to. Second, either $\oplus$, $\ominus$, or both of them, one after the other, can be temporarily assigned and propagated. Third, the collection $\mathcal{O}$ determines the propagation operators to be applied inside lookahead, which can be

[7] Note that we consider lookahead primarily as a propagation operation, such as $\mathcal{P}$, $\mathcal{B}$, and $\mathcal{U}$. Supplementary, lookahead is often also used for gathering heuristic values for the selection of choices. As with *smodels* and *dlv*, this information is exploited by *nomore++* as well.

different from the ones used outside lookahead. The general definition allows us to describe and to compare different variants of lookahead.

In what follows, we detail *nomore++*'s hybrid lookahead on heads and bodies and show that it is strictly stronger than uniform lookahead on only either heads or bodies, without being computationally more expensive. To start with, observe that *full* hybrid lookahead by $\mathcal{L}^{\mathcal{O}}(A, head(\Pi) \cup body(\Pi))$ is the most powerful lookahead operation with respect to some $\mathcal{O}$. That is, anything inferred by a restricted lookahead is also inferred by full hybrid lookahead. Given that full hybrid lookahead has to temporarily assign both values, $\oplus$ and $\ominus$, to each unassigned head and body, it is also the computationally most expensive lookahead operation. In the worst case, there might be $2 * (|head(\Pi)| + |body(\Pi)|)$ applications of ℓ without inferring anything.

The high computational cost of full hybrid lookahead is the reason why *nomore++* applies a *restricted* hybrid lookahead. Despite the restrictions, *nomore++*'s hybrid lookahead does not sacrifice propagational strength and is in combination with propagation as powerful as full hybrid lookahead (see *3.* in Theorem 3 below). The observations made on choices in the previous section provide an explanation on how a more powerful hybrid lookahead operation can be obtained without reasonably increasing the computational cost in comparison to uniform lookahead on only either heads or bodies: Assigning $\ominus$ to a head subsumes assigning $\ominus$ to one of its bodies, assigning $\oplus$ to a body subsumes assigning $\oplus$ to one of its heads. That is why *nomore++*'s hybrid lookahead applies $\ell^{\ominus,\mathcal{O}}$ to heads and $\ell^{\oplus,\mathcal{O}}$ to bodies only. Provided that $\mathcal{P}$ belongs to $\mathcal{O}$ and that all operators in $\mathcal{O}$ are monotonic (like, for instance, $\mathcal{P}$, $\mathcal{B}$, and $\mathcal{U}$), *nomore++*'s hybrid lookahead has the same propagational strength as full hybrid lookahead.

Theorem 3. *Let Π 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

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

Fact *3.* states that *nomore++*'s lookahead is, in combination with propagation, as powerful as full hybrid lookahead. Facts *1.* and *2.* constitute that it is always at least as powerful as any kind of uniform lookahead. Thereby, 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 synchronising heads and bodies. In practise, lookahead is interleaved with $\mathcal{P}$ anyway, since it is integrated into propagation, viz. $(\mathcal{PBUL})^*$. More importantly, *nomore++*'s restricted hybrid lookahead, assigning $\ominus$ to heads and $\oplus$ to bodies only, faces approximately the same computational efforts as encountered in the uniform case and not more than the most consuming uniform lookahead, since $2 * min\{|head(\Pi)|, |body(\Pi)|\} \leq |head(\Pi)| + |body(\Pi)| \leq 2 * max\{|head(\Pi)|, |body(\Pi)|\}$.[8]

[8] For both, heads and bodies, we have $|head(\Pi)| \leq |\Pi|$ and $|body(\Pi)| \leq |\Pi|$, respectively. In uniform cases, factor 2 accounts for assigning *both* values, $\oplus$ and $\ominus$, one after the other.

$$\Pi_b^n = \left\{ \begin{array}{llll} r_0 : x \leftarrow not\ x & & & \\ r_1 : x \leftarrow a_1, b_1 & r_2 : a_1 \leftarrow not\ b_1 & r_3 : b_1 \leftarrow not\ a_1 & \\ \vdots & & & \\ r_{3n-2} : x \leftarrow a_n, b_n & r_{3n-1} : a_n \leftarrow not\ b_n & r_{3n} : b_n \leftarrow not\ a_n & \end{array} \right\}$$

$$\Pi_h^n = \left\{ \begin{array}{llll} r_0 : x \leftarrow c_1, \ldots, c_n, not\ x & & & \\ r_1 : c_1 \leftarrow a_1 & r_2 : c_1 \leftarrow b_1 & r_3 : a_1 \leftarrow not\ b_1 & r_4 : b_1 \leftarrow not\ a_1 \\ \vdots & & & \\ r_{4n-3} : c_n \leftarrow a_n & r_{4n-2} : c_n \leftarrow b_n & r_{4n-1} : a_n \leftarrow not\ b_n & r_{4n} : b_n \leftarrow not\ a_n \end{array} \right\}$$

Fig. 1. Lookahead programs Π_b^n and Π_h^n for some $n \geq 0$.

Finally, let us demonstrate that *nomore++*'s hybrid lookahead is in fact *strictly* more powerful than uniform ones. Consider Programs Π_b^n and Π_h^n, given in Figure 1. Both programs have, due to rule r_0 in the respective program, no answer sets and are thus unsatisfiable. For Program Π_b^n, this can be found out by assigning $\oplus$ to bodies of the form $\{a_i, b_i\}$ ($1 \leq i \leq n$) and by backward propagation via $\mathcal{B}$. With Program Π_h^n, assigning $\ominus$ to an atom c_i ($1 \leq i \leq n$) leads to a conflict by backward propagation via $\mathcal{B}$. Provided that $\mathcal{B}$ belongs to $\mathcal{O}$ in $\mathcal{L}^{\mathcal{O}}$,[9] body-based lookahead detects the unsatisfiability of Π_b^n, and head-based lookahead does the same for Π_h^n. Hence, *nomore++*'s hybrid lookahead detects the unsatisfiability of both programs without any choices being made. Unlike this, detecting the unsatisfiability of Π_b^n with head-based lookahead and choices restricted to heads (*smodels*' strategy) requires exponentially many choices in n. The same holds for Π_h^n with body-based lookahead and choices restricted to bodies (*noMoRe*'s strategy). Respective benchmark results are provided in Section 6.

5 Maintaining unfounded-freeness

A characteristic feature, distinguishing logic programming from propositional logic, is that true atoms must be derived via the rules of a logic program. For problems that involve reasoning, e.g. Hamiltonian cycles, this allows for more elegant and compact encodings in logic programming than in propositional logic. Such logic programming encodings produce *non-tight* programs [11, 12], for which there is a mismatch between answer sets and the models of programs' completions [8]. The mismatch is due to the potential of circular support among atoms. Such circularity is prohibited by the answer set semantics, but not by the semantics of propositional logic. The necessity of supporting true atoms non-circularly is reflected by propagation operator $\mathcal{U}$ in Section 3.

We detail in this section how our extended concept of an assignment, incorporating bodies in addition to heads, can be used for avoiding that atoms assigned $\oplus$ are subsequently detected to be unfounded. (Note that such a situation results in a conflict.) More formally, our goal is to avoid that atoms belonging to $A^{\oplus}$ in an assignment A

[9] If $\mathcal{B} \notin \mathcal{O}$, neither variant of lookahead detects unsatisfiability without making choices.

are contained in $U(B)$ for some extension B of A, i.e. $A \sqsubseteq B$. We therefore devise a computational strategy that is based on a modified choice operator, largely preventing conflicts due to true atoms becoming unfounded as a result of some later step. Finally, we point out how our computational strategy facilitates the implementation of operator $\mathcal{U}$ and which measures must be taken in the implementation of operators $\mathcal{B}$ and $\mathcal{L}$.

Let us first reconsider program Π_2 in (2) and (3) for illustrating the problem of true atoms participating in an unfounded set. Assume that the collection $(\mathcal{PBU})$ of operators is used for propagation and that we start with $A_0 = (\mathcal{PBU})^*((\emptyset,\emptyset)) = (\emptyset,\emptyset)$. Let our first choice be applying $\mathcal{C}^\oplus$ to atom e. We obtain

$$A_1 = (\mathcal{PBU})^*((\{e\},\emptyset)) = (\{e, f, \mathit{body}(r_7)\},\emptyset).$$

At this point, we cannot determine a rule for deriving the true atom e, since we have two possibilities, r_5 and r_6. Let us apply $\mathcal{C}^\oplus$ to atom d next. We obtain

$$A_2 = (\mathcal{PBU})^*(A_1 \sqcup (\{d\},\emptyset)) = A_1 \sqcup (\{d, \mathit{body}(r_4)\}, \{c, \mathit{body}(r_3)\}).$$

Still we do not know whether to use r_5 or r_6 for deriving e. Our next choice is applying $\mathcal{C}^\oplus$ to atom a, and propagation via $(\mathcal{PB})$ yields

$$\begin{aligned} A_2' &= (\mathcal{PB})^*(A_2 \sqcup (\{a\},\emptyset)) \\ &= A_2 \sqcup (\{a, \mathit{body}(r_1), \mathit{body}(r_6)\}, \{b, \mathit{body}(r_2), \mathit{body}(r_5)\}). \end{aligned}$$

We have $U(A_2') = \{b, c, e, f, \mathit{body}(r_6), \mathit{body}(r_7)\}$, and $\mathcal{U}(A_2')$ yields a conflict on atoms e and f and on bodies $\mathit{body}(r_6)$ and $\mathit{body}(r_7)$.

The reason for such a conflict is applying choice operator $\mathcal{C}^\oplus$ to a head or a body lacking an established non-circular support. Consider a head h that is in $A^\oplus$, but not in $T(A)$, that is, h has not been derived by a rule yet. Supposing that h is not unfounded with respect to A, i.e. $h \notin U(A)$, some of the bodies in $\mathit{body}(h)$ might still be assigned $\ominus$ in the ongoing computation. As a consequence, all bodies potentially providing a non-circular support for h might be contained in $B^\ominus$ for some extension B of A, that is, $A \sqsubseteq B$. For such an assignment B, we then have $h \in U(B)$, and propagation via $\mathcal{U}$ leads to a conflict. Similarly, a body b that is in $A^\oplus$ but not supported with respect to A, i.e. $b \notin S(A)$, can be unfounded in an assignment B such that $A \sqsubseteq B$, as some positive literal in b^+ might be contained in $U(B)$.

Conflicts due to $\oplus$-assigned heads and bodies becoming unfounded cannot occur when non-circular support is already established. That is, every head in $A^\oplus$ must be derived by a body that is in $A^\oplus$, too. Similarly, the positive part b^+ of a body b in $A^\oplus$ must be derived by other bodies in $A^\oplus$. This leads us to the following definition.

Definition 9. *Let Π be a logic program and let A be an assignment of $\mathit{head}(\Pi) \cup \mathit{body}(\Pi)$. We define A as unfounded-free, if*

$$(\mathit{head}(\Pi) \cap A^\oplus) \cup (\textstyle\bigcup_{b \in \mathit{body}(\Pi) \cap A^\oplus} b^+) \subseteq \mathit{Cn}(\{r \in \Pi \mid \mathit{body}(r) \in A^\oplus\}^\emptyset) \,.$$

Heads and bodies in the positive part, $A^\oplus$, of an unfounded-free assignment A cannot be unfounded with respect to any extension of A.

Theorem 4. *Let Π be a logic program and let A be an unfounded-free assignment of $head(\Pi) \cup body(\Pi)$. Then, $A^{\oplus} \cap U(B) = \emptyset$ for any assignment B such that $A \sqsubseteq B$.*

Unfounded-freeness is maintained by forward propagation operator $\mathcal{P}$. That is, when applied to an unfounded-free assignment, operator $\mathcal{P}$ produces again an unfounded-free assignment.

Theorem 5. *Let Π be a logic program and let A be an unfounded-free assignment of $head(\Pi) \cup body(\Pi)$. If $\mathcal{P}(A)$ is defined, then $\mathcal{P}(A)$ is unfounded-free.*

For illustrating the above result, reconsider Π_2 in (2) and (3) and assignment $A = (\{body(r_5)\}, \emptyset)$. A is unfounded-free because $body(r_5)^+ = \emptyset \subseteq Cn(\{e \leftarrow\}) = \{e\}$. We obtain $\mathcal{P}^*(A) = (\{body(r_5), e, body(r_7), f\}, \emptyset)$, which is again unfounded-free, since $\{e, f\} \cup body(r_5)^+ \cup body(r_7)^+ = \{e, f\} \subseteq Cn(\{e \leftarrow, f \leftarrow e\}) = \{e, f\}$.

In order to guarantee unfounded-freeness for choice operator $\mathcal{C}^{\oplus}$, the set X of heads and bodies to choose from has to be restricted appropriately. To this end, *nomore++* provides the following instance of $\mathcal{C}$.

Definition 10. *Let Π be a logic program and let A be a partial assignment of $head(\Pi) \cup body(\Pi)$. We define*

1. $\mathcal{D}^{\oplus}_{\Pi}(A) = \mathcal{C}^{\oplus}_{\Pi}(A, (body(\Pi) \cap S(A)))$;
2. $\mathcal{D}^{\ominus}_{\Pi}(A) = \mathcal{C}^{\ominus}_{\Pi}(A, (body(\Pi) \cap S(A)))$.

Operator $\mathcal{D}$ differs from $\mathcal{C}$ in restricting its choices to supported bodies. This still guarantees completeness, as an assignment A that is closed under $(\mathcal{PBU})$ is total if $(body(\Pi) \cap S(A)) \setminus (A^{\oplus} \cup A^{\ominus}) = \emptyset$.[10]

Like $\mathcal{P}$, operator $\mathcal{D}$ maintains unfounded-freeness.

Theorem 6. *Let Π be a logic program and let A be an unfounded-free partial assignment of $head(\Pi) \cup body(\Pi)$ such that $(body(\Pi) \cap S(A)) \setminus (A^{\oplus} \cup A^{\ominus}) \neq \emptyset$. Then, $\mathcal{D}^{\circ}(A)$ is unfounded-free for $\circ \in \{\oplus, \ominus\}$.*

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

Unlike $\mathcal{P}$ and $\mathcal{D}$, backward propagation $\mathcal{B}$ and lookahead $\mathcal{L}$ can generally not maintain unfounded-freeness, as they assign $\oplus$ for other reasons than support. That is why we introduce at the implementation level a *weak* counterpart of $\oplus$, denoted by $\otimes$.[11] Value $\otimes$ indicates that some head or body, for which non-circular support is not yet established, must eventually be assigned $\oplus$. In the implementation, only $\mathcal{P}$ and $\mathcal{D}$ assign

[10] Note that any body whose literals are true is in $A^{\oplus}$ due to $\mathcal{P}$. All other bodies either contain a false literal and are in $A^{\ominus}$ due to $\mathcal{P}$, or they positively depend on unfounded atoms in $U(A)$ and are in $A^{\ominus}$ due to $\mathcal{U}$.

[11] Similar to *dlv*'s *must-be-true* [13]; see Section 7.

$\oplus$, while operators $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{L}$ can only assign $\otimes$ (or $\ominus$).[12] Any head or body in $A^{\otimes}$ can be turned into $\oplus$ by $\mathcal{P}$ without causing a conflict. So, by distinguishing two types of "true", we guarantee unfounded-freeness for the $\oplus$-assigned part of an assignment.

Maintaining unfounded-freeness allows for a lazy implementation of operator $\mathcal{U}$. That is, the scope of $U(A)$ (cf. Definition 5) can be restricted to $(head(\Pi) \cup body(\Pi)) \setminus (A^{\oplus} \cup A^{\ominus})$, taking the non-circular support of $A^{\oplus}$ for granted. In other words, the computation of $U(A)$ is restricted to heads and bodies being either unassigned or assigned $\otimes$. Beside the fact that $\mathcal{D}$ can assign $\oplus$ and $\mathcal{C}$ only $\otimes$, using $\mathcal{D}$ instead of $\mathcal{C}$ helps in avoiding that true atoms lead to a conflict by participating in an unfounded set. This can be crucial for efficiently computing answer sets of non-tight programs, as the benchmark results in the next section demonstrate.

6 System design and experimental results

nomore++ is implemented in C++ and uses *lparse* as parser. A salient feature of *nomore++* is that it facilitates the use of different sets of operators. For instance, if called with command line option "`--choice-op C --lookahead-op PB`", it uses operator $\mathcal{C}$ for choices and $(\mathcal{PB})$ for propagation within lookahead. The default strategy of *nomore++* is applying $(\mathcal{PBUL})$ for propagation, where lookahead by $\mathcal{L}$ works as detailed in Section 4, and $\mathcal{D}$ as choice operator. By default, $(\mathcal{PBU})$ is used for propagation within lookahead. The system is freely available at [14].

Due to space limitations, we confine our listed experiments to selected benchmarks illustrating the major features of *nomore++*. A complete evaluation, including further ASP solvers, e.g. *assat* and *cmodels*, can be found at the ASP benchmarking site [15]. All tests were run on an AMD Athlon 1.4GHz PC with 512MB RAM. As in the context of [15], a memory limit of 256MB as well as a time limit of 900s have been 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. [15]).[13] 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 and *dlv*. For a complement, we also give tests for *nomore++* with body-based lookahead $\mathcal{L}^{(\mathcal{PBU})}(A, body(\Pi))$ for an assignment A and a program Π, abridged $\mathcal{L}_b$. The tests with *nomore++*'s hybrid lookahead rely on $\mathcal{L}^{\oplus,(\mathcal{PBU})}(A, body(\Pi)) \sqcup \mathcal{L}^{\ominus,(\mathcal{PBU})}(A, head(\Pi))$, abbreviated by $\mathcal{L}_{bh}$.

For illustrating the effect of maintaining unfounded-freeness, Table 1 shows results obtained on Hamiltonian cycle problems on complete graphs with n nodes (HC_n), both

[12] 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 $A^{\otimes}$.

[13] Other apt benchmarks are "Factoring" and "Schur Numbers" (cf. [15]); in both cases, *smodels* still outperforms *nomore++* by an order of magnitude.

HC_n	*dlv*	*smodels*	*nomore++* $(PBVL_b)^*D$	*nomore++* $(PBVL_{bh})^*D$	*dlv*	*smodels*	*nomore++* $(PBVL_b)^*D$	*nomore++* $(PBVL_{bh})^*D$
3	(0.00)	1 (0.00)	1 (0.00)	1 (0.00)	(0.00)	1 (0.00)	1 (0.00)	1 (0.00)
4	(0.00)	2 (0.01)	2 (0.01)	2 (0.00)	(0.00)	5 (0.00)	5 (0.00)	5 (0.00)
5	(0.00)	3 (0.00)	3 (0.00)	3 (0.01)	(0.01)	26 (0.00)	23 (0.02)	23 (0.02)
6	(0.01)	4 (0.01)	4 (0.01)	4 (0.01)	(0.02)	305 (0.02)	119 (0.11)	119 (0.11)
7	(0.01)	3(0.01)	5 (0.02)	5 (0.02)	(0.14)	4,814 (0.38)	719 (0.83)	719 (0.85)
8	(0.01)	8 (0.00)	6 (0.03)	6 (0.03)	(1.06)	86,364 (7.29)	5,039 (7.40)	5,039 (7.60)
9	(0.02)	48 (0.01)	7 (0.05)	7 (0.05)	(10.02)	1,864,470(177.91)	40,319 (73.94)	40,319 (76.09)
10	(0.03)	1,107 (0.18)	8 (0.08)	8 (0.08)	(109.21)	n/a	362,879 (818.73)	362,879 (842.57)
11	(0.03)	18,118 (2.88)	9 (0.13)	9 (0.12)	n/a	n/a	n/a	n/a
12	(0.05)	398,306 (65.29)	10 (0.19)	10 (0.20)	n/a	n/a	n/a	n/a
13	(0.06)	n/a	11 (0.29)	11 (0.30)	n/a	n/a	n/a	n/a

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

Π_b^n	*dlv*	*smodels*	*nomore++* $(PBVL_b)^*D$	*nomore++* $(PBVL_{bh})^*D$	Π_h^n	*dlv*	*smodels*	*nomore++* $(PBVL_b)^*D$	*nomore++* $(PBVL_{bh})^*D$
0	(0.04)	0 (0.00)	0 (0.01)	0 (0.01)	0	(0.07)	0 (0.01)	0 (0.01)	0 (0.01)
2	(0.04)	0 (0.00)	0 (0.01)	0 (0.01)	2	(0.04)	0 (0.01)	0 (0.01)	0 (0.01)
4	(0.04)	3 (0.00)	0 (0.01)	0 (0.01)	4	(0.04)	0 (0.01)	3 (0.01)	0 (0.01)
6	(0.04)	15 (0.00)	0 (0.01)	0 (0.01)	6	(0.04)	0 (0.01)	15 (0.01)	0 (0.01)
8	(0.05)	63 (0.00)	0 (0.01)	0 (0.01)	8	(0.05)	0 (0.01)	63 (0.01)	0 (0.01)
10	(0.06)	255 (0.00)	0 (0.01)	0 (0.01)	10	(0.06)	0 (0.01)	255 (0.03)	0 (0.01)
12	(0.10)	1,023 (0.01)	0 (0.01)	0 (0.01)	12	(0.10)	0 (0.01)	1,023 (0.09)	0 (0.02)
14	(0.26)	4,095 (0.03)	0 (0.02)	0 (0.02)	14	(0.29)	0 (0.01)	4,095 (0.33)	0 (0.02)
16	(0.93)	16,383 (0.11)	0 (0.02)	0 (0.02)	16	(1.06)	0 (0.01)	16,383 (1.27)	0 (0.02)
18	(3.60)	65,535 (0.43)	0 (0.03)	0 (0.02)	18	(4.14)	0 (0.01)	65,535 (5.04)	0 (0.02)
20	(14.46)	262,143 (1.71)	0 (0.03)	0 (0.03)	20	(16.61)	0 (0.01)	262,143 (20.37)	0 (0.02)
22	(57.91)	1,048,575 (6.92)	0 (0.03)	0 (0.03)	22	(66.80)	0 (0.01)	1,048,575 (81.24)	0 (0.03)
24	(233.44)	4,194,303 (27.70)	0 (0.03)	0 (0.03)	24	(270.43)	0 (0.01)	4,194,303 (322.73)	0 (0.03)
26	n/a	16,777,215 (111.42)	0 (0.03)	0 (0.03)	26	n/a	0 (0.01)	n/a	0 (0.04)
28	n/a	67,108,863 (449.44)	0 (0.04)	0 (0.04)	28	n/a	0 (0.01)	n/a	0 (0.04)
30	n/a	n/a	0 (0.04)	0 (0.04)	30	n/a	0 (0.01)	n/a	0 (0.04)

Table 2. Results for (a) Π_b^n; (b) Π_h^n

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* needs an exponential number of choices, even for finding the first answer set. The usage of choice operator $\mathcal{D}$ enforces that rules are chained in the appropriate way for solving HC_n programs. We note that, on HC_n programs, *dlv* performs even better regarding time (cf. [15]); the different concept of "choice points" makes *nomore++* and *dlv* incomparable in this respect.

The results in Table 2, obtained on programs Π_b^n and Π_h^n from Figure 1, aim at supporting *nomore++*'s hybrid lookahead. 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 programs Π_h^n, yet a bad one on Π_b^n. The converse is true when restricting *nomore++* to lookahead on bodies only (command line option "`--body-lookahead`"). *nomore++* with hybrid lookahead performs optimal regarding choice points on both types of programs. Also, a comparison of the two *nomore++* variants shows that hybrid lookahead does not introduce a computational overhead. Note that *dlv* performs similar to the worst approach on both Π_b^n and Π_h^n.

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 assignment, including bodies in addition to atoms, (ii) the more powerful lookahead operation, and (iii) the computational strategy of maintaining unfounded-freeness. We draw from previous work on the *noMoRe* system [5], whose approach to answer set computation is based on "colouring" the rule dependency graph (RDG) of a program. *noMoRe* pursues a rule-based approach, which amounts to restricting the domain of assignments to $body(\Pi)$. The functionality of *noMoRe* has been described in [9] by graph-theoretical operators similar to $\mathcal{P}$, $\mathcal{U}$, $\mathcal{C}$, and $\mathcal{D}$. *nomore++*'s operators for backward propagation ($\mathcal{B}$) and lookahead ($\mathcal{L}$) were presented here for the first time.[14] In general, operator-based specifications facilitate formal comparisons between techniques used by different ASP solvers. Operators capturing propagation in *dlv* are given in [18]. Pruning operators based on Fitting's [7] and well-founded semantics [6] are investigated in [19]. The full paper contains a detailed comparison of these operators.

smodels [1] and *dlv* [2] pursue a literal-based approach, which boils down to restricting the domain of assignments to $head(\Pi)$. However, in both systems, propagation keeps track of the state of rules, which bears more redundancy than using bodies.[15] *nomore++*'s strategy of maintaining unfounded-freeness is closely related to some concepts used in *dlv*, but still different. In fact, the term "unfounded-free" is borrowed from [20], where it is used for assessing the complexity of unfounded set checks and characterising answer sets in the context of disjunctive logic programs. We, however, address assignments in which the non-circular support of true atoms is guaranteed. Also, *dlv* selects its choices among so-called *possibly-true literals* [13], corresponding to a literal-based version of choice operator $\mathcal{D}$. But, as discussed in Section 5, unfounded-freeness in our context cannot be achieved by choosing atoms to be true.

We conclude with outlining some subjects to future development and research. First, the low-level implementation of *nomore++* will be improved further in order to be closer to more mature ASP solvers, such as *smodels* and *dlv*. Second, aggregates, like *smodels*' cardinality and weight constraints, will be supported in future versions of *nomore++*, in order to enable more compact problem encodings. Finally, we detail in the full paper that restricting choices to either heads or bodies leads to exponentially worse proof complexity. Although choice operator $\mathcal{D}$ is valuable for handling non-tight programs, it is directly affected, as it restricts choices to bodies.[16] Thus, conditions for allowing non-supported choices, though still preferring supported choices, will be explored, which might lead to new powerful heuristics for answer set solving.

Acknowledgements. We are grateful to Yuliya Lierler, Tomi Janhunen, and anonymous referees for their helpful comments. This work was supported by DFG under grant SCHA 550/6-4 as well as the EC through IST-2001-37004 WASP project.

[14] Short or preliminary, respectively, notes on *nomore++* can be found in [16, 17].

[15] The number of unique bodies in a program is always less than or equal to the number of rules.

[16] Note that literal-based solvers, such as *smodels* and *dlv*, suffer from exponential worst-case complexity as well.

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). Springer (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 (LPNMR'01), Springer (2001) 406–410
6. van Gelder, A., Ross, K., Schlipf, J.: The well-founded semantics for general logic programs. Journal of the ACM **38** (1991) 620–650
7. Fitting, M.: Fixpoint semantics for logic programming: A survey. Theoretical Computer Science **278** (2002) 25–51
8. Clark, K.: Negation as failure. In Gallaire, H., Minker, J., eds.: Logic and Data Bases. Plenum Press (1978) 293–322
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). CEUR (2003) 137–150
10. Cook, S., Reckhow, R.: The relative efficiency of propositional proof systems. Journal of Symbolic Logic **44** (1979) 36–50
11. Fages, F.: Consistency of clark's completion and the existence of stable models. Journal of Methods of Logic in Computer Science **1** (1994) 51–60
12. Erdem, E., Lifschitz, V.: Tight logic programs. Theory and Practice of Logic Programming **3** (2003) 499–518
13. Faber, W., Leone, N., Pfeifer, G.: Pushing goal derivation in DLP computations. In Gelfond, M., Leone, N., Pfeifer, G., eds.: Proceedings of the Fifth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'99). Springer (1999) 177–191
14. (`http://www.cs.uni-potsdam.de/nomore`)
15. (`http://asparagus.cs.uni-potsdam.de`)
16. Anger, C., Gebser, M., Linke, T., Neumann, A., Schaub, T.: The `nomore++` system. In Baral, C., Greco, G., Leone, N., Terracina, G., eds.: Proceedings of the Eighth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'05). Springer (2005) 422–426
17. Anger, C., Gebser, M., Linke, T., Neumann, A., Schaub, T.: The `nomore++` approach to answer set solving. In Vos, M.D., Provetti, A., eds.: Proceedings of the Third International Workshop on Answer Set Programming (ASP'05). CEUR (2005) 163–177
18. Faber., W.: Enhancing Efficiency and Expressiveness in Answer Set Programming Systems. Dissertation, Technische Universität Wien (2002)
19. Calimeri, F., Faber, W., Leone, N., Pfeifer, G.: Pruning operators for answer set programming systems. In Benferhat, S., Giunchiglia, E., eds.: Proceedings of the Nineth International Workshop on Non-Monotonic Reasoning (NMR'02). (2002) 200–209
20. Leone, N., Rullo, P., Scarcello, F.: Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation. Information and Computation **135** (1997) 69–112