The *nomore++* System

Christian Anger, Martin Gebser, Thomas Linke, André Neumann, Torsten Schaub[⋆]

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

Abstract. We present a new answer set solver *nomore++*. Distinguishing features include its treatment of heads and bodies equitably as computational objects and a new hybrid lookahead. *nomore++* is close to being competitive with state-of-the-art answer set solvers, as demonstrated by selected experimental results.

1 Introduction

A large part of the success of Answer Set Programming (ASP) is owed to the easy availability of efficient solvers. We present a new ASP solver, called *nomore++* that pursues a hybrid approach in combining features from literal-based approaches, like *smodels* [1] and *dlv* [2], with the rule-based approach of its predecessor *noMoRe* [3]. 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. In particular, we demonstrate that the resulting hybrid lookahead operation allows for propagating more than previous approaches. Also, we detail a special strategy, keeping assignments unfounded-free and empirically show that it outperforms *smodels* on relevant benchmarks. Another feature of *nomore++* is its configurable operator-based design. The system is available at [4].

2 Theoretical Background

The *nomore++* system deals with normal logic programs under the *answer set semantics* [5]. A *normal logic program* is a finite set of rules of the form $p_0 \leftarrow p_1, \dots, p_m, not\ p_{m+1}, \dots, not\ p_n$, where $n \geq m \geq 0$, and each p_i $(0 \leq i \leq n)$ is an *atom*. A *literal* is an atom p or its (default) negation $not\ p$. For such a rule r, let $head(r) = p_0$ be the *head* of r and $body(r) = \{p_1, \dots, p_m, not\ p_{m+1}, \dots, not\ p_n\}$ be the *body* of r. For a program Π, we write $head(\Pi) = \{head(r) \mid r \in \Pi\}$ and $body(\Pi) = \{body(r) \mid r \in \Pi\}$. Without loss of generality, we assume that each atom of a program is the head of at least one rule in the program.[1]

Given a normal logic program Π, *nomore++* computes the answer sets of Π. Unlike other solvers, such as *smodels* [1] and *dlv* [2],[2] the *nomore++* approach is based

[⋆] Affiliated with the School of Computing Science at Simon Fraser University, Burnaby, Canada.
[1] Atoms not occurring as heads are necessarily false. *nomore++* removes such atoms during preprocessing.
[2] Note that *dlv* is designed to handle disjunctive logic programs, which are on a higher complexity level than normal ones.

on an extended concept of assignments. That is, *nomore++* maps heads *and* bodies of a program into $\{\oplus, \ominus\}$, indicating whether a head or body is true or false, respectively. Given program Π, we define a (partial) assignment as a partial mapping $A : \mathit{head}(\Pi) \cup \mathit{body}(\Pi) \rightarrow \{\oplus, \ominus\}$. For simplicity, we often represent such assignments as pairs $(A^{\oplus}, A^{\ominus})$, where $A^{\oplus} = \{x \mid A(x) = \oplus\}$ and $A^{\ominus} = \{x \mid A(x) = \ominus\}$. Treating heads and bodies equitably as computational objects provides great flexibility. Bodies can be viewed as conjunctions, and their explicit representation allows for reasoning about applicability of rules, in addition to atoms' truth values. Structurally more complex objects have recently been deployed in SAT [6] and neighboring fields [7, 8]. However, to the best of our knowledge this has not been done in ASP, so far.

nomore++ is a highly flexible, runtime-configurable system. Flexibility roots on an operator-based design, featuring (among others) the following basic operators: i) Forward propagation operator $\mathcal{P}$, ii) Backward propagation operator $\mathcal{B}$, iii) *Unfounded set* [9] operator $\mathcal{U}$, and iv) Choice operator $\mathcal{C}$.[3] Omitting details, $\mathcal{P}$ generalizes *Fitting's operator* [11] to bodies, combined operators $(\mathcal{PU})$ coincide with the *well-founded operators* [9], and $(\mathcal{PB})$ and $\mathcal{U}$ correspond to *smodels*' functions *atleast* and *atmost* [1]. Differences to atom- and rule-based approaches come up at *nomore++*'s more general choice operator $\mathcal{C}$. For instance, $\mathcal{C}$ can assign $\oplus$ to a body, enabling propagation to decide all the body's literals and heads. Such complex choices are not possible with assignments restricted to atoms.

Following [12], we characterize the process of answer set formation by a sequence of assignments. Based on the above operators, one possible strategy is to determine deterministic consequences with propagation operators $\mathcal{P}$, $\mathcal{B}$, and $\mathcal{U}$ and to apply choice operator $\mathcal{C}$ whenever a fix-point of propagation is not total. We abbreviate this strategy by $(\mathcal{PBU})^*\mathcal{C}$, where $(\mathcal{PBU})^*$ denotes the closure under operators $\mathcal{P}$, $\mathcal{B}$, and $\mathcal{U}$. The $\oplus$-assigned atoms in a total assignment, constructed by $(\mathcal{PBU})^*\mathcal{C}$, form an answer set.[4] However, other strategies for answer set formation can also be shown to be sound and complete. For instance, we can safely skip either backward or forward propagation, yielding strategies $(\mathcal{PU})^*\mathcal{C}$ and $(\mathcal{BU})^*\mathcal{C}$. Due to its configurable design, *nomore++* can handle those different strategies as well.

An important concept, distinguishing answer set programming from propositional logic, is well-foundedness. It is thus desirable that no $\oplus$-assigned atom becomes unfounded later on. In fact, no atom in the positive part $A^{\oplus}$ of an assignment can become unfounded, if each atom in $A^{\oplus}$ is the head of a rule whose body is non-circularly justified in $A^{\oplus}$. We call such an assignment *unfounded-free*,[5] and *nomore++* supports the computation of unfounded-free assignments by providing choice operator $\mathcal{D}$ as an alternative to $\mathcal{C}$. As opposed to $\mathcal{C}$, $\mathcal{D}$ is restricted to bodies whose positive preconditions are already present in $A^{\oplus}$ and where only negative literals are undecided. On the one hand, $\mathcal{D}$ restricts possible choices. But on the other hand, assignments are kept unfounded-free, which pays off on non-tight problems (cf. Section 4).

[3] For a detailed operational characterization of *nomore++*, please refer to [10].

[4] Note that we derive a contradiction when different values are to be assigned to some head or body. In this case, a total assignment cannot be constructed.

[5] A related notion for disjunctive programs is described in [13].

In addition to propagation operators $\mathcal{P}$, $\mathcal{B}$, and $\mathcal{U}$, lookahead strengthens propagation by conflict-driven assertions (*nomore*++ provides operator $\mathcal{L}$ for this [10]). Answer set solvers like *smodels* and *dlv* apply lookahead to atoms only. In contrast to them, *nomore*++ provides a *hybrid* lookahead considering both heads and bodies. In order to limit efforts to approximately the same amount as with lookahead on atoms, *nomore*++'s hybrid lookahead assigns $\oplus$ to bodies and $\ominus$ to atoms only. As shown in [10], this restriction does not decrease strength of propagation. Rather we demonstrate in Section 4 that hybrid lookahead can save exponentially many choices in comparison to lookahead applied to either atoms or bodies only.

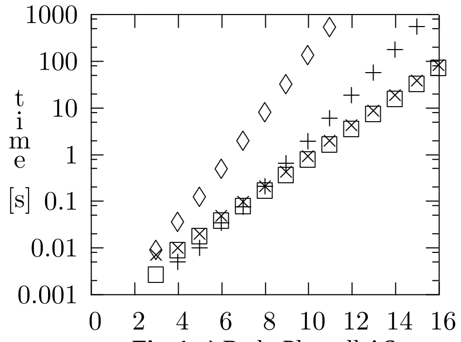

Fig. 1-a) Body-Plus, all AS

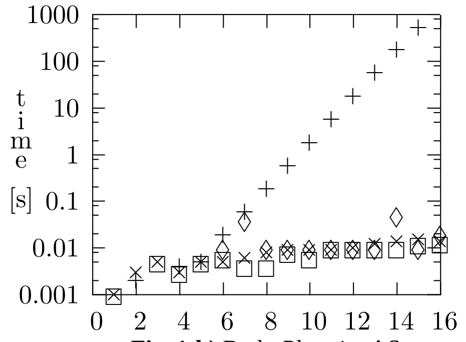

Fig. 1-b) Body-Plus, 1st AS

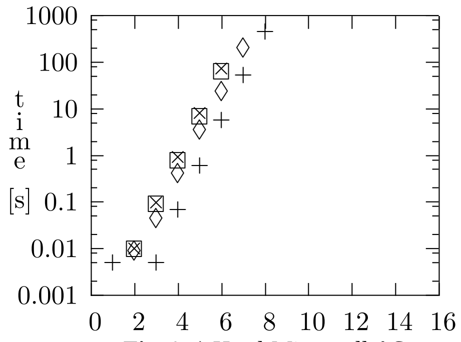

Fig. 1-c) Head-Minus, all AS

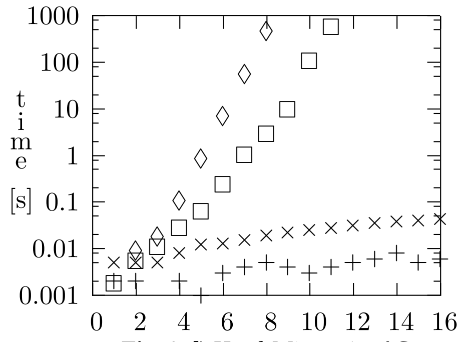

Fig. 1-d) Head-Minus, 1st AS

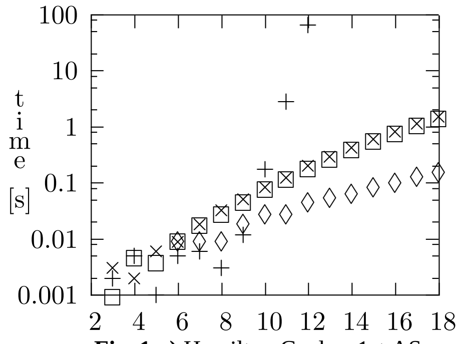

Fig. 1-e) Hamilton Cycles, 1st AS

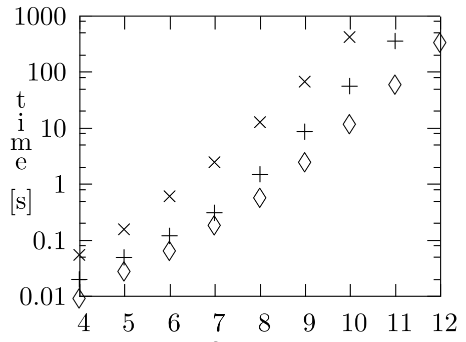

Fig. 1-f) EqTest, 1st AS

dlv ◊ *smodels* + *nomore*++(Body LaH) □ *nomore*++(Hybrid LaH) ×

3 System

The input language of *nomore++* is generated by the grounder lparse [14] from normal logic programs obeying the format “Logic Programs V1.0” as defined in [15].[6] A major feature of *nomore++* is that operators can be selected at runtime, enabling the use of a multitude of strategies (combinations of operators). Via command line option `-op`, the propagation and choice operators to be used can be determined. As lookahead allows for different degrees of propagation within, one can also determine which set of operators to use during lookahead via command line option `-laop`.

nomore++’s default strategy applies operators $\mathcal{P}$, $\mathcal{B}$, and $\mathcal{U}$ in usual propagation as well as in lookahead. Furthermore, $\mathcal{D}$ is the default choice operator. Note that operators $\mathcal{P}$, $\mathcal{U}$, and $\mathcal{D}$ keep a given assignment unfounded-free, which is not guaranteed for $\mathcal{B}$ and lookahead. At the implementation level, *nomore++* uses the additional truth value $\otimes$ for distinguishing between the unfounded-free part of an assignment and the part that must eventually be true but is not non-circularly justified, yet. The virtue of this is that the scope of unfounded set operator $\mathcal{U}$ can be restricted to $\otimes$-assigned and unassigned heads and bodies. In fact, $\mathcal{U}$ is implemented in a “lazy fashion” ignoring the $\oplus$-part of an assignment. The *dlv* system uses a similar feature, the truth value “must be true” [16].

Internally, *nomore++*’s primary data structure consists of a body-head dependency graph [17]. This is a very efficient structure, as it only stores each head-atom and each distinct body of a program, instead of each rule as most other ASP-solvers do. E.g., measuring over 241 randomly chosen ground programs in [15], the ratio of the number of distinct bodies over the number of rules is 0.41.

4 Selected experimental results

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, like *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. A memory limit of 256MB as well as a time limit of 900s were enforced. All results given in Figure 1 reflect the average of 10 runs.

Benchmarks 1-a to 1-d are taken from [4] and demonstrate the advantage of the hybrid lookahead strategy. For comparisons, we have in *nomore++* implemented body-based lookahead (“Body LaH”) in addition to hybrid lookahead (“Hybrid LaH”). Values on the x-axis are a measurement for the size of the problem, please check [4] for details. Examples denoted with “Body-Plus” (Figures 1-a and 1-b) are better suited for a body-based lookahead. The *nomore++* version with body-based lookahead outperforms *smodels* on these. Examples “Head-Minus” (Figures 1-c and 1-d) can be solved optimally with a head-based lookahead. Consequently, we have *smodels* outperforming *nomore++* with body-based lookahead. Please note, that *nomore++* with hybrid lookahead always performs similar to the better suited approach.

Benchmark 1-e demonstrates the advantage of *nomore++*’s strategy of keeping assignments unfounded-free. The figure reflects results obtained on classical Hamiltonian

[6] *nomore++* currently does not support *smodels*-style cardinality and weight constraints.

cycle problems on complete graphs, where values on the x-axis reflect the number of nodes in the graph.

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*. To reflect on this, we have in Figure 1-f included results from the "Equality Testing" benchmark taken from [15]. Please observe that, while *nomore++* performs worse than either *smodels* or *dlv*, it scales like the other two systems, indicating that only improvements with the implementation are needed.

Acknowledgments This work was supported by DFG under grant SCHA 550/6-4 as well as the EC through IST-2001-37004 WASP project.

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. 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 Int'l Conference on Logic Programming and Nonmonotonic Reasoning, Springer (2001) 406–410
4. (`http://www.cs.uni-potsdam.de/nomore`)
5. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of the International Conference on Logic Programming, MIT Press (1988) 1070–1080
6. Järvisalo, M., Junttila, T., Niemelä, I.: Unrestricted vs restricted cut in a tableau method for Boolean circuits. Annals of Mathematics and Artificial Intelligence (to appear)
7. Baaz, M., Egly, U., Leitsch, A.: Normal form transformations. [8] chapter 5 273–333
8. Robinson, J., Voronkov, A., eds.: Handbook of Automated Reasoning. MIT Press (2001)
9. van Gelder, A., Ross, K., Schlipf, J.: The well-founded semantics for general logic programs. Journal of the ACM **38** (1991) 620–650
10. Anger, C., Gebser, M., Linke, T., Neumann, A., Schaub, T.: The nomore++ approach to answer set solving. Submitted for publication (2005)
11. Fitting, M.: Fixpoint semantics for logic programming: A survey. Theoretical Computer Science **278** (2002) 25–51
12. Konczak, K., Linke, T., Schaub, T.: Graphs and colorings for answer set programming. Theory and Practice of Logic Programming (2005) To appear.
13. Leone, N., Rullo, P., Scarcello, F.: Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation. Information and Computation **135** (1997) 69–112
14. (`http://www.tcs.hut.fi/Software/smodels/`)
15. (`http://asparagus.cs.uni-potsdam.de`)
16. Faber, W., Leone, N., Pfeifer, G.: Pushing goal derivation in dlp computations. In: Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'99). (1999) 177–191
17. Linke, T., Sarsakov, V.: Suitable graphs for answer set programming. In Baader, F., Voronkov, A., eds.: Proceedings of the International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'05). Springer (2005) 154–168