What's a head without a body?

Christian Anger[1] and **Martin Gebser**[1] and **Tomi Janhunen**[2] and **Torsten Schaub**[1,3]

Abstract. Concepts in Answer Set Programming (ASP) are normally defined in terms of atoms. We show that the treatment of atoms and bodies (of rules) as equitable computational objects may yield exponential speed-ups, even for standard ASP-solvers such as *smodels*. To this end, we give simple transformations providing solvers with access to both kinds of objects and show that some families of examples can be solved exponentially faster after they have been transformed. We prove that these transformations may yield exponentially smaller search spaces.

1 SIMULATING HEADS AND BODIES

For a detailed introduction to Answer Set Programming (ASP), we refer the reader to the literature (cf. [3]) and confine ourselves to formalities essential to our contribution: Given an alphabet $\mathcal{P}$, a *logic program* is a finite set of rules, r, of the form $p_0 \leftarrow p_1, \ldots, p_m, not\ p_{m+1}, \ldots, not\ p_n$ where $n \geq m \geq 0$ and $p_i \in \mathcal{P}$ is an *atom* for $0 \leq i \leq n$. 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. The set of atoms occurring in a logic program Π is given by $atom(\Pi)$. The set of bodies in Π is $body(\Pi) = \{body(r) \mid r \in \Pi\}$. An *answer set* of a logic program Π is a model of Π that satisfies a certain "stability criterion" (cf. [3]).

We consider the following systems: (i) *smodels* [7], as a an atom-based ASP-solver, assigning truth values to atoms only; (ii) *dlv* [5], also atom-based, yet designed for handling disjunctive logic programs; (iii) *nomore++* [1], pursuing a hybrid approach, wherein truth values are assigned to both atoms and bodies; (iv) a restriction of *nomore++* to assignments to bodies only, indicated by *nomore++*^{B}. [4] All systems incorporate techniques such as propagation and lookahead in their solving procedures [1, 7].

Inspired by structural normal form translations [2], we give two simple transformations introducing new symbols for bodies and atoms, respectively. This allows standard ASP-solvers to access the underlying structures and implicitly incorporate them in their solving procedures. To this end, we extend the alphabet $\mathcal{P}$ of a program Π by adding a new atom α_a for each $a \in atom(\Pi)$ and a new atom β_B for each $B \in body(\Pi)$. We then define a transformation over the extended alphabet:

$$\mathcal{T}_B(\Pi) = \left\{ \begin{array}{rcl} \beta_B & \leftarrow & B \\ head(r) & \leftarrow & \beta_B \end{array} \;\middle|\; r \in \Pi, B = body(r) \right\}$$

This enables ASP-solvers branching on atoms, like *smodels*, to indirectly incorporate bodies as computable objects in their solving procedures. Vice versa, a different transformation over the extended alphabet enables body-based systems (such as *nomore++*^{B}) to indirectly incorporate atoms:

$$\mathcal{T}_a(\Pi) = \left\{ \begin{array}{rcl} \alpha_a & \leftarrow & body(r) \\ a & \leftarrow & \alpha_a \end{array} \;\middle|\; r \in \Pi, a = head(r) \right\}$$

As an example, consider program Π along with its transforms:

$$\Pi = \left\{ \begin{array}{rcl} a & \leftarrow & b, not\ c \\ a & \leftarrow & d, not\ e \end{array} \right\}$$

$$\mathcal{T}_B(\Pi) = \left\{ \begin{array}{rcl} \beta_{\{b,not\ c\}} & \leftarrow & b, not\ c \\ a & \leftarrow & \beta_{\{b,not\ c\}} \\ \beta_{\{d,not\ e\}} & \leftarrow & d, not\ e \\ a & \leftarrow & \beta_{\{d,not\ e\}} \end{array} \right\}$$

$$\mathcal{T}_a(\Pi) = \left\{ \begin{array}{rcl} \alpha_a & \leftarrow & b, not\ c \\ \alpha_a & \leftarrow & d, not\ e \\ a & \leftarrow & \alpha_a \end{array} \right\}$$

To begin with, we empirically underpin our claim that treating atoms and bodies as equitable computational objects may yield exponential speed-ups by some experimental results. As common in proof complexity [4], we need an infinite family of logic programs witnessing an exponential behavior in the best-case. For this, we consider the following programs Π_B^n and Π_a^n for $n \geq 0$:

$$\Pi_B^n = \left\{ \begin{array}{ll} r_0 : x \leftarrow not\ x & \\ r_1 : x \leftarrow not\ a_1, not\ b_1 & \\ \vdots & \\ r_n : x \leftarrow not\ a_n, not\ b_n & \\ r_{n+1} : a_1 \leftarrow not\ b_1 & r_{n+2} : b_1 \leftarrow not\ a_1 \\ \vdots & \vdots \\ r_{3n-1} : a_n \leftarrow not\ b_n & r_{3n} : b_n \leftarrow not\ a_n \end{array} \right\}$$

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

None of these programs has an answer set, that is, all of them are *unsatisfiable*. Importantly, this can be determined *linearly*. In both programs, the source of unsatisfiability lies in r_0. For Programs Π_B^n, this can be found out linearly by assigning "true" to bodies of the form $\{not\ a_i, not\ b_i\}$ $(1 \leq i \leq n)$ and by subsequent propagation. For *smodels*, transformation $\mathcal{T}_B$ makes these bodies indirectly accessible. With Programs Π_a^n, assigning "false" to an atom c_i $(1 \leq i \leq n)$ leads to a conflict by propagation. For *nomore++*^{B}, transformation $\mathcal{T}_a$ makes these atoms indirectly accessible. The programs are composed in such a way that lookahead on the crucial structures (bodies and atoms, respectively) is able to de-

[1] Universität Potsdam, Postfach 90 03 27, D–14439 Potsdam, Germany
[2] Helsinki University of Technology, P.O. Box 5400, FI-02015 TKK, Finland.
[3] Affiliated with Simon Fraser University, Canada.
[4] To distinguish full-fledged *nomore++* in (iii) from this restricted variant, we sometimes add superscript $a + B$, yielding *nomore++*$^{a+B}$.

termine the optimal choices and thus ensures that the minimal number of choices is obtained. [5] Unlike this, detecting the unsatisfiability of Π_B^n with the atom-based *smodels* strategy requires exponentially many choices. The same holds for Π_a^n with *nomore++*^{B}'s strategy. The *dlv* system employs a slightly different lookahead strategy with fewer lookahead operations, which is why it is unable to identify the best choices for the given program classes. We give *dlv* results purely for the record.

We have implemented the above transformations and run benchmarks on a number of instances of Π_B^n and Π_a^n and their transforms. The implementation is available at [6]. Results of these benchmarks are given in Table 1 and 2. All tests were run on an AMD Athlon 1.4GHz PC with 512MB RAM. We limited memory to 256MB and time to 900s. All results are given in terms of number of choices[6] and seconds (in parentheses), reflecting the average of 10 runs. The application of a translation is indicated by $\mathcal{T}_B$ or $\mathcal{T}_a$, respectively, otherwise a hyphen indicates no such application. We see that the application of transformation $\mathcal{T}_B$ leads to an exponential speed-up for *smodels* on Π_B^n, while $\mathcal{T}_a$ exponentially speeds up *nomore++*^{B} on Π_a^n. On the other hand, a superfluous transformation (as in the case of $\mathcal{T}_a(\Pi_a^n)$ when running *smodels*) seems not to hurt the performance significantly. In terms of choices, the hybrid approach of *nomore++*$^{a+B}$ lets it perform optimally on both Π_B^n and Π_a^n.

	dlv (2006.01.12)		*smodels* (2.28)		*nomore++* (1.4)			
					B		a+B	
n	–	$\mathcal{T}_B$	–	$\mathcal{T}_B$	–	$\mathcal{T}_B$	–	$\mathcal{T}_B$
2	(0.00)	(0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)
4	(0.00)	(0.01)	3 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)
6	(0.00)	(0.01)	15 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.02)
8	(0.00)	(0.01)	63 (0.01)	0 (0.01)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
10	(0.01)	(0.03)	255 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
12	(0.05)	(0.07)	1,023 (0.02)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
14	(0.20)	(0.24)	4,095 (0.03)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
16	(0.81)	(0.93)	16,383 (0.12)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
18	(3.25)	(3.76)	65,535 (0.45)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
20	(13.16)	(15.05)	262,143 (1.77)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
22	(53.06)	(61.17)	1,048,575 (7.09)	0 (0.02)	0 (0.01)	0 (0.03)	0 (0.01)	0 (0.02)
24	(213.93)	(247.57)	4,194,303 (28.54)	0 (0.02)	0 (0.01)	0 (0.03)	0 (0.01)	0 (0.02)
26	(862.69)	> 900	16,777,215 (114.56)	0 (0.02)	0 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
28	> 900	> 900	67,108,863 (460.30)	0 (0.02)	0 (0.02)	0 (0.03)	0 (0.02)	0 (0.03)
30	> 900	> 900	> 900	0 (0.03)	0 (0.02)	0 (0.03)	0 (0.02)	0 (0.03)

Table 1. Results for family $\{\Pi_B^n\}$.

	dlv (2006.01.12)		*smodels* (2.28)		*nomore++* (1.4)			
					B		a+B	
n	–	$\mathcal{T}_a$	–	$\mathcal{T}_a$	–	$\mathcal{T}_a$	–	$\mathcal{T}_a$
2	(0.00)	(0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)
4	(0.00)	(0.01)	0 (0.01)	0 (0.01)	3 (0.01)	0 (0.01)	0 (0.01)	0 (0.01)
6	(0.00)	(0.01)	0 (0.01)	0 (0.01)	15 (0.01)	0 (0.02)	0 (0.01)	0 (0.02)
8	(0.01)	(0.02)	0 (0.01)	0 (0.02)	63 (0.02)	0 (0.02)	0 (0.01)	0 (0.02)
10	(0.02)	(0.05)	0 (0.01)	0 (0.02)	255 (0.03)	0 (0.02)	0 (0.01)	0 (0.02)
12	(0.06)	(0.16)	0 (0.01)	0 (0.01)	1,023 (0.07)	0 (0.02)	0 (0.01)	0 (0.02)
14	(0.24)	(0.60)	0 (0.01)	0 (0.02)	4,095 (0.25)	0 (0.02)	0 (0.01)	0 (0.02)
16	(0.95)	(2.47)	0 (0.01)	0 (0.02)	16,383 (0.98)	0 (0.02)	0 (0.01)	0 (0.02)
18	(3.86)	(10.18)	0 (0.01)	0 (0.02)	65,535 (3.88)	0 (0.03)	0 (0.01)	0 (0.02)
20	(15.61)	(42.04)	0 (0.01)	0 (0.02)	262,143 (15.51)	0 (0.03)	0 (0.02)	0 (0.02)
22	(63.36)	(173.98)	0 (0.01)	0 (0.02)	1,048,575 (61.99)	0 (0.03)	0 (0.02)	0 (0.02)
24	(257.14)	(718.17)	0 (0.01)	0 (0.02)	4,194,303 (248.74)	0 (0.03)	0 (0.02)	0 (0.02)
26	> 900	> 900	0 (0.02)	0 (0.02)	> 900	0 (0.03)	0 (0.02)	0 (0.03)
28	> 900	> 900	0 (0.01)	0 (0.03)	> 900	0 (0.04)	0 (0.02)	0 (0.03)
30	> 900	> 900	0 (0.02)	0 (0.03)	> 900	0 (0.04)	0 (0.02)	0 (0.03)

Table 2. Results for family $\{\Pi_a^n\}$.

Finally, let us show that our observation is not accidental and formally prove that the availability of both atoms and bodies may lead to exponentially smaller search spaces than obtainable in either restricted case. To this end, we apply well-known concepts from *proof complexity* [4] and show that there are infinite witnessing families of programs for which even the best-case complexity is exponential. To be more precise, we show that minimal refutations for Π_B^n and Π_a^n, i.e. proofs that the respective programs have no answer sets, require exponentially many choices in the cases of Π_B^n with *smodels* and Π_a^n with *nomore++*^{B}, while a linear number of choices is required for the transformed programs.

Theorem 1 *There is an infinite family $\{\Pi^n\}$ of logic programs such that the minimal number of choices made by smodels for determining the unsatisfiability of Π^n is $O(2^n)$ and of $\mathcal{T}_B(\Pi^n)$ is $O(n)$.*

Theorem 2 *There is an infinite family $\{\Pi^n\}$ of logic programs such that the minimal number of choices made by nomore++*^{B} *for determining the unsatisfiability of Π^n is $O(2^n)$ and of $\mathcal{T}_a(\Pi^n)$ is $O(n)$.*

In neither case is it possible to simulate this behavior polynomially, as witnessed by families $\{\Pi_B^n\}$ and $\{\Pi_a^n\}$, respectively. Finally, we note that the hybrid approach pursued by *nomore++*$^{a+B}$ gives a best-case complexity of $O(n)$ for both classes of programs: A hybrid approach can polynomially simulate both uniform approaches.

2 DISCUSSION

We have shown that the integration of bodies as explicitly referable objects into atom-based ASP-solvers may have a great computational impact. The same holds for purely body-based (or rule-based) approaches, when introducing special atoms. Considering bodies rather than more complex formulas is motivated by the fact that they allow for replacing rules in characterizing computational concepts (see below) and by applying an "input strategy" (similar to *linear resolution*). Very simple transformations allow for exponential reductions of the best-case complexity on witnessing families of logic programs. That is, any uniform ASP-solver even equipped with the optimal heuristics will be unable to solve all witnessing programs in polynomial time. This is only possible by means of a genuinely hybrid approach, as realized in *nomore++*, or via respective transformations.

Our discovery is insofar surprising as standard ASP-solvers, such as *smodels*, rely already on twofold data structures but somehow stop "halfway": The abstract *smodels* algorithm [7] relies on atoms (in branching and assigning truth values). However, underlying propagation and its implementation must also take rules (or bodies) into account. Consequently, the primary data structure of *smodels* is an "atom-rule graph." Furthermore, propagation takes advantage of the concept of "active" rules, that is, rules whose bodies are not false [7]. Although bodies are vital objects in *smodels*' propagation, they are ignored when branching.

Acknowledgements. The first, second, and fourth author were supported by DFG under grant SCHA 550/6-4, TP C.

REFERENCES

[1] C. Anger, M. Gebser, T. Linke, A. Neumann, and T. Schaub, 'The nomore++ approach to answer set solving', in *Proceedings of LPAR'05*, eds., G. Sutcliffe and A. Voronkov, pp. 95–109. Springer-Verlag, (2005).

[2] M. Baaz, U. Egly, and A. Leitsch, 'Normal form transformations', in *Handbook of Automated Reasoning*, eds., J. Robinson and A. Voronkov, 273–333, Elsevier, (2001).

[3] C. Baral, *Knowledge Representation, Reasoning and Declarative Problem Solving*, Cambridge University Press, 2003.

[4] P. Beame and T. Pitassi, 'Propositional proof complexity: Past, present, and future', *Bulletin of EATCS*, **65**, 66–89, (1998).

[5] N. Leone, W. Faber, G. Pfeifer, T. Eiter, G. Gottlob, C. Koch, C. Mateis, S. Perri, and F. Scarcello, 'The DLV system for knowledge representation and reasoning', *ACM TOCL*, (2006). To appear.

[6] http://www.cs.uni-potsdam.de/nomore.

[7] P. Simons, 'Extending and implementing the stable model semantics', Helsinki University of Technology, (2000). Doctoral dissertation.

[5] With lookahead disabled, propagation does not suffice to detect unsatisfiability. Thus, linearly many choices must be made.

[6] The choices of *dlv* are omitted since they rely on a different concept.