A Polynomial Reduction from ASPDA to ASP*

Wolfgang Faber

Department of Mathematics
University of Calabria
87030 Rende (CS), Italy
wf@wfaber.com

Abstract. ASPDA is a framework for expressing defeasibility in Answer Set Programs via so-called argumentation theories, proposed by Wan, Kifer, and Grosof in [2]. The authors describe a reduction from ASPDA to plain Answer Set Programming, which however exponentially inflate programs. In this note, we present an alternative reduction, which does not suffer from this problem. As a side-effect, complexity results for ASPDA are established.

1 Introduction

ASPDA is a framework for expressing defeasibility in Answer Set Programs via so-called argumentation theories, proposed by Wan, Kifer, and Grosof in [2]. ASPDA programs provide a very general means for defeating literals and rules and capture several earlier proposals for defeasibility in logic programming. In [2], the authors also provide a reduction from ASPDA to ASP (in the sense of [1]). However, this reduction can easily lead to ASP programs that are exponentially larger than the original ASPDA programs. In this paper, we show that this exponential behavior is not necessary, by providing an alternative reduction. Different to the reduction in [2], ours introduces new symbols and also needs a concept of rule identifier, which make proving correctness of the reduction slightly more cumbersome. However, this reduction immediately provides complexity results for ASPDA, in particular showing that (virtually all) computational tasks over ASPDA programs have the same complexity as those over ASP programs.

2 ASPDA: Syntax and Semantics

We briefly review syntax and semantics of ASPDA, for details we refer to [2]. The language assumes a set of atoms; in [2] this set is not fixed, here we assume it to consist of first-order or propositional atomic formulas. There are two kinds of negation, and a literal is either an atom A, `neg` A, `naf` A, or `naf neg` A. A rule is of the form

$$@r\ L_1 \vee \cdots \vee L_k :- Body \tag{1}$$

where $k \geq 0$, r is a term and the tag of the rule (different rules can share the same rule tag), each L_i ($0 < i \leq k$) is a literal, and $Body$ is a conjunction of literals. Given a rule

* This work was supported by M.I.U.R. within the PRIN project LoDeN.

of the form (1), the term $h(r, L_i)$ ($handle(r, L_i)$ in [2]) is the handle for each of the head literals L_i ($0 \leq i \leq k$). Each rule can be either defeasible or strict.

An argumentation theory AT is a set of strict rules of the form (1), which makes use of a distinguished predicate $\$defeated_{AT}$ that may occur only in rule heads. The subscript AT is usually omitted when the context is clear. An answer-set program with defaults and argumentation theories (ASPDA) is a set of rules of the form (1), which may comprise an argumentation theory. In [2], the argumentation theory is usually considered separated from the program, but since it is syntactically and semantically the same as a special kind of program, we consider it as part of the program for simplicity.

Herbrand universe and base are defined in the standard way, where the Herbrand base consists not just of ground atoms, but of ground `naf`-free literals. An (Herbrand) interpretation is a subset of the Herbrand base, and we will assume consistent interpretations, i.e. no interpretation contains both A and `neg` A.

A `naf`-free literal L is true in an interpretation I if $L \in I$, `naf` L is true in I if $L \notin I$; otherwise these literals are false in I. A strict rule is satisfied in I if at least one head literal is true in I whenever all body literals are true in I. A defeasible rule of the form (1) is satisfied if it either meets the condition for a strict rule or if $\$defeated(h(r, L_i))$ is true in I for all $0 < i \leq k$. As usual, an interpretation I is a model of an ASPDA program P if I satisfies all rules in P. A model of a program P is minimal if none of its subsets is a model of P.

For defining answer sets, [2] define the quotient $\frac{P}{I}$ for an ASPDA program P and an interpretation I in four steps: (i) Delete every rule in P in which a `naf` body literal is false in I; (ii) in each defeasible rule of the form (1), delete all L_i that are true in I; if all L_i are deleted, delete the complete rule; (iii) remove all `naf`-literals of the remaining rules; (iv) remove tags from the remaining rules. An interpretation I is an answer set of an ASPDA P if I is a minimal model of $\frac{P}{I}$.

Traditional ASP can be viewed as a special case of ASPDA. ASPDA programs that have no defeasible rules and empty argumentation theory can be viewed as ASP programs. It is easy to show that the quotient $\frac{P}{I}$ coincides with the reduct P^I [1] for such programs (the only difference are the rule tags, which are irrelevant for these programs).

3 A Polynomial Reduction from ASPDA to ASP

In [2] a reduction from ASPDA to ASP is provided that preserves answer sets, which however, produces an exponential number of rules in general. We provide an alternative reduction, which does not suffer from this exponential increase in size.

Definition 1. *Given an ASPDA P, for each defeasible rule of the form (1), create*

$$\$der(r, L_1) \vee \cdots \vee \$der(r, L_k) :- Body, \texttt{naf}\ \$rdef(rid) \tag{2}$$

$$\$rdef(rid) :- \$defeated(h(r, L_1)), \ldots, \$defeated(h(r, L_k)) \tag{3}$$

where $\$rdef$ and $\$der$ are fresh predicates, rid is a rule identifier (obtained for example by the index of a fixed enumeration of rules; note that the rule tag cannot serve as

the rule identifier), and for each $0 < i \leq k$ *create*

$$L_i :- \ \$der(r, L_i) \tag{4}$$
$$\$der(r, L_i) :- \ L_i, \texttt{naf}\ \$defeated(h(r, L_i)) \tag{5}$$
$$:- \ \$der(r, L_i), \$defeated(h(r, L_i)) \tag{6}$$

For each strict rule, delete its rule tag. We refer to the obtained program as $tr(P)$.

Example 1. For $@r\ a \vee b :-$ the reduction of [2] generates

$$a \vee b :- \texttt{naf}\ \$defeated(h(r, a)), \texttt{naf}\ \$defeated(h(r, b))$$
$$a :- \texttt{naf}\ \$defeated(h(r, a)), \$defeated(h(r, b))$$
$$b :- \$defeated(h(r, a)), \texttt{naf}\ \$defeated(h(r, b))$$

(and also $:-\$defeated(h(r, a)), \$defeated(h(r, b))$, but this seems to be due to a typo). The reduction of Definition 1 generates

$\$der(r, a) \vee \$der(r, b) :- \texttt{naf}\ \$rdef(rid) \qquad a :- \$der(r, a) \qquad b :- \$der(r, b)$
$\$rdef(rid) :- \$defeated(h(r, a)), \$defeated(h(r, b))$
$\$der(r, a) :- a, \texttt{naf}\ \$defeated(h(r, a)) \qquad :- \$der(r, a), \$defeated(h(r, a))$
$\$der(r, b) :- b, \texttt{naf}\ \$defeated(h(r, b)) \qquad :- \$der(r, b), \$defeated(h(r, b))$

In general, for each rule with k head literals, the reduction of Definition 1 creates $3k+2$ rules, while the one of [2] creates $2^k - 1$ rules.

Theorem 1. *Given an ASPDA* P*, there is a one-to-one relationship between the answer sets of* P *and those of* $tr(P)$*. In particular, for each answer set* A *of* P*,* $tr(A) = A \cup \{\$der(r, L) \mid$ *a defeasible rule with tag* r *in* P *exists with true body and* L *in its head, s.t.* $\$defeated(h(r, L)) \notin A$ *and* $L \in A\ \} \cup \{\$rdef(rid) \mid$ *a defeasible rule with identifier* rid *and tag* r *in* P *exists s.t. for each head literal* L*,* $\$defeated(h(r, L)) \in A$ *holds* $\}$ *is an answer set of* $tr(P)$*, and these are the only answer sets of* $tr(P)$.

Proof. Assume that A is an answer set of P (hence a minimal model of $\frac{P}{A}$). We show that $tr(A)$ is a minimal model of $\frac{tr(P)}{tr(A)}$. First observe that for each rule in P which is deleted in step (i) of the definition of $\frac{P}{A}$, the rule itself or its corresponding rule (4) is not in $\frac{tr(P)}{tr(A)}$ either. Moreover, a defeasible rule in P for which $\$defeated(h(r, L)) \in A$ holds for all head literals L is not in $\frac{P}{A}$ due to step (ii) of the definition of $\frac{P}{A}$, and no reduct of the corresponding rule (4) is in $\frac{tr(P)}{tr(A)}$ either, since by construction $\$rdef(rid) \in tr(A)$. For all other defeasible rules of form (1) in P (i.e. those not deleted in steps (i) and (ii) of the definition of $\frac{P}{A}$), $\frac{P}{A}$ contains $\bigvee_{L \in K} L :- Body'$, s.t. K is the set of head literals s.t. $\$defeated(h(r, L) \notin A$ and $Body'$ is $Body$ without `naf`-literals. $\frac{tr(P)}{tr(A)}$ instead has $\$der(r, L_1) \vee \cdots \vee \$der(r, L_k) :- Body'$, $\$der(r, L_i) :- L_i$ for L_i s.t. $\$defeated(h(r, L_i)) \notin A$ and also all rules of type (3), (4), and (6). By construction, $tr(A)$ is a model of $\frac{tr(P)}{tr(A)}$. To see minimality, observe that $\$der(r, L_i)$ take the place of L_i in rule heads of reducts in $\frac{tr(P)}{tr(A)}$. So if there is a model $N \subsetneq tr(A)$ for

$\frac{tr(P)}{tr(A)}$ not containing some $\$der(r, L_i)$, then also L_i must not be in that model because of a rule of type (4). Removing other literals from $tr(A)$ cannot yield a model of $\frac{tr(P)}{tr(A)}$.

Assume now that M is an answer set for $tr(P)$. We show that $M = tr(A)$ for some answer set A of P. Let M' denote the set M after removing all literals $\$der(r, L)$ and $\$rdef(rid)$. First we note that M' satisfies all strict rules in the reduct $\frac{P}{M'}$. Concerning defeasible rules, if $\$rdef(rid) \in M$, then the rule with identifier rid is not in $\frac{P}{M'}$. Otherwise, rule (2) is in $\frac{tr(P)}{M}$ and a corresponding rule r' obtained from r is also in $\frac{P}{M'}$. Note that r' in general has fewer head literals than (2), however, we observe that for each L_i that was removed from r when creating r' in $\frac{P}{M'}$ there is a rule (6) and for each of these L_i, $\$defeated(h(r, L_i)) \in M'$ and $\$defeated(h(r, L_i)) \in M$ and so $\$der(r, L_i) \notin M$. Hence if rule (2) has a true body in M, $\$der(r, L_j) \in M$ holds only if $\$defeated(h(r, L_j)) \notin M$ and $\$defeated(h(r, L_j)) \notin M'$. Moreover, rules (4) enforce that $L_j \in M$ and $L_j \in M'$. It follows that all rules in $\frac{P}{M'}$ that stem from defeasible rules are satisfied by M' and hence M' is a model of $\frac{P}{M'}$. Minimality of M' then follows from the minimality of M and the fact that the satisfaction patterns of rule heads of the reducts of defeasible rules and the corresponding reducts of rules (2) coincide. Therefore M' is an answer set of P and $M = tr(M')$.

The computational complexity of reasoning tasks over ASPDA programs was left open in [2]. It is obvious that the reduction in Definition 1 runs in polynomial time, hence the reduction provides a tight upper bound for the complexity of all computational tasks of ASPDA, where the corresponding task for ASP is at least polynomial.

Corollary 1. *Given a computational task over ASP programs wwhich is complete for located a complexity class that contains P, the corresponding task over ASPDA programs is located in the same complexity class.*

Since traditional ASP programs are a special case of ASPDA, lower bounds extend trivially from ASP to ASPDA.

4 Conclusion

We have provided an alternative reduction from ASPDA to ASP, which avoids an exponential increase in space and thus is an immediate improvement over an analogous reduction in [2]. Contrary to the earlier reduction, it makes use of additional symbols and also needs the concept of a rule identifier. As an immediate consequence of the reduction, we obtain results on the computational complexity of computational tasks over ASPDA, which coincide with those of ASP for practically all relevant tasks.

References

1. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, 365–385 (1991)
2. Wan, H., Kifer, M., Grosof, B.N.: Defeasibility in answer set programs via argumentation theories. In: Hitzler, P., Lukasiewicz, T. (eds.) Web Reasoning and Rule Systems - Fourth International Conference (RR 2010). LNCS, vol. 6333, pp. 149–163. (2010)