Iteratively-Supported Formulas and Strongly Supported Models for Kleene Answer Set Programs (Extended Abstract)

Patrick Doherty[1], Jonas Kvarnström[1], and Andrzej Szałas[1,2]

[1] Dept. of Computer and Information Science
Linköping University, SE-581 83 Linköping, Sweden
patrick.doherty@liu.se, jonas.kvarnstrom@liu.se
[2] Institute of Informatics, University of Warsaw, Banacha 2, 02-097 Warsaw, Poland
andrzej.szalas@{liu.se, mimuw.edu.pl}

Abstract. In this extended abstract, we discuss the use of iteratively-supported formulas (ISFs) as a basis for computing strongly-supported models for Kleene Answer Set Programs (ASP^K). ASP^K programs have a syntax identical to classical ASP programs. The semantics of ASP^K programs is based on the use of Kleene three-valued logic and strongly-supported models. For normal ASP^K programs, their strongly supported models are identical to classical answer sets using stable model semantics. For disjunctive ASP^K programs, the semantics weakens the minimality assumption resulting in a classical interpretation for disjunction. We use ISFs to characterize strongly-supported models and show that they are polynomially bounded.

1 Introduction

Classical answer set programming, ASP, has been intensively studied during the past three decades [3, 5, 9]. In addition, a great deal of attention has been devoted to ASP implementations [4, 7, 8, 12, 16]. One of the prominent techniques proposed earlier for computing answer sets is based on translating ASP programs into classical propositional formulas and then applying SAT solvers to generate answer sets. In [6, 12] it is shown that Clark's completion together with loop formulas characterize answer sets for ASP programs. One of the obstacles in characterizing answer sets using propositional formulas is their Σ_2^P complexity. Loop formulas contribute to this because one may require exponentially many of them [10]. The current extended abstract provides an alternative to loop formulas, iteratively-supported formulas, that ameliorates this problem. Polynomial translations of normal ASP programs have also been considered in [11, 13, 14]. However, our translation is extended to disjunctive programs in a natural way.

In [15] a possible model semantics for disjunctive programs is proposed. It is formulated with the use of split programs and there can be exponentially many of them comparing to the original program. Similar semantics was independently proposed in [1] under the name of the possible world semantics. In [2] we have analyzed minimality and supportedness in the context of ASPs and proposed Kleene Answer Set Programs (ASP^K) using the concept of strongly supported models.The semantics used for Kleene Answer Set programs is based on Kleene logic, K_3, with an extra weak negation. In [2]

it is shown that the problem of showing whether an ASP^K program has a strongly supported model is in NP (i.e., in Σ_1^P). This result applies to both normal and disjunctive ASP^K programs. For disjunctive ASP^K programs, the minimality assumption is relaxed, resulting in a classical interpretation of disjunction.[3] The ability to fine-tune the separation of supportedness and minimality in the disjunctive case results in a lower complexity for generating strongly supported models. In comparison to [15], ASP^K programs allow for strong negation and a three-valued model-theoretic semantics is provided. The presence of both default and strong negation in ASP^K provides a tool to close the world locally in a contextual manner, more flexible than possible model negation proposed in [15]. Though defined independently and using different foundations, both semantics appear compatible on positive programs, so the results of the current paper apply to possible model semantics of [15], too.

The main contribution of the current paper is the definition and use of ISFs to characterize strongly supported models for both normal and disjunctive ASP^K programs. Such formulas are shown to be polynomially bounded in both cases. As a derivative result, in the case of normal ASP programs and due to a correspondence between answer sets and strongly supported models, ISFs provide a more efficient alternative to loop formulas when using SAT solvers. For disjunctive ASP^K programs, the use of supported models and ISFs provide an efficient means for using SAT solvers, but with an alternative semantics that interprets disjunction classically due to a relaxation of minimality assumptions.

The paper is structured as follows. In Section 2 we introduce basic definitions related to both classical ASP programs and ASP^K programs in addition to strong supportedness. Section 3 introduces ISFs used to characterize normal and disjunctive ASP^K programs. Section 4 concludes the paper.

2 Kleene Answer Set Programs

In this paper, the syntax for Kleene ASP^K programs is identical for that of classical ASP programs. The semantics for Kleene ASP^K programs is based on the use of a three-valued Kleene logic K_3 and strongly-supported models presented in [2]. The semantics for classical ASP programs is based on stable model semantics [9]. For the sake of clarity we consider propositional programs only. Truth values are denoted by T (true), F (false) and U (unknown). The empty conjunction is T and the empty disjunction is F.

Definition 1. By a *positive literal* (or an *atom*) we mean any propositional variable of $\mathcal{P}$. A *negative literal* is an expression of the form $\neg r$, where $r \in \mathcal{P}$. A *classical literal* is a positive or a negative literal. A set of literals is *consistent* if it does not contain a literal ℓ together with its negation $\neg\ell$.[4] By an *extended literal* we understand a classical literal or an expression of the form $not\ \ell$, where ℓ is a classical literal. If γ is an expression (formula, program, etc.) then $Lit(\gamma) \stackrel{\text{def}}{=} \{p, \neg p \mid p \in \mathcal{P}$ occurs in $\gamma\}$ and $\mathcal{P}(\Pi) \stackrel{\text{def}}{=} \mathcal{P} \cap Lit(\Pi)$.

[3] Note that minimality is sometimes not required or may even be undesirable [2, 3, 15, 17], e.g., in the context of programs that use disjunctive rules.

[4] We always remove double strong negations using $\neg(\neg\ell) \stackrel{\text{def}}{=} \ell$.

An *interpretation* is a finite consistent set of literals. Interpretation I *satisfies* a classical literal ℓ iff $\ell \in I$ and I *satisfies* an extended literal $not\,\ell$ iff $\ell \notin I$. The satisfiability relation is denoted by $I \models \ell$. ◁

Definition 2. By an ASP^K rule we understand an expression ϱ of the form:

$$\ell_1 \vee \ldots \vee \ell_k \leftarrow \ell_{k+1}, \ldots, \ell_m, not\,\ell_{m+1}, \ldots, not\,\ell_n, \tag{1}$$

where $n \geq m \geq k \geq 0$, $\ell_1, \ldots, \ell_k, \ell_{k+1}, \ldots, \ell_m, \ell_{m+1}, \ldots \ell_n$ are (positive or negative) literals. The expression at the lefthand side of '$\leftarrow$' in (1), denoted by $h(\varrho)$, is called the *head* and the righthand side of '$\leftarrow$', denoted by $B(\varrho)$, is called the *body* of the rule. The rule is called *disjunctive* if $k > 1$.

An ASP^K *program* Π is a finite set of rules. A program is *normal* if each of its rules has at most one literal in its head. If a program contains a disjunctive rule, we call it *disjunctive*. By $Disj(\Pi)$ we denote the set of disjunctive rules appearing in Π.

The set of rules with the empty body is denoted by $Fct(\Pi)$ and the set of rules with the empty head is denoted by $Ctr(\Pi)$. Members of $Fct(\Pi)$ and $Ctr(\Pi)$ are called *facts* and *constraints*, respectively. The set of rules whose bodies and heads are nonempty is denoted by $Rul(\Pi)$.

An interpretation I *satisfies* a rule ϱ of the form (1), denoted by $I \models \varrho$, if whenever $\ell_{k+1}, \ldots, \ell_m \in I$ and $\ell_{m+1}, \ldots, \ell_n \notin I$, we have $\ell_i \in I$ for some $1 \leq i \leq k$. An interpretation I *satisfies* an ASP^K program Π, denoted by $I \models \Pi$, if for all rules $\varrho \in \Pi$, $I \models \varrho$. ◁

The following definition is needed to define strong supportedness (a construction similar in spirit is considered in [18]).

Definition 3. Given interpretations I and J, the *value of a formula A w.r.t.* (I, J), denoted by $(I, J)(A)$, is defined as follows:

$$(I, J)(A) \stackrel{\text{def}}{=} \begin{cases} \text{T} \text{ when } I \models reduct^J(A); \\ \text{F} \text{ when } I \models reduct^J(\neg A); \\ \text{U} \text{ otherwise.} \end{cases} \tag{2}$$

where $reduct^J(A)$ (respectively, $reduct^J(\neg A)$) is a formula obtained from A ($\neg A$) by substituting subformulas of the form $not\,\ell$ by their truth values evaluated in J. ◁

Definition 4. An interpretation N is a *strongly supported model* of an ASP^K program Π provided that N satisfies Π and there exists a sequence of interpretations $I_0 \subseteq I_1 \subseteq \ldots \subseteq I_n$ where $n \geq 0$ such that $I_0 = Fct(\Pi)$, $N = I_n$, and:

1. for every $1 \leq i \leq n$ and every rule $\ell_1 \vee \ldots \vee \ell_k \leftarrow B$ of Π,
   if $\big(I_{i-1}, N\big)(B) = \text{T}$ then a nonempty subset of $\{\ell_1, \ldots, \ell_k\}$
   is included in I_i;
2. for $i = 1, \ldots, n$, I_i can only contain literals obtained by applying point 1. ◁

3 Iteratively-Supported Formulas

Let p be a propositional variable. Then p_i (respectively $\bar{p}_i$) denotes the fact that in the i-th iteration, p (respectively, $\neg p$) is in the computed candidate for a strongly supported

model. Thus, $\neg p_i$ (respectively $\neg \bar{p}_i$) denotes the fact that in the i-th iteration p_i (respectively, $\neg p_i$) is *not* in the computed candidate for a strongly supported model.

The number of different literals in heads of $Rul(\Pi)$ is denoted by $\#\Pi$. Since support can only be generated for up to $\#\Pi$ distinct literals, $\#\Pi$ iterations will be sufficient to provide support for all literals in any strongly supported model.

Definition 5. The translation function is defined as follows, where $1 \leq i \leq \#\Pi$ and ℓ is an extended literal:

$$Tr_\Pi(i,\ell) \stackrel{\text{def}}{=} \begin{cases} p_i & \text{when } \ell = p; \\ \bar{p}_i & \text{when } \ell = \neg p; \\ \neg p_{\#\Pi} & \text{when } \ell = not\, p; \\ \neg \bar{p}_{\#\Pi} & \text{when } \ell = not\, \neg p. \end{cases} \tag{3}$$

We extend the translation for bodies and heads of rules by setting:

$$Tr_\Pi(i,B) \stackrel{\text{def}}{=} \bigwedge_{\ell \in B} Tr_\Pi(i,\ell) \text{ and } Tr_\Pi(i,H) \stackrel{\text{def}}{=} \bigvee_{\ell \in H} Tr_\Pi(i,\ell).$$

Definition 6. By a *support* of a classical literal ℓ in a normal ASP^K program Π at i ($i > 0$) we understand the formula:

$$Supp^i_\Pi(\ell) \stackrel{\text{def}}{=} \big[\, Tr_\Pi(i,\ell) \equiv \big(Tr_\Pi(i-1,\ell) \vee \bigvee_{\varrho \in \Pi : \ell = h(\varrho)} Tr_\Pi(i-1, B(\varrho)) \big) \,\big]. \tag{4}$$

Definition 7. By the *iteratively-supported formula* for a normal ASP^K program Π we understand the following formula of classical propositional calculus:

$$ISF(\Pi) \stackrel{\text{def}}{=} \bigwedge_{0 \leq i \leq \#\Pi} \bigwedge_{p \in \mathcal{P}(\Pi)} \neg\big(p_i \wedge \bar{p}_i\big) \wedge \tag{5}$$

$$\bigwedge_{F \in Fct(\Pi)} Tr_\Pi(0, h(F)) \;\wedge \bigwedge_{\ell \in Lit(\Pi) - \{h(F) \mid F \in Fct(\Pi)\}} \neg\, Tr_\Pi(0,\ell) \wedge \tag{6}$$

$$\bigwedge_{1 \leq i \leq \#\Pi} \bigwedge_{\ell \in Lit(\Pi)} Supp^i_\Pi(\ell) \wedge \tag{7}$$

$$\bigwedge_{\varrho \in \Pi} \big(Tr_\Pi(\#\Pi, B(\varrho)) \rightarrow Tr_\Pi(\#\Pi, h(\varrho)) \big). \tag{8}$$

We have the following theorem for normal ASP^K programs.

Theorem 1. For any normal ASP^K program Π, I is a strongly supported model of Π iff there is a valuation v satisfying $ISF(\Pi)$ such that:

$$I = \{p \mid v(p_{\#\Pi}) = \mathrm{T}\} \cup \{\neg p \mid v(\bar{p}_{\#\Pi}) = \mathrm{T}\}. \qquad \lhd$$

Since for normal ASP^K programs strongly supported models are also classical answer sets, Theorem 1 applies to classical ASP, too.

Given a disjunctive ASP^K program Π, the support of literals appearing only in non-disjunctive heads remains unchanged. For literals appearing in disjunctive heads we have the following definition.

Definition 8. By a *support* of a classical literal ℓ occurring in a disjunctive head in an ASP^K program Π at i ($i > 0$) we understand the formula:

$$Supp^i_\Pi(\ell) \stackrel{\text{def}}{=} \big[\, Tr_\Pi(i,\ell) \rightarrow \big(\, Tr_\Pi(i-1,\ell) \;\vee \bigvee_{\varrho\in\Pi:\ell\in h(\varrho)} Tr_\Pi(i-1,B(\varrho))\big)\,\big] \wedge \big[\, Tr_\Pi(i-1,\ell) \;\rightarrow Tr_\Pi(i,\ell)\,\big]. \tag{9}$$

For other literals, the support of ℓ is still specified by formula (4) in Definition 6. ◁

Definition 9. By an *iteratively-supported formula* for a disjunctive ASP^K program Π we understand the formula (7) with $Supp^i_\Pi()$ understood as in Definition 8. ◁

We now have the following generalization of Theorem 1.

Theorem 2. For any (normal or disjunctive) ASP^K program Π, I is a strongly supported model of Π iff there is a valuation v satisfying $ISF(\Pi)$ such that:

$$I = \{p \mid v(p_{\#\Pi}) = \mathrm{T}\} \cup \{\neg p \mid v(\bar{p}_{\#\Pi}) = \mathrm{T}\}.$$ ◁

Note that for any ASP^K program Π, the number of different literals in heads of $Rul(\Pi)$ (i.e., $\#\Pi$) is linear in the size of Π. Therefore we have the following lemma.

Lemma 1. For any (normal or disjunctive) ASP^K program Π, the size of $ISF(\Pi)$ is polynomial in the size of Π. ◁

4 Conclusions

In this extended abstract, we have defined iteratively-supported formulas expressed in classical propositional logic and used them to characterize strongly supported models for ASP^K programs. For normal ASP^K programs, I is a classical answer set of the program iff I is a strongly supported model of the program. Since iteratively-supported formulas provide polynomially bounded characterizations of supported models for normal ASP^K programs, they also provide polynomially bounded characterizations of classical answer sets for normal ASP programs. In contrast, use of loop formulas could result in formulas of exponential size for normal ASP programs..

ISFs also characterize strongly supported models for disjunctive ASP^K programs and guarantee that all conclusions are grounded in facts or default reasoning based on extended literals (using default negation not). Additionally, due to a weakened minimization assumption, disjunction is interpreted classically which results in a semantics enjoying among other properties, a Σ^P_1 complexity for computing strongly-supported models. This, together with a polynomial bound on ISFs, is a striking theoretical improvement compared to the Σ^P_2 complexity of computing classical answer sets for ASP programs.

Acknowledgments This work is partially supported by the Swedish Research Council (VR) Linnaeus Center CADICS, the ELLIIT network organization for Information and Communication Technology, the Swedish Foundation for Strategic Research (CUAS Project, SymbiKCloud Project), the EU FP7 project SHERPA (grant agreement 600958), and Vinnova NFFP6 Project 2013-01206.

References

1. Chan, P.: A possible world semantics for disjunctive databases. IEEE Trans. Knowl. Data Eng. 5(2), 282–292 (1993)
2. Doherty, P., Szałas, A.: Stability, supportedness, minimality and Kleene Answer Set Programs. In: Eiter, T., Strass, H., Truszczyński, M., Woltran, S. (eds.) Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation, LNCS, vol. 9060, pp. 125–140. Springer International Publishing (2015)
3. Ferraris, P., Lifschitz, V.: On the minimality of stable models. In: Balduccini, M., Son, T. (eds.) Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning. LNCS, vol. 6565, pp. 64–73. Springer (2011)
4. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: *clasp* : A conflict-driven answer set solver. In: 9th Int. Conf. LPNMR.07. pp. 260–265 (2007)
5. Gelfond, M., Kahl, Y.: Knowledge Representation, Reasoning, and the Design of Intelligent Agents - The Answer-Set Programming Approach. Cambridge University Press (2014)
6. Lee, J., Lifschitz, V.: Loop formulas for disjunctive logic programs. In: Palamidessi, C. (ed.) Proc. ICLP'03. LNCS, vol. 2916, pp. 451–465. Springer (2003)
7. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Trans. Comput. Logic 7(3), 499–562 (2006)
8. Lierler, Y.: cmodels - SAT-based disjunctive answer set solver. In: Logic Programming and Nonmonotonic Reasoning, 8th International Conference, LPNMR 2005. pp. 447–451 (2005)
9. Lifschitz, V.: Thirteen definitions of a stable model. In: Blass, A., Dershowitz, N., Reisig, W. (eds.) Fields of Logic and Computation. LNCS, vol. 6300, pp. 488–503. Springer (2010)
10. Lifschitz, V., Razborov, A.: Why are there so many loop formulas? ACM Trans. Comput. Log. 7(2), 261–268 (2006)
11. Lin, F., Zhao, J.: On tight logic programs and yet another translation from normal logic programs to propositional logic. In: Gottlob, G., Walsh, T. (eds.) Proc. IJCAI-03. pp. 853–858. Morgan Kaufmann (2003)
12. Lin, F., Zhao, Y.: ASSAT: computing answer sets of a logic program by SAT solvers. Artif. Intell. 157(1-2), 115–137 (2004)
13. Liu, G., Janhunen, T., Niemelä, I.: Answer set programming via mixed integer programming. In: Brewka, G., T., E., McIlraith, S. (eds.) Proc. KR'12. AAAI Press (2012)
14. Pelov, N., Ternovska, E.: Reducing inductive definitions to propositional satisfiability. In: Gabbrielli, M., Gupta, G. (eds.) Proc. ICLP. LNCS, vol. 3668, pp. 221–234. Springer (2005)
15. Sakama, C., Inoue, K.: An alternative approach to the semantics of disjunctive logic programs and deductive databases. J. Autom. Reasoning 13(1), 145–172 (1994)
16. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence 138(1-2), 181–234 (2002)
17. Soininen, T., Niemelä, I.: Developing a declarative rule language for applications in product configuration. In: Gupta, G. (ed.) Proc. PADL'99. LNCS, vol. 1551, pp. 305–319. Springer (1999)
18. Son, T., Pontelli, E.: A constructive semantic characterization of aggregates in answer set programming. TPLP 7(3), 355–375 (2007)