On Establishing Robust Consistency in Answer Set Programs (Extended Abstract)*

Andre Thevapalan[1 [0000−0001−5679−6931]] and Gabriele Kern-Isberner[1 [0000−0001−8689−5391]]

Technische Universität Dortmund, 44227 Dortmund, Germany
{andre.thevapalan,gabriele.kern-isberner}@cs.tu-dortmund.de

1 Introduction

Knowledge bases of real-world applications are generally not static but rather very dynamic in the sense that they are adapted to each individual case and can also be prone to various updates. Imagine a system that outputs the possible treatment plans for a given patient based on a suitable set of program rules. For each patient, the application has to combine the general knowledge about possible treatment plans (*problem encoding*) with the data regarding the patient (*problem instance*) [1]. However, at any point where such an application is used, it has to be ensured that the respective program remains consistent when merged with the patient data. In this paper, we show how to ensure the robust consistency of a program in that it remains non-contradictory given any allowed set of such input data.

2 Resolving potential conflicts

Let $\mathcal{P}$ be an *extended logic program* (ELP) as defined in [2]. Additionally, by an *extended literal* L^*, we either mean a (classical) literal L or a default-negated literal $\sim L$. We distinguish between the set $\mathcal{P}_\mathcal{F}$ of facts (*input*) and the set $\mathcal{P}_\mathcal{C}$ of rules (*program core*). A *valid input* $\mathcal{P}_\mathcal{F}$ *for* $\mathcal{P}_\mathcal{C}$ is a consistent set of facts over atoms that occur in the rule bodies but not in any rule head. We presuppose that $\mathcal{P}_\mathcal{C}$ can only comprise rules with a head and a non-empty body. $\mathcal{P}$ is *contradictory* if there is a set M of classical literals that contains complementary literals and s. t. M is a minimal model of $\mathcal{P}^M$. The aim of our approach is to make the program core $\mathcal{P}_\mathcal{C}$ of $\mathcal{P}$ *uniformly non-contradictory*, i. e., for every valid input $\mathcal{P}_\mathcal{F}$ for $\mathcal{P}_\mathcal{C}$, $\mathcal{P}_\mathcal{C} \cup \mathcal{P}_\mathcal{F}$ is not contradictory.

Causes for potential contradictions in a program are *conflicting rules*. Two rules $r, s \in \mathcal{P}_\mathcal{C}$, $r \neq s$, are conflicting if $H(r)$ and $H(s)$ are complementary and there exists a consistent set M of classical literals s. t. both $B(r)$ and $B(s)$ are true in M [3]. We denote the set of rules that are conflicting with r by $Adv(r)$. For every pair (r, s) of conflicting rules, we add an extended literal to the body of r that shares no atoms with the original body literals. Such an extension is

* This extended abstract informally summarises the main contributions of [4].

called a *λ-extension for r,* $\lambda(r)$. A λ-extension $\lambda(r)$ for r is *conflict-resolving* iff $Adv(r') = \emptyset$ with r': $H(r) \leftarrow B(r), \lambda(r)$. Thus, conflict-resolving λ-extensions allow the resolution of multiple conflicts at once.

A main feature of our approach is that λ-extensions are *informative* in that $\lambda(r)$ only contains atoms that occur in the rule bodies of $Adv(r)$. This means that we explore the application-related structure of conflicts to come up with a professionally adequate solution. We introduce the technical term of blankets and apply suitable negations (N) to show our main result:

Corollary 3. *All conflicts involving r are resolved simultaneously if r is replaced by* $r' \in \{H(r) \leftarrow B(r), \lambda(r). \mid \lambda(r) \in N_{min}(blankets(Adv(r)))\}$.

Here, any λ-extension for r is conflict-resolving (*blankets*), informative (N), and subset-minimal (N_{min}). Consequently, any program core $\mathcal{P}_\mathcal{C}$ of an ELP $\mathcal{P}$ with conflicting rules can be transformed to a uniformly non-contradictory core by successively extending conflicting rules with suitable λ-extensions.

It may happen that a rule r does not have λ-extensions that resolves all of its conflicts at once. In those cases the presented approach should be applied to a subset of the conflicts of r or to a rule $s \in Adv(r)$. Similarly, given multiple rules r with head literal a that are in conflict with multiple rules s with head $\overline{a}$, λ-extensions can be computed by processing one rule r after another. A program $\mathcal{P}$ with constraints $\leftarrow a.$ can be handled by rewriting them to $x \leftarrow a, {\sim}x.$ beforehand, where x is an atom that does not appear in $\mathcal{P}$.

3 Outlook and Summary

The presented approach constitutes crucial groundwork to allow program encodings to be used with any (valid) input data. A uniformly non-contradictory program core guarantees that no contradictions can arise and we have shown how any program core can be modified to a uniformly non-contradictory core by using appropriate λ-extensions. Using informative λ-extensions we furthermore present modifications that are based on the actual knowledge that is contained in the program core and not purely technical devices to prevent contradictions. In general, there can exist a vast amount of λ-extensions for a conflicting rule r[1]. We therefore propose that this approach is used in a larger framework where the knowledge expert is able to interactively resolve each conflict by choosing the most suitable λ-extensions for r. In [5], we show how an order over the conflicts of a program and their possible solutions can be computed to ensure an efficient conflict resolution process. This results in a modified and uniformly non-contradictory program core that contains knowledge that is professionally adequate and consistent.

[1] A detailed method on how to compute all informative extensions for a conflicting rule can be found in [4].

References

1. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2012). https://doi.org/10.2200/S00457ED1V01Y201211AIM019
2. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Gener. Comput. **9**(3/4), 365–386 (1991). https://doi.org/10.1007/BF03037169, https://doi.org/10.1007/BF03037169
3. Thevapalan, A., Kern-Isberner, G.: Towards interactive conflict resolution in ASP programs. In: Martínez, M.V., Varzinczak, I. (eds.) 18th International Workshop on Non-Monotonic Reasoning, Workshop Notes. pp. 29–36 (September 2020)
4. Thevapalan, A., Kern-Isberner, G.: On establishing robust consistency in answer set programs. Theory and Practice of Logic Programming p. 1–34 (2022). https://doi.org/10.1017/S1471068422000357
5. Thevapalan, A., Kern-Isberner, G.: Sorting strategies for interactive conflict resolution in ASP. CoRR **abs/2308.15889** (2023). https://doi.org/10.48550/arXiv.2308.15889, https://doi.org/10.48550/arXiv.2308.15889