Łukaszewicz-style Answer Set Programming: A Preliminary Report

J. P. Delgrande[1], M. Gharib[2], R. E. Mercer[3], V. Risch[4], and T. Schaub[2⋆]

[1] School of Computing Science, Simon Fraser University , Burnaby, B.C., Canada V5A 1S6
[2] Institut für Informatik, Universität Potsdam, D–14415 Potsdam, Germany
[3] Cognitive Engineering Laboratory, Department of Computer Science,
The University of Western Ontario, London, Ontario, Canada N6A 5B7
[4] InCA Team, LSIS – UMR CNRS 6168,
Domaine Universitaire de Saint-Jérôme, avenue Escadrille Normandie Niemen,
F–13397 Marseille cédex 20, France

Abstract. The correspondence between Reiter's default reasoning and logic programming has been exhaustively studied (e.g. [1], [2], [3]). *A Contrario* the relation with the many variants of the initial theory of Reiter seems far less known. This paper aims to present a preliminary investigation on applying a variant of default reasoning proposed by Witold Łukaszewicz [5] to extended logic programs. We show that the modification made to the notion of extension by Łukaszewicz has its counterpart as a relaxed notion of answer set of an extended logic program. As can be expected from this correspondence: (1) any extended logic program has always at least one relaxed answer set; (2) classical answer sets can be completely characterized among the set of relaxed answer sets of an extended logic program.

Keywords: Logic Programming, Default Logic, justified extensions, answer sets

1 Introduction

This paper aims to present a preliminary investigation on applying Łukaszewicz approach of default reasoning to extended logic programming. Following Gelfond and Lifschitz [3] who established a one-to-one correspondence between the classical extensions of Reiter's default reasoning and the answer sets of an extended logic program, we show that justified extensions have their exact counterpart as a notion of relaxed answer set. However, we adopt a different approach than the one chosen in [3] since, instead of working at the level of fixed-points, we rather try to establish a correspondence between the set of generating defaults of a justified extension and a relaxed answer set of an extended logic program. In the first section below we briefly recall some basic features of extended logic programming and default reasoning. An alternative characterization of justified extensions is also recalled. The second section defines the notion of relaxed answer set and establish a correspondence with justified extensions.

⋆ Affiliated with the School of Computing Science at Simon Fraser University, Burnaby, Canada.

2 Preliminaries

We consider that the fundamentals of both default reasoning and logic programming are familiar and we only briefly give a short reminder in order to fix our notations. The reader is referred to the basic sources on the subject [8] [4] for a complete introduction.

Throughout this paper, we consider only a restricted form of propositional default theories (W, D) where (1) the language has no disjunction except in the special case of horn clauses in W, (2) W contains no conjunctions, (3) the consequence and each justification is an atom, (4) W is emptied by the following transformation: facts are transformed into prerequisite-free, justification-free rules (or in the sense of logic programs, bodiless rules) and horn clauses are transformed into justification-free rules (or rules without 'not's in the logic program sense. This leads us to use the language of atoms with classical negation for default theories and extended logic programs. Hence, we abuse the use of the generic term "atom" to include atoms with classical negation, considering that the context will make clear what is meant. The logical closure of a set of formulas S is denoted by $Th(S)$. We also use the following notions (most of them developed in [6] and [7]): Atoms and not-atoms (formulas of the form not(a)) are called *literals*. For a set of literals S, by S^+ (resp. S^-) we denote the set of atoms (resp. not-atoms) in S. Moreover, we consider $|S| = \{a \mid \text{not}(a) \in S\}$ and $\neg S = \{\neg a \mid a \in S^+\}$. Hence, logic programs consist of rules of the form

$$head \leftarrow body^+, body^-$$

corresponding modularly to default rules of the form

$$\frac{\bigwedge body^+ : \neg\, |body^-|}{head}$$

where *head* is an atom and *body* is a set of literals (cf. [3]).

Given a logic program Π, this allows us to denote the corresponding default theory by Π as well since it should be clear from the context what we mean. Given a rule $d \in \Pi$, by $head(d), body^+(d), body^-(d)$ we denote respectively the corresponding parts of the rule d.

The most important objects regarding an extended logic program are the answer sets generated by this program. Consider Π_g, the set of ground instances of the logic program Π. The *reduct* Π_g^S with respect to a set S of atoms S is obtained from Π_g by first deleting each rule that has not(a) in its body with $a \in S$, then deleting all not(l) in the remaining rules. Inspiring ourselves from Niemelä and Simons [7], we summarize these notions as follows:

Definition 1. *Consider Π an extended ground logic program, and L a set of literals.*

- *The* reduction of Π with respect to L *is the set:*

$$R(\Pi, L) = \{head \leftarrow body^+ \mid head \leftarrow body^+, body^- \in \Pi, body^- \subseteq L^-\}.$$

- *The* deductive closure *[7] of a set of ground rules Π of a logic program and a set of literals L, denoted $Dcl(\Pi, L)$, is the smallest set of atoms which contains L^+ and is closed under the inference rules of the reduction $R(\Pi, L)$.*
- *The* reduct *with respect to a set S of atoms, written Π^S is the reduction of Π with respect to* not($\overline{S}$) *where $\overline{S}$ denotes the complement of S, i.e. the atoms not in S. In other words, $\Pi^S = R(\Pi, \text{not}(\overline{S}))$.*
- *A set of atoms S is an* answer set *for Π iff $S = Dcl(\Pi, \text{not}(\overline{S}))$.*

Classical extension is the notion on the side of default theories corresponding to answer set. *PREREQ*(Π), *JUST*(Π) and *CONS*(Π) are respectively the sets of all pre-requisites, justifications and consequents that come from defaults in a set Π, that is respectively all the $\bigwedge body^+$, $\neg \mid body^- \mid$, and *head* in Π. An extension is then usually defined as a smallest fixed point of a set of formulas. It contains W (but remember that we only consider here default theories with an empty W), is logically closed, and the defaults whose consequents belong to the extension verify a property which actually allows them to be used. The manner in which this property is considered is related to the variant of default reasoning under consideration. It is know that regarding Reiter's approach of default reasoning, with *Th* denoting the logical closure, for any program Π, any set S of atoms, S is an answer set for Π iff $Th(S)$ is a classical extension of Π (see [3]). We will show that a similar result holds with justified extensions, such as defined by Łukaszewicz in [5]. So let us first give a brief account on justified extensions.

In his original paper, Łukaszewicz gives a fixed-point definition of the notion of justified extension, similar to Reiter. However this definition is quite complex, involving two fixed points and a heavy theoretical framework. In what follows, we move directly to a restricted form of the characterization given in [9]. Consider first the following notion of *S-groundedness* as a restricted case (empty W) of the definition given by Schwind[1] [10]: A set Π of defaults is *S-grounded* iff for all $d \in \Pi$ there is a finite sequence $d_0, \ldots, d_k$ of elements of Π such that (1) *PREREQ*($\{d_0\}$) is an empty body rule of Π, (2) for $1 \leq i \leq k-1$, *PREREQ*($\{d_{i+1}\}$) $\in Th(CONS(\{d_0, \ldots, d_i\}))$, and $d_k = d$. The restricted form (W empty) of the characterization given in [9] is then:

[1] Note that the term used by Schwind is *grounded*. We use here the term *S-grounded* in order to avoid any confusion with the usual notion of *ground program*.

Theorem 1. *Let Π be an extended logic program. E is a justified extension with respect to F for Π iff there is Π' a maximal S-grounded subset of Π such that $E = Th(CONS(\Pi'))$, $F = JUST(\Pi')$, and for each default $d \in \Pi$, of the form $\frac{\bigwedge body^+ : \neg |body^-|}{head}$:*

(i) If $d \in \Pi'$ then ($\bigwedge body^+ \in E$ and for each $b \in |body^-|, b \notin E$).

Remark 1. The difference between Reiter's and Łukaszewicz's approaches holds in *(i)*: if the *if* condition is changed to an *iff* one gets the definition of Reiter extensions. Another way to stress this is to take into account the behavior of the defaults that do not participate in the construction of an extension: in Reiter's approach, these defaults must verify an additional condition that indeed allows them not to participate in the extension being constructed (this results from the contrapositive form of the necessary condition in the *iff*). In other words, according to Łukaszewicz, we should never be allowed to revise a justification already used for deriving the consequent of a default (contrary to Reiter's approach). As a consequence of this:

- A default theory always has a justified extension whereas it may have no classical extension.
- Every classical extension (if any) is a justified extension.

Note that Π' is the set of *generating defaults* of E, also written $GD(\Pi, E)$.

3 Relaxing answer sets

Let us now relax the notion of answer set:

Definition 2. *S is a* relaxed answer set *of an extended logic program Π iff $S \subseteq_{Max} Dcl(\Pi, \text{not}(\overline{S}))$, i.e. S is a maximal subset of the deductive closure of Π and* $\text{not}(\overline{S})$.

Remark 2. We get immediately that answer sets are relaxed answer sets for which the extra condition $Dcl(\Pi, \text{not}(\overline{S})) \subseteq S$ holds.

We show now that, just as classical extensions are in one-to-one correspondence with answer sets (cf. [3]), relaxed answer sets are in bijection with justified extensions. Unlike in [3], and instead of using the fixed-point definition of an extension, we relate the set of generating defaults of a justified extension with the reduct used to produce the relaxed answer set of a program. We make use the following lemmas:

Lemma 1. *If S is a relaxed answer set of an extended logic program then $Th(S) \cap Lit = S$.*

Proof. Follows the guideline of the proof of the similar theorem given in [3]. Let S be a relaxed answer set of an extended logic program Π. If Π is contradictory then $S = Lit$ (Proposition 1 of [3]), and consequently $Th(S) \cap Lit = Lit$. If not, then S is a subset of a consistent set of ground literals, i.e. $S \subseteq_{\text{Max}} Dcl(\Pi, \text{not}(\overline{S}))$, so that the ground literals that logically follows from S are precisely the elements of S.

Lemma 2. *For any extended logic program Π, any $d \in \Pi$,*

(i) Let S be a relaxed answer set of Π, then:

$$d \in \Pi^S \Rightarrow \textit{for each } \mathrm{not}(b) \in \textit{body}^-(d), b \notin \textit{Th}(S)$$

(ii) Let E be a justified extension of Π, then:

$$d \in GD(\Pi, E) \Rightarrow d \in \Pi^{E \cap Lit}$$

Proof. *(i)* Assume $d \in \Pi^S$ i.e. $d \in R(\Pi, \mathrm{not}(\overline{S}))$, that is $body^- \subseteq (\mathrm{not}(\overline{S}))^-$ from Definition 1, i.e. $body^- \subseteq \mathrm{not}(\overline{S})$ (since $(\mathrm{not}(\overline{S}))^- = \mathrm{not}(\overline{S})$), that is for each $b \in \textit{body}^-(d), b \in \overline{S}$, hence $b \notin S$. Assume now that there exists $b \in \textit{body}^-(d), b \in \textit{Th}(S)$, that is $b \in S$ (2) (since $b \in \textit{Lit}$ and from Lemma 1). This is a contradiction.
(ii) Assume $d \in GD(\Pi, E)$, i.e. $\bigwedge \textit{body}^+ \in E$ and for each $b \in |\textit{body}^-|, b \notin E$ (from Theorem 1). Assume $d \notin \Pi^{E \cap Lit}$, that is $d \notin R(\Pi, \mathrm{not}(\overline{E \cap \textit{Lit}}))$. From Definition 1 we get $body^-(d) \not\subseteq (\mathrm{not}(\overline{E \cap \textit{Lit}}))^-$ that is, for each $\mathrm{not}(b) \in \textit{body}^-(d), \mathrm{not}(b) \notin \mathrm{not}(\overline{E \cap \textit{Lit}})$. Hence $\mathrm{not}(b) \in \mathrm{not}(E \cap \textit{Lit})$ i.e. $b \in (E \cap \textit{Lit})$, which contradicts the assumption $b \notin E$.

Theorem 2. *For any extended program Π,*

(i) If S is a relaxed answer set of Π then the logical closure of S is a justified extension.
(ii) Every justified extension of Π is the logical closure of exactly one justified answer set of Π.

Proof. *(i)* Consider S, a relaxed answer set of Π, that is $S \subseteq_{\mathrm{Max}} \textit{Dcl}(\Pi, \mathrm{not}(\overline{S}))$. For any $d \in R(\Pi, \mathrm{not}(\overline{S}))(= \Pi^S)$, if $\textit{body}^+(d) \in S$ then $\textit{body}^+(d) \in \textit{Th}(S)$ and $\textit{head}(d) \in \textit{Dcl}(\Pi, \mathrm{not}(\overline{S}))$. Since by Lemma 2 *(i)*, for each $\mathrm{not}()b) \in \textit{body}^-(d), b \notin \textit{Th}(S)$, and since S is maximal, then $d \in GD(\textit{Th}(S), \Pi)$ hence $\textit{head}(d) \in \textit{Th}(S)$ i.e. $\textit{Th}(S)$ is a justified extension of Π.
(ii) We have to show that $E \cap \textit{Lit} \subseteq_{\mathrm{Max}} \textit{Dcl}(\Pi, \mathrm{not}(\overline{E \cap \textit{Lit}}))$. Assume there exists $b \in E \cap \textit{Lit}$ such that $b \notin \textit{Dcl}(\Pi, \mathrm{not}(\overline{E \cap \textit{Lit}}))$. b is the head of some rule of d of Π, i.e. there exists $d : b \leftarrow \textit{body}^+, \textit{body}^-$ such that $d \in GD(E, \Pi)$ and $d \notin R(\Pi, \mathrm{not}(\overline{E \cap \textit{Lit}}))$ which contradicts Lemma 2 *(ii)*, hence $E \cap \textit{Lit} \subseteq \textit{Dcl}(\Pi, \mathrm{not}(\overline{E \cap \textit{Lit}}))$. $E \cap \textit{Lit}$ is maximal because $E = \textit{Th}(\textit{CONS}(\Pi'))$ is maximal with Π' maximal in Π. It remains to show that, for any relaxed answer set S, $\textit{Th}(S) = E$ only if $S = E \cap \textit{Lit}$. By Lemma 1 $E \cap \textit{Lit} = \textit{Th}(S) \cap \textit{Lit} = S$.

Example 1. Consider the extended logic program Π composed of the "pathological" rule:

$$a \leftarrow \mathrm{not}(a)$$

- Π has no answer set, neither has it a classical extension.
- Π has $\emptyset$ as only relaxed answer set, which corresponds to $\textit{Th}(\emptyset)$ as only justified extension.

Example 2. Consider the extended logic program Π:

$$a \leftarrow \mathrm{not}(\neg b)$$
$$b \leftarrow \mathrm{not}(a)$$

- $\{a\}$ is an answer set of Π, but $\{b\}$ is not.$\textit{Th}(a)$ is the only classical extension of Π.
- $\{a\}$ and $\{b\}$ are both relaxed answer sets of Π. $\textit{Th}(a)$, $\textit{Th}(b)$ are justified extensions of Π.

4 Refinements

In classical answer set programming, problems are usually formulated by decomposition into a generation and a test part. While the generation part is arguably possible by means of relaxed answer sets as well, this does not transfer directly to the test part, which normally relies on integrity constraints for eliminating invalid candidate answer sets. An integrity constraint is a headfree rule of the form

$$\leftarrow \mathit{body}^+, \mathit{body}^- \ .$$

In classical answer set programming such a rule can be encoded as

$$x \leftarrow \mathit{body}^+, \mathit{body}^-, \mathit{not}(x) \ ,$$

where x is a new symbol. So, informally, whenever body^+ and body^- are satisfied the putative answer set at hand is destroyed. Such an encoding is inappropriate in our setting since the resulting rules are always inapplicable and can thus never destroy a putative relaxed answer set.

Finally, because there may be many more (even exponentially many) relaxed answer sets than classical ones, the need for a formal means of elimination is even more acute in our setting than in the classical one.

For addressing this, we propose to encode the above integrity constraint as

$$x \leftarrow \mathit{body}^+, \mathit{body}^- \ ,$$

where x is a new symbol, not occuring in the underlying program. We say that a relaxed answer set S of some program Π satisfies the integrity constraints included in Π, if S does not contain the special symbol x. For directly characterising relaxed answer sets that satisfy all integrity constraint, one simply has to enforce the exclusion of x in the maximisation described in Definition 2.

In fact, without integrity constraints, the approach is monotonic in the sense that the addition of rules never eliminates any existing relaxed answer sets. To see this, consider the following variant of Example 2:

$$\begin{aligned} a &\leftarrow \\ b &\leftarrow \mathrm{not}(a) \end{aligned}$$

This program has two relaxed answer sets, $\{a\}$ and $\{b\}$, which is arguably counterintuitive, because the monotonic inference of a does not override the nonmonotonic inference of b to eliminate the second relaxed answer set $\{b\}$.

In analogy to the above, we propose to address this by incorporating an additional condition into the maximisation described in Definition 2. To be more precice, we require that any relaxed answer set S contains $\mathit{Dcl}(\Pi, \emptyset)$, the set of monotonic consequences of Π. In this way, we eliminate all relaxed answer sets that override monotonic consequences. Alternatively, we may even stipulate that any relaxed answer set contains the set of well-founded consequences of Π. In both cases, we obtain only the relaxed answer set $\{a\}$ from the above program.

A prototypical implementation in Java exists and can be obtained upon request from the authors.

5 Conclusion

The notion of justified extension comes from knowledge representation ; it has interesting properties: justified extensions always exist, and classical extensions are a special case easily characterized among justified extensions. From Theorem 2 these properties hold also for relaxed answer sets. Another interesting characteristic of justified extensions is that they are easier to compute than classical extensions. A question to be further investigated is whether this is still true with relaxed answer sets compared to answer sets, and how the known methods could be adapted for computing relaxed answer sets.

Acknowledgements

The first and third authors were partially funded by NSERC (Canada). The second and fifth authors were partially funded by the IST programme of the EU under project IST-2001-37004 WASP. The fifth author was partially supported by the German Science Foundation (DFG) under grant SCHA 550/6, TP C.

References

1. N. Bidoît and C. Froidevaux. Minimalism subsumes default logic and circumscription. In *Proceedings of LICS–87*, pages 89–97, 1987.
2. N. Bidoît and C. Froidevaux. Negation by default and nonstratifiable logic programs. Technical Report 437, Université Paris XI, 1987.
3. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
4. V. Lifschitz. Foundations of logic programming. In Gerhard Brewka, editor, *Principle of knowledge representation*, pages 69–127. The University of Chicago Press, 1996.
5. W. Lukaszewicz. Considerations on default logic — an alternative approach. *Computational Intelligence*, 4:1–16, 1988.
6. I. Niemelä. Towards efficient default reasoning. In *International Joint Conference on Artificial Intelligence, IJCAI'95*, pages 312–318, 1995.
7. I. Niemelä and P. Simons. Smodels — an implementation of the stable model and the well founded semantics for normal logic programs. In *4th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR'97*, pages 420–429, July 1997.
8. R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.
9. V. Risch. Analytic tableaux for default logics. *Journal of Applied Non-Classical Logics*, 6:71–88, 1996.
10. C. Schwind. A tableau-based theorem prover for a decidable subset of default logic. In *10th International Conference on Automated Deduction, CADE'10*, pages 541–546, 1990.