Translating P-log, LP^{MLN}, LPOD, and CR-Prolog2 into Standard Answer Set Programs

Zhun Yang
School of Computing, Informatics, and Decision Systems Engineering, Arizona State University
[Arizona State University, P.O. Box 878809, Tempe, AZ 85287, United States]
zyang90@asu.edu

Abstract

Answer set programming (ASP) is a particularly useful approach for nonmonotonic reasoning in knowledge representation. In order to handle quantitative and qualitative reasoning, a number of different extensions of ASP have been invented, such as quantitative extensions LP^{MLN} and P-log, and qualitative extensions LPOD, and CR-Prolog$_2$.

Although each of these formalisms introduced some new and unique concepts, we present reductions of each of these languages into the standard ASP language, which not only gives us an alternative insight into the semantics of these extensions in terms of the standard ASP language, but also shows that the standard ASP is capable of representing quantitative uncertainty and qualitative uncertainty. What's more, our translations yield a way to tune the semantics of LPOD and CR-Prolog$_2$. Since the semantics of each formalism is represented in ASP rules, we can modify their semantics by modifying the corresponding ASP rules.

For future work, we plan to create a new formalism that is capable of representing quantitative and qualitative uncertainty at the same time. Since LPOD rules are simple and informative, we will first try to include quantitative preference into LPOD by adding the concept of weight and tune the semantics of LPOD by modifying the translated standard ASP rules.

2012 ACM Subject Classification Knowledge representation and reasoning

Keywords and phrases answer set programming, preference, LPOD, CR-Prolog

Digital Object Identifier 10.4230/OASIcs.ICLP.2018.17

Acknowledgements This work was partially supported by the National Science Foundation under IIS-1526301.

1 Introduction and Problem Description

In answer set programming, each answer set encodes a solution to the problem that is being modeled. There is often a need to express how likely a solution is, so several extensions of answer set programs, such as LP^{MLN} [19] and P-log [7], were made to express a quantitative uncertainty for each answer set. LP^{MLN} extends answer set programs by adopting the log-linear weight scheme of Markov Logic. P-log is a probabilistic extension of ASP with sophisticated semantics. Similarly, since there is often a need to express that one solution is preferable to another, several extensions of answer set programs, such as Logic Programs with Ordered Disjunction (LPOD) [8], CR-Prolog [5], and CR-Prolog$_2$ [6], were made to express a qualitative preference over answer sets. In LPOD, the qualitative preference is introduced by the construct of ordered disjunction in the head of a rule: $A \times B \leftarrow Body$ intuitively means, when $Body$ is true, if possible then A, but if A is not possible, then at least B. CR-Prolog$_2$ also has order rules as in LPOD, and it introduces consistency-restoring rules – rules that can be added only when they can make an inconsistent program consistent.

Technical Communications of the 34th International Conference on Logic Programming (ICLP 2018).
Editors: Alessandro Dal Palu', Paul Tarau, Neda Saeedloei, and Paul Fodor; Article No. 17; pp. 17:1–17:10
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

It remains an open question whether these formalisms can be reduced back to standard answer set programs. In other words, whether ASP is expressive enough to express the semantics of all these extensions? There were few attentions to this question where no positive answer had been proposed. Lee et al. [19] showed that a subset of P-log can be represented by LP^{MLN}, which is very similar to ASP except the introducing of weight for each rule. However, the feature of dynamic probability assignment in P-log is not preserved, and the reduction from LP^{MLN} to ASP was still unclear. Proposition 2 from [8] states that there is no reduction of LPOD to disjunctive logic programs [17] based on the fact that the answer sets of disjunctive logic programs are subset-minimal whereas LPOD answer sets are not necessarily so. However, this justification is limited to translations that preserve the underlying signature. Indeed, our paper "LP^{MLN}, Weak Constraints, and P-log" [20] and our ICLP paper that is being evaluated provides a positive answer to this question.

We present a reduction of P-log to LP^{MLN} and a reduction of LP^{MLN} to answer set programs with weak constraints. These translations show how the weights in the weak constraints can be used to denote quantitative uncertainty and, further, to represent probabilities. We also present a reduction of LPOD and CR-Prolog$_2$ to standard answer set programs by compiling away ordered disjunctions and consistency-restoring rules. These translations show how qualitative uncertainty is handled by the "definition" rules in ASP.

Since our research shows that ASP is capable of representing quantitative and qualitative uncertainty, it naturally follows a question that: can we combine quantitative uncertainty and qualitative preference in a single formalism? We are looking forward to answering this question in our future work.

The paper will give a summary of my research, including some background knowledge and reviews of existing literature (Section 2), goal of my research (Section 3), the current status of my research (Section 4), the preliminary results we accomplished (Section 5), and some open issues and expected achievements (Section 6).

2 Background and Overview of the Existing Literature

We only review the syntax and semantics of LP^{MLN} and LPOD. Please refer to [7] and [6] for the syntax and semantics of P-log and CR-Prolog$_2$, whose semantics are all based on a long translation to answer set programs.

2.1 Review: LP^{MLN}

We review the definition of LP^{MLN} from [19]. In fact, we consider a more general syntax of programs than the one from [19], but this is not an essential extension. We follow the view of [15] by identifying logic program rules as a special case of first-order formulas under the stable model semantics. For example, rule $r(x) \leftarrow p(x), not\ q(x)$ is identified with formula $\forall x(p(x) \wedge \neg q(x) \rightarrow r(x))$. An LP^{MLN} program is a finite set of weighted first-order formulas $w : F$ where w is a real number (in which case the weighted formula is called *soft*) or α for denoting the infinite weight (in which case it is called *hard*). An LP^{MLN} program is called *ground* if its formulas contain no variables. We assume a finite Herbrand Universe. Any LP^{MLN} program can be turned into a ground program by replacing the quantifiers with multiple conjunctions and disjunctions over the Herbrand Universe. Each of the ground instances of a formula with free variables receives the same weight as the original formula.

For any ground LP^{MLN} program Π and any interpretation I, $\overline{\Pi}$ denotes the unweighted formula obtained from Π, and Π_I denotes the set of $w : F$ in Π such that $I \models F$, and SM[Π] denotes the set $\{I \mid I$ is a stable model of $\overline{\Pi_I}\}$ (We refer the reader to the stable model semantics of first-order formulas in [15]). The *unnormalized weight* of an interpretation I under Π is defined as LP^{MLN}

$$W_\Pi(I) = \begin{cases} exp\left(\sum\limits_{w:F\ \in\ \Pi_I} w\right) & \text{if } I \in \text{SM}[\Pi]; \\ 0 & \text{otherwise.} \end{cases}$$

The *normalized weight* (a.k.a. *probability*) of an interpretation I under Π is defined as

$$P_\Pi(I) = \lim_{\alpha\to\infty} \frac{W_\Pi(I)}{\sum\limits_{J\in \text{SM}[\Pi]} W_\Pi(J)}.$$

I is called a *(probabilistic) stable model* of Π if $P_\Pi(I) \neq 0$.

2.2 Review LPOD

We review the definition of LPOD from [8], which assumes propositional programs.

Syntax: A (propositional) LPOD Π is $\Pi_{reg} \cup \Pi_{od}$, where its *regular part* Π_{reg} consists of usual ASP rules $Head \leftarrow Body$, and its *ordered disjunction part* Π_{od} consists of *LPOD rules* of the form

$$C^1 \times \cdots \times C^n \leftarrow Body \tag{1}$$

in which C^i are atoms, n is at least 2, and $Body$ is a conjunction of atoms possibly preceded by *not*.[1] Rule (1) says "when $Body$ is true, if possible then C^1; if C^1 is not possible then C^2; $\dots$; if all of $C^1, \dots, C^{n-1}$ are not possible then C^n".

Semantics: For an LPOD rule (1), its *i-th option*, where $i \in \{1, \dots, n\}$, is defined as $C^i \leftarrow Body, not\ C^1, \dots, not\ C^{i-1}$.

Let Π be an LPOD. A *split program* of Π is obtained from Π by replacing each rule in Π_{od} by one of its options. A set S of atoms is a *candidate answer set* of Π if it is an answer set of a split program of Π. A split program of Π may be inconsistent (i.e., may not necessarily have an answer set).

A candidate answer set S of Π is said to *satisfy* rule (1)

- to degree 1 if S does not satisfy $Body$;
- to degree j ($1 \leq j \leq n$) if S satisfies $Body$ and $j = min\{k \mid C^k \in S\}$.

For a set S of atoms, let $S^i(\Pi)$ denote the set of rules in Π_{od} satisfied by S to degree i. For candidate answer sets S_1 and S_2 of Π, [9] introduces the following four preference criteria.

1. **Cardinality-Preferred:** S_1 is *cardinality-preferred* to S_2 ($S_1 >^c S_2$) if there is a positive integer i such that $|S_1^i(\Pi)| > |S_2^i(\Pi)|$, and $|S_1^j(\Pi)| = |S_2^j(\Pi)|$ for all $j < i$.
2. **Inclusion-Preferred:** S_1 is *inclusion-preferred* to S_2 ($S_1 >^i S_2$) if there is a positive integer i such that $S_2^i(\Pi) \subset S_1^i(\Pi)$, and $S_1^j(\Pi) = S_2^j(\Pi)$ for all $j < i$.
3. **Pareto-Preferred:** S_1 is *pareto-preferred* to S_2 ($S_1 >^p S_2$) if there is a rule that is satisfied to a lower degree in S_1 than in S_2, and there is no rule that is satisfied to a lower degree in S_2 than in S_1.

[1] In [8], a usual ASP rule is viewed as a special case of a rule with ordered disjunction when $n = 1$ but in this paper, we distinguish them. This simplifies the presentation of the translation and also allows us to consider LPOD programs that are more general than the original definition by allowing modern ASP constructs such as aggregates.

4. **Penalty-Sum-Preferred:** S_1 is *penalty-sum-preferred* to S_2 ($S_1 >^{ps} S_2$) if the sum of the satisfaction degrees of all rules is smaller in S_1 than in S_2.

A set S of atoms is a *k-preferred* ($k \in \{c, i, p, ps\}$) *answer set* of an LPOD Π if S is a candidate answer set of Π and there is no candidate answer set S' of Π such that $S' >^k S$.

2.3 Existing Literature

There are quite a lot of formalisms made to represent quantitative uncertainty.

LP^{MLN} [19] is a probabilistic logic programming language that extends answer set programs [16] with the concept of weighted rules, whose weight scheme is adopted from that of Markov Logic [23], a probablistic extension of SAT. It is shown in [19, 18] that LP^{MLN} is expressive enough to embed Markov Logic and several other probabilistic logic languages, such as ProbLog [13], Pearls' Causal Models [22], and a fragment of P-log [7]. On the other hand, [2] proposed an embedding from LP^{MLN} into P-log.

Another famous quantitative extension of ASP are weak constraints [12], which are to assign a quantitative preference over the stable models of non-weak constraint rules: weak constraints cannot be used for deriving stable models.

Many formalisms are made to represent qualitative uncertainty. Most of them are extensions of ASP, where their semantics or implementations are also based on answer set programs.

In [11], LPOD is implemented using SMODELS. The implementation interleaves the execution of two programs–a generator which produces candidate answer sets and a tester which checks whether a given candidate answer set is maximally preferred or produces a more preferred candidate if it is not. An implementation of CR-Prolog reported in [3] uses a similar algorithm.

[14] finds the "order preserving answer sets" of an ordered logic program (where a strict partial order is assigned among some rules) by meta-programming. Our translations are similar to the meta-programming approach to handle preference in ASP in that we turn LPOD and CR-Prolog$_2$ into answer set programs that do not have the built-in notion of preference.

In contrast, the reductions shown in this paper can be computed by calling an answer set solver one time without the need for iterating the generator and the tester. This feature may be useful for debugging LPOD and CR-Prolog$_2$ programs because it allows us to compare all candidate and preferred answer sets globally.

Asprin [10] provides a flexible way to express various preference relations over answer sets and is implemented in CLINGO. Similar to the existing LPOD solvers, CLINGO makes iterative calls to find preferred answer sets, unlike the one-shot execution as we do.

In [1], Asuncion *et al.* extended propositional LPODs to the first order case, where the candidate answer sets of a first order LPOD can be obtained by finding the models of a second-order formula.

3 Goal of the Research

The following are our research objectives.

- **Find a translation** *plog2asp* **from P-log to answer set programs** We design a one-time translation *plog2asp* such that for any P-log Π, the answer sets of the answer set program *plog2asp*(Π) agree with (i.e., their explanation to the domain are the same) the possible worlds of Π.

- **Find a translation** $lpmln2asp$ **from** LP^{MLN} **to answer set programs** We design a one-time translation $lpmln2asp$ such that for any LP^{MLN} program Π, the answer sets of the answer set program $lpmln2asp(\Pi)$ agree with the probabilistic answer sets of Π.
- **Analyze how quantitative uncertainty can be expressed in standard answer set programs** We compare the two translations $plog2asp$ and $lpmln2asp$, and analyze how quantitative uncertainty represented by weight (in LP^{MLN}) and sophisticated probability assignment (in P-log) can be expressed in standard answer set programs.
- **Find a translation** $lpod2asp$ **from LPOD to answer set programs** We design a one-time translation $lpod2asp$ such that for any LPOD Π, the optimal answer sets of the answer set program lpod2asp(Π) "report" all the candidate answer sets of Π in different name spaces and whether each of them is a preferred answer set.
- **Find a translation** $crpt2asp$ **from CR-Prolog$_2$ to answer set programs** We design a one-time translation $crpt2asp$ such that for any CR-Prolog$_2$ program Π, the optimal answer sets of the answer set program $crpt2asp(\Pi)$ "report" all the generalized answer sets of Π in different name spaces and whether each of them is also a candidate answer sets or a preferred answer sets.
- **Analyze how qualitative uncertainty can be expressed in standard answer set programs** We compare the two translations $lpod2asp$ and $crpt2asp$, and analyze how qualitative preference represented by ordered disjunction and consistency-restoring rules can be expressed in standard answer set programs.
- **Design a single formalism to represent both quantitative and qualitative uncertainty** We design a new formalism that can be used to represent quantitative and qualitative uncertainty at the same time. The semantics of the new formalism is defined as a reduction to standard answer set programs as we did for those four formalisms.

4 Current Status of the Research

This research is at a middle phase.

The first 2 bullets of our goals are done in our paper accepted by AAAI 2017 [20], where we proposed a translation $plog2lpmln$ from P-log to LP^{MLN}, and a translation $lpmln2wc$ from LP^{MLN} to answer set programs with weak constraints. The translations $lpod2asp$ and $crpt2asp$ are also completed in our paper accepted by ICLP 2018 [21]. We also compared all these four translations and have some ideas about how standard answer set programs handle quantitative and qualitative uncertainty.

Currently, we are testing our ideas by introducing quantitative uncertainty into LPOD. The experiments are based on our reduction from LPOD to answer set programs. We are tuning the semantics of LPOD by modifying on the translated rules.

5 Preliminary Results Accomplished

In this section, we will present our main theorems, along with some examples to illustrate how our translations work.

5.1 From LP^{MLN} to Answer Set Programs

▶ **Theorem 1.** *(from [20]) For any LP^{MLN} program Π, the most probable stable models (i.e., the stable models with the highest probability) of Π are precisely the optimal stable models of the program with weak constraints* lpmln2wc(Π).

▶ **Example 2.** Consider the LP^{MLN} program Π_1 in Example 1 from [19].

$$\begin{array}{lll} \alpha: & Bird(Jo) \leftarrow ResidentBird(Jo) & (r1) \\ \alpha: & Bird(Jo) \leftarrow MigratoryBird(Jo) & (r2) \\ \alpha: & \bot \leftarrow ResidentBird(Jo), MigratoryBird(Jo) & (r3) \\ 2: & ResidentBird(Jo) & (r4) \\ 1: & MigratoryBird(Jo) & (r5) \end{array}$$

The (simplified) translation $\mathsf{lpmln2wc}(\Pi_1)$ is as follows, which simply removes α from each hard rule and turns each soft rule into a choice rule and a weak constraint.

$$\begin{array}{ll} Bird(Jo) \leftarrow ResidentBird(Jo) & \\ Bird(Jo) \leftarrow MigratoryBird(Jo) & \\ \bot \leftarrow ResidentBird(Jo), MigratoryBird(Jo) & \\ \{ResidentBird(Jo)\}^{\text{ch}} & \\ \{MigratoryBird(Jo)\}^{\text{ch}} & \\ :\sim ResidentBird(Jo) & [-2@0] \\ :\sim MigratoryBird(Jo) & [-1@0] \end{array}$$

There are three probabilistic stable models of Π_1: $\emptyset$, $\{Bird(Jo), ResidentBird(Jo)\}$, and $\{Bird(Jo), MigratoryBird(Jo)\}$. Among them, $\{Bird(Jo), ResidentBird(Jo)\}$ is the most probable stable model of Π_1 since it is associated with a highest weight. It is also an optimal stable model of $\mathsf{lpmln2wc}(\Pi_1)$ since it has the least penalty -2 at level 0.

5.2 From P-log to LP^{MLN}

▶ **Theorem 3.** *(from [20]) Let Π be a consistent P-log program. There is a 1-1 correspondence ϕ between the set of the possible worlds of Π with non-zero probabilities and the set of probabilistic stable models of* $\mathsf{plog2lpmln}(\Pi)$.

▶ **Example 4.** Consider a variant of the Monty Hall Problem encoding in P-log from [7] to illustrate the probabilistic nonmonotonicity in the presence of assigned probabilities. There are four doors, behind which are three goats and one car. The guest picks door 1, and Monty, the show host who always opens one of the doors with a goat, opens door 2. Further, while the guest and Monty are unaware, the statistics is that in the past, with 30% chance the prize was behind door 1, and with 20% chance, the prize was behind door 3. Is it still better to switch to another door? This example can be formalized in P-log program Π_2, using both assigned probability and default probability, as

$$\begin{array}{l} \sim CanOpen(d) \leftarrow Selected = d. \quad (d \in \{1,2,3,4\}) \\ \sim CanOpen(d) \leftarrow Prize = d. \\ CanOpen(d) \leftarrow not\ \sim CanOpen(d). \\ random(Prize). \qquad random(Selected). \\ random(Open : \{x : CanOpen(x)\}). \\ pr(Prize = 1) = 0.3. \;\; pr(Prize = 3) = 0.2. \\ Obs(Selected = 1). \;\; Obs(Open = 2). \;\; Obs(Prize \neq 2). \end{array}$$

Intuitively, the translation $\mathsf{plog2lpmln}(\Pi_2)$ (i) reifies each atom $c = v$ in Π_2 into a form of $Poss(c = v)$, $PossWithAssPr(c = v)$, and $PossWithDefPr(c = v)$; (ii) defines each of these reified atoms by hard rules, e.g., $\alpha : \; Poss(Prize = d) \leftarrow not\ Intervene(Prize)$; and (iii) assigns the probabilities by soft rules, e.g., $ln(0.3) : \; \bot \leftarrow not\ AssPr(Prize = 1)$. The full translation is too long to be put here, please refer to Example 3 in [20] for details.

5.3 From LPOD to Answer Set Programs

▶ **Theorem 5.** *(from [21]) Under any of the four preference criteria, the preferred answer sets of an LPOD Π of signature σ are exactly the preferred answer sets on σ of* lpod2asp(Π).

▶ **Example 6.** Consider the following LPOD Π_3 about picking a hotel near the Grand Canyon. $hotel(1)$ is a 2-star hotel but is close to the Grand Canyon, $hotel(2)$ is a 3-star hotel and the distance is medium, and $hotel(3)$ is a 4-star hotel but is too far.

$$
\begin{array}{ll}
close \times med \times far \times tooFar. & \leftarrow hotel(2),\ not\ med. \\
star4 \times star3 \times star2. & \leftarrow hotel(2),\ not\ star3. \\
1\{hotel(X) : X = 1..3\}1. & \leftarrow hotel(3),\ not\ tooFar. \\
\leftarrow hotel(1),\ not\ close. & \leftarrow hotel(3),\ not\ star4. \\
\leftarrow hotel(1),\ not\ star2. &
\end{array}
$$

The translation lpod2asp(Π_3) is based on the definition of the *assumption program*, $AP(x_1, x_2)$, where $x_1 \in \{0, \dots, 4\}$ and $x_2 \in \{0, \dots, 3\}$. Intuitively, the value of x_i denotes an assumption about LPOD rule i: if $x_i = 0$, the body of rule i is false, thus no atom will be derived by rule i; if $x_i > 0$, the boy of rule i is true, and the x_i-th atom will be derived by rule i (which requires that all atoms in the head of rule i with a index lower than x_i must be false). An assumption program $AP(x_1, x_2)$ is initialized by a choice rule and a weak constraint (which makes sure that all consistent assumption programs are considered).

```
{ap(X1,X2): X1=0..4, X2=0..3}.              :~ ap(X1,X2). [-1, X1, X2]
```

The assumption programs include all regular rules in Π. Note that (i) we turn each atom a in Π into $a(X_1, X_2)$ so that the answer sets of assumption program $AP(x_1, x_2)$ are in its own name space (x_1, x_2); (ii) we add $ap(X_1, X_2)$ in the body of each rule so that these rules will not be "effective" if the assumption program $AP(X_1, X_2)$ is inconsistent.

```
1{hotel(H,X1,X2): H=1..3}1 :- ap(X1,X2).
:- ap(X1,X2), hotel(1,X1,X2), not close(X1,X2).
:- ap(X1,X2), hotel(1,X1,X2), not star2(X1,X2).
:- ap(X1,X2), hotel(2,X1,X2), not med(X1,X2).
:- ap(X1,X2), hotel(2,X1,X2), not star3(X1,X2).
:- ap(X1,X2), hotel(3,X1,X2), not tooFar(X1,X2).
:- ap(X1,X2), hotel(3,X1,X2), not star4(X1,X2).
```

Besides, the assumption programs include all assumptions that we record in (x_1, x_2).

```
% close * med * far * tooFar.
body_1(X1,X2) :- ap(X1,X2).
:- ap(X1,X2), X1=0, body_1(X1,X2).
:- ap(X1,X2), X1>0, not body_1(X1,X2).

close(X1,X2) :- body_1(X1,X2), X1=1.
med(X1,X2) :- body_1(X1,X2), X1=2.
far(X1,X2) :- body_1(X1,X2), X1=3.
tooFar(X1,X2) :- body_1(X1,X2), X1=4.

X1=1 :- body_1(X1,X2), close(X1,X2).
X1=2 :- body_1(X1,X2), med(X1,X2), not close(X1,X2).
X1=3 :- body_1(X1,X2), far(X1,X2), not close(X1,X2), not med(X1,X2).
X1=4 :- body_1(X1,X2), tooFar(X1,X2), not close(X1,X2),
        not med(X1,X2), not far(X1,X2).

% star4 * star3 * star2.
```

```
body_2(X1,X2) :- ap(X1,X2).

:- ap(X1,X2), X2=0, body_2(X1,X2).
:- ap(X1,X2), X2>0, not body_2(X1,X2).

star4(X1,X2) :- body_1(X1,X2), X2=1.
star3(X1,X2) :- body_1(X1,X2), X2=2.
star2(X1,X2) :- body_1(X1,X2), X2=3.

X2=1 :- body_1(X1,X2), star4(X1,X2).
X2=2 :- body_1(X1,X2), star3(X1,X2), not star4(X1,X2).
X2=3 :- body_1(X1,X2), star2(X1,X2), not star4(X1,X2),
        not star3(X1,X2).
```

To calculate the satisfaction degrees D_1, D_2 of two LPOD rules, lpod2asp(Π_3) contains

```
degree(ap(X1,X2), D1, D2) :- ap(X1,X2), D1=#max{1;X1}, D2=#max{1;X2}.
```

Note that all answer sets of $AP(x_1, x_2)$ will have a same satisfaction degree for each LPOD rule. Thus we also use $ap(x_1, x_2)$ to denote an answer set of $AP(x_1, x_2)$ in the following set of rules. To compare two candidate answer set S_1 and S_2 according to, say, Pareto-preference, and to determine whether an answer set of $AP(x_1, x_2)$ is a Pareto-preferred answer set, lpod2asp(Π_3) contains

```
equ(S1,S2) :- degree(S1,D1,D2), degree(S2,D1,D2).

prf(S1,S2) :- degree(S1,D11,D12), degree(S2,D21,D22), not equ(S1,S2),
              D11<=D21, D12<=D22.

pAS(X1, X2) :- ap(X1, X2), {prf(S, ap(X1,X2))}0.
```

5.4 From CR-Prolog$_2$ to Answer Set Programs

▶ **Theorem 7.** *(from [21]) For any CR-Prolog$_2$ program Π of signature σ, (a) the projections of the generalized answer sets of Π onto σ are exactly the generalized answer sets on σ of* crp2asp(Π). *(b) the projections of the candidate answer sets of Π onto σ are exactly the candidate answer sets on σ of* crp2asp(Π). *(c) the preferred answer sets of Π are exactly the preferred answer sets on σ of* crp2asp(Π).

▶ **Example 8.** (From [4]) Consider the following CR-Prolog$_2$ program Π_4:

$$\begin{array}{lll} q \leftarrow t. & p \leftarrow not\ q. & 1: \quad t \stackrel{+}{\leftarrow}. \\ s \leftarrow t. & r \leftarrow not\ s. & 2: \quad q \times s \stackrel{+}{\leftarrow}. \\ & \leftarrow p, r. & \end{array}$$

The idea behind crp2asp is similar to that for lpod2asp. crp2asp(Π_4) consists of (i) all consistent assumption programs

```
{ap(X1,X2): X1=0..1, X2=0..2}.          :~ ap(X1,X2). [-1,X1,X2]

q(X1,X2) :- ap(X1,X2), t(X1,X2).
s(X1,X2) :- ap(X1,X2), t(X1,X2).
p(X1,X2) :- ap(X1,X2), not q(X1,X2).
r(X1,X2) :- ap(X1,X2), not s(X1,X2).
:- ap(X1,X2), p(X1,X2), r(X1,X2).
```

```
% 1: t <+-.
t(X1,X2) :- ap(X1,X2), X1=1.

% 2: q*s <+-.
q(X1,X2) :- ap(X1,X2), X2=1.
s(X1,X2) :- ap(X1,X2), X2=2.
```

(ii) the definition of dominate as well as the definition of candidate answer set

```
dominate(ap(X1,X2), ap(Y1,Y2)) :- ap(X1,X2), ap(Y1,Y2), 0<X1, X1<Y1.
dominate(ap(X1,X2), ap(Y1,Y2)) :- ap(X1,X2), ap(Y1,Y2), 0<X2, X2<Y2.

candidate(X1,X2) :- ap(X1,X2), {dominate(SP,ap(X1,X2))}0.
```

(iii) the definition of lessCrRuleApplied as well as the definition of preferred answer set

```
lessCrRuleApplied(ap(X1,X2), ap(Y1,Y2)) :- candidate(X1,X2),
        candidate(Y1,Y2), 1{X1!=Y1;X2!=Y2}, X1<=Y1, X2<=Y2.

pAS(X1,X2) :- candidate(X1,X2), {lessCrRule(SP,ap(X1,X2))}0.
```

6 Open Issues and Expected Achievements

One issue is that, among the 4 translations, only lpmln2wc has an implemented compiler. So, for now, most translations must be done manually. However, we may not implement the compilers for the translations lpod2asp and crpt2asp, since they are exponential to the number of non-regular rules.

Another issue is, currently, we are working on combining quantitative and qualitative uncertainty in a single formalism, but it is still not clear how these two kinds of uncertainty merge together. For example, if there is a preference rule saying "football > ping-pong > basketball" with a quantitative confidence 5, and there is another preference rule saying "indoor game > outdoor game" with confidence 10, what should be the order of these activities? To answer this question, we should first answer "how should the confidence be arranged in a rule without loss of generality?" The follow-up question is "what is the confidence of basketball if there is a probability of 70% that it is an indoor game?"

As for the future work, we will check whether the recent approach, Asprin [10], can be used to implement LPOD, CR-Prolog$_2$, LP^{MLN}, and even P-log. At the meantime, we will start to combine quantitative and qualitative uncertainty from tuning the semantics of LPOD to include quantitative uncertainty in its syntax and semantics. After the formalism is created and well defined, we will prove its expressivity and implement a compiler for it.

References

1 Vernon Asuncion, Yan Zhang, and Heng Zhang. Logic programs with ordered disjunction: first-order semantics and expressiveness. In *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 2–11. AAAI Press, 2014.

2 Evgenii Balai and Michael Gelfond. On the relationship between P-log and LP$^{\text{MLN}}$. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 915–921, 2016.

3 Marcello Balduccini. Cr-models: an inference engine for cr-prolog. In *Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 18–30. Springer-Verlag, 2007.

4 Marcello Balduccini, Marcello Balduccini, and Veena Mellarkod. Cr-prolog with ordered disjunction. In *In ASP03 Answer Set Programming: Advances in Theory and Implementation, volume 78 of CEUR Workshop proceedings*, 2003.

5 Marcello Balduccini and Michael Gelfond. Logic programs with consistency-restoring rules. In *International Symposium on Logical Formalization of Commonsense Reasoning, AAAI 2003 Spring Symposium Series*, pages 9–18, 2003.

6 Marcello Balduccini and Veena Mellarkod. A-prolog with cr-rules and ordered disjunction. In *Intelligent Sensing and Information Processing, 2004. Proceedings of International Conference on*, pages 1–6. IEEE, 2004.

7 Chitta Baral, Michael Gelfond, and J. Nelson Rushton. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming*, 9(1):57–144, 2009.

8 Gerhard Brewka. Logic programming with ordered disjunction. In *AAAI/IAAI*, pages 100–105, 2002.

9 Gerhard Brewka. Preferences in answer set programming. In *CAEPIA*, volume 4177, pages 1–10. Springer, 2005.

10 Gerhard Brewka, James P Delgrande, Javier Romero, and Torsten Schaub. asprin: Customizing answer set preferences without a headache. In *AAAI*, pages 1467–1474, 2015.

11 Gerhard Brewka, Ilkka Niemelä, and Tommi Syrjänen. Implementing ordered disjunction using answer set solvers for normal programs. In *European Workshop on Logics in Artificial Intelligence*, pages 444–456. Springer, 2002.

12 Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Enhancing disjunctive datalog by constraints. *IEEE Transactions on Knowledge and Data Engineering*, 12(5):845–860, 2000.

13 Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, pages 2462–2467, 2007.

14 James P Delgrande, Torsten Schaub, and Hans Tompits. A framework for compiling preferences in logic programs. *Theory and Practice of Logic Programming*, 3(2):129–187, 2003.

15 Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. Stable models and circumscription. *Artificial Intelligence*, 175:236–263, 2011.

16 Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080. MIT Press, 1988.

17 Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

18 Joohyung Lee, Yunsong Meng, and Yi Wang. Markov logic style weighted rules under the stable model semantics. In Technical Communications of the 31st International Conference on Logic Programming, 2015.

19 Joohyung Lee and Yi Wang. Weighted rules under the stable model semantics. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 145–154, 2016.

20 Joohyung Lee and Zhun Yang. LPMLN, weak constraints, and P-log. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 1170–1177, 2017.

21 Joohyung Lee and Zhun Yang. Translating lpod and cr-prolog2 into standard answer set programs. *arXiv preprint arXiv:1805.00643*, 2018.

22 Judea Pearl. *Causality: models, reasoning and inference*, volume 29. Cambridge Univ Press, 2000.

23 Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.