Reasoning and Learning with Probabilistic Answer Set Programming

by

Yi Wang

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved March 2019 by the
Graduate Supervisory Committee:

Joohyung Lee, Chair
Chitta Baral
Subbarao Kambhampati
Sriraam Natarajan
Siddharth Srivastava

ARIZONA STATE UNIVERSITY

May 2019

ABSTRACT

Knowledge Representation (KR) is one of the prominent approaches to Artificial Intelligence (AI) that is concerned with representing knowledge in a form that computer systems can utilize to solve complex problems. Answer Set Programming (ASP), based on the stable model semantics, is a widely-used KR framework that facilitates elegant and efficient representations for many problem domains that require complex reasoning.

However, while ASP is effective on deterministic problem domains, it is not suitable for applications involving quantitative uncertainty, for example, those that require probabilistic reasoning. Furthermore, it is hard to utilize information that can be statistically induced from data with ASP problem modeling.

This dissertation presents the language LP^{MLN}, which is a probabilistic extension of the stable model semantics with the concept of weighted rules, inspired by Markov Logic. An LP^{MLN} program defines a probability distribution over "soft" stable models, which may not satisfy all rules, but the more rules with the bigger weights they satisfy, the bigger their probabilities. LP^{MLN} takes advantage of both ASP and Markov Logic in a single framework, allowing representation of problems that require both logical and probabilistic reasoning in an intuitive and elaboration tolerant way.

This dissertation establishes formal relations between LP^{MLN} and several other formalisms, discusses inference and weight learning algorithms under LP^{MLN}, and presents systems implementing the algorithms. LP^{MLN} systems can be used to compute other languages translatable into LP^{MLN}. The advantage of LP^{MLN} for probabilistic reasoning is illustrated by a probabilistic extension of the action language $\mathcal{BC}+$, called $p\mathcal{BC}+$, defined as a high-level notation of LP^{MLN} for describing transition systems. Various probabilistic reasoning about transition systems, especially probabilistic diagnosis, can be modeled in $p\mathcal{BC}+$ and computed using LP^{MLN} systems.

$p\mathcal{BC}+$ is further extended with the notion of utility, through a decision-theoretic extension of LP^{MLN}, and related with Markov Decision Process (MDP) in terms of policy optimization problems. $p\mathcal{BC}+$ can be used to represent (PO)MDP in a succinct and elaboration tolerant way, which enables planning with (PO)MDP algorithms in action domains whose description requires rich KR constructs, such as recursive definitions and indirect effects of actions.

To my parents

ACKNOWLEDGMENTS

During my long journey to attain my Ph.D, I have received support from many people. First of all, I would like to thank my advisor, Dr. Joohyung Lee. I first met Dr. Lee at his Introduction to Theoretical Computer Science course. As a beginner to the field of "theoretical" computer science, and an auditing student at that point, I was impressed by the clarity of his lecture. Subsequently, I decided to take several more courses from Dr. Lee later and eventually became his Ph.D student. Dr. Lee's patient supervision has greatly improved my academic skills, especially helping me develop a strong pursuit of technical clarity and soundness. Six years spent as part of Dr. Lee's group has changed my personality as a researcher, from one who's reluctant to express their own opinions to one who can speak out ideas confidently. I am also grateful for being given countless opportunities to present at conferences as well as industrial interships. The style of Dr. Lee's supervision allowed me to be productive at research while still having time to enjoy hobbies. Thank you Dr. Lee, I enjoyed being your Ph.D student.

Next, I would like to thank the other students in Dr. Lee's group. Those who are more senior than me, Yunsong Meng, Michael Bartholomew, Joseph Babb and Yu Zhang, helped grow my passion for computer science and AI with inspiring discussions and words of encouragement. Those who joined the group later than me, Zhun Yang, Jiaxuan Pang and Man Luo, offered great help in improving my work with valuable comments, and kept my life as a Ph.D student pleasant and cheerful with various out-of-lab activities. I feel privileged to work with all these brilliant and kind people.

I would like to also thank my committee members, Chitta Baral, Subbarao Kambhampati, Sriraam Natarajan and Siddharth Srivastava, for their helpful comments and suggestions.

Finally, I would like to say thank you to my parents from the bottom of my heart for being supporting and encouraging no matter what decision I make for my life. My father kindled my passion for computer science at the very beginning by showing me and letting me play with a real computer when I was 4 years old. He also taught me the basics of programming and the binary number system when I was in primary school, in a way that is so lively and entertaining, which turned abstract mathematical concepts into something even a small child can enjoy. Interactions with my father throughout my life shaped my critical thinking ability as well as my passion for intellectual pursuits.

My work in this thesis was partially supported by the National Science Foundation under Grants IIS-1526301, IIS-1319794 and IIS-1815337.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

Knowledge Representation (KR) is one of the prominent approaches to Artificial Intelligence (AI). It solves problems in AI by creating representations of the problem domain in a form that can facilitate automated reasoning about the problem domain. Once the representation is created, the solutions of the problem can be derived automatically from the semantics of the underlying formalism.

Answer Set Programming (ASP) is a widely-used KR framework that can facilitate compact and intuitive representations for many problem domains that require commonsense reasoning. The problem domain is encoded as an answer set program, so that the "answer sets" of the program, which can be found automatically by ASP solvers, correspond to the solutions of the problem. The nonmonotonicity of the underlying semantics of ASP, the stable model semantics, enables various types of reasoning including defeasible reasoning, causal reasoning, diagnostic reasoning, etc., many of which are hard to be modeled with SAT based logic formalisms. Useful knowledge representation constructs and efficient solvers allow ASP to handle various combinatorial search and commonsense reasoning problems in knowledge intensive domains elegantly and efficiently.

However, difficulty remains when it comes to domains with quantitative uncertainty, for example, reasoning tasks that involve probabilistic inference. The fact that ASP does not distinguish answer sets that are more likely to be true limits its application domains. Furthermore, due to the "crisp" nature of the stable model semantics, it is hard to utilize information that can be statistically induced from data with ASP problem modeling, since data is noisy in most real-world situations.

On the other hand, Markov Logic (Domingos and Lowd (2009)) is a prominent approach in Statistical Relational Learning (SRL), aimed at combining probabilistic graphical models and logic. The idea is to assign machine-learnable weights to logic formulas, so that a model of the logic theory does not have to satisfy all formulas, and the weight scheme induces a probability distribution over all models. However, since Markov Logic is based on standard first-order semantics, it is weak for commonsense reasoning. For example, causality and inductive definitions are hard to be succinctly represented with Markov Logic.

To overcome the difficulty in modeling quantitative uncertainty with ASP, we propose the language LP^{MLN}, which is a probabilistic extension of ASP that combines the advantages of ASP and Markov Logic in a single framework. In LP^{MLN}, we introduce the notion of weighted rules under the stable model semantics, following the log-linear models of Markov Logic. LP^{MLN} allows representations of commonsense reasoning problems that require both logical and probabilistic reasoning in an intuitive and elaboration tolerant way. Furthermore, thanks to its close relation to Markov Logic, some learning methods developed for Markov Logic can be adapted for LP^{MLN} learning, in this way bringing machine learning algorithms in the context of a KR formalism.

The relation between LP^{MLN} and Markov Logic is analogous to the known relationship between ASP and SAT, in that the former follows the stable model semantics and the latter follows a standard SAT semantics. The relationship between LP^{MLN} and ASP is analogous to the relationship between Markov Logic and SAT, in that the former is a weighted extension of the latter. Figure 1.1 summarizes the relationships between ASP, Markov Logic, SAT and LP^{MLN}.

In this dissertation, we define the syntax and semantics of LP^{MLN}, and discuss its formal relations to other formalisms such as ASP, Markov Logic, ProbLog, P-

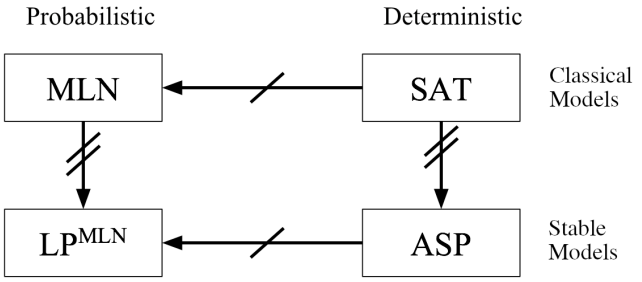

Figure 1.1: The Relation between ASP, Markov Logic, SAT and LP$^{\text{MLN}}$

log and Pearl's Causal Model (PCM). Based on the relationships, we implemented two systems LPMLN2ASP and LPMLN2MLN for LP$^{\text{MLN}}$ inference. The LP$^{\text{MLN}}$ inference systems can be used to compute other languages that are translatable into LP$^{\text{MLN}}$. We present the LP$^{\text{MLN}}$ weight learning system and illustrate how we can learn weights for probabilistic extensions of knowledge-rich domains that involve reachability analysis and reasoning about action dynamics, where ASP has been useful in the deterministic case.

To illustrate LP$^{\text{MLN}}$'s capability of reasoning in action domains, we present a probabilistic extension of the action language $\mathcal{BC}+$, called $p\mathcal{BC}+$, which is a high-level notation of LP$^{\text{MLN}}$ for describing transition systems. We show how probabilistic reasoning about dynamic domains, such as prediction and postdiction, as well as probabilistic diagnosis, can be modeled in the probabilistic language and computed using LP$^{\text{MLN}}$ inference and learning system.

In many probabilistic decision problems, the goal is to find a decision choice that yields the maximum expected utility. We extend LP$^{\text{MLN}}$ with the notion of utility, resulting in Decision-Theoretic LP$^{\text{MLN}}$ (DT-LP$^{\text{MLN}}$). based on DT-LP$^{\text{MLN}}$, we introduce the notion of utility in $p\mathcal{BC}+$, and further define the policy optimization problem in the context of $p\mathcal{BC}+$. We show that the policy optimization problem in terms of $p\mathcal{BC}+$ coincide with that of Markov Decision Process (MDP). We demonstrate that $p\mathcal{BC}+$ can be used to represent (PO)MDP in a succinct and elaboration tolerant way,

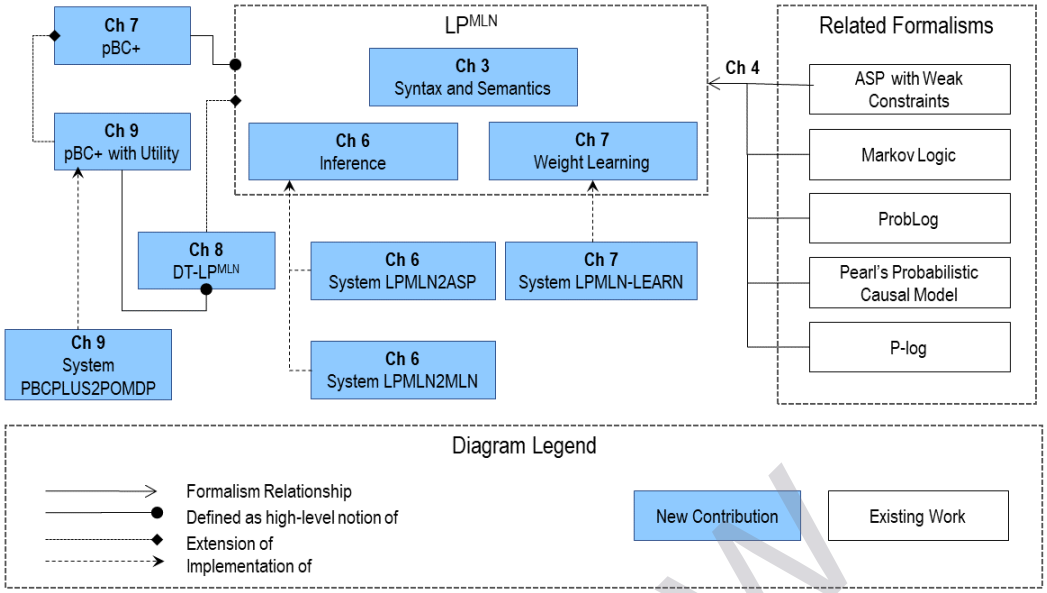

Figure 1.2: Dissertation Outline

which enables planning with (PO)MDP algorithms in action domains whose description requires rich KR constructs, such as recursive definitions and indirect effects of actions.

In these initial works on LP$^{\text{MLN}}$, our focus is on defining the language, studying its theoretical properties and exploring its expressivity. Computational efficiency is not the focus of this dissertation and the systems presented here are prototypical. Improving the efficiency of the systems and conducting empirical study of LP$^{\text{MLN}}$ in real-world applications are important future works. Figure 1.2 summarizes the contributions from this dissertation and indicates where each contribution is presented in this dissertation.

The rest of this dissertation is organized as follows: Chapter 2 reviews necessary background information, including the stable model semantics and Markov Logic. Chapter 3 defines the syntax and semantics of language LP$^{\text{MLN}}$. Chapter 4 discusses the formal relationship between LP$^{\text{MLN}}$ and related formalisms. In Chapter 5 and

Chapter 6 we discuss inference and learning methods for LP^{MLN}, respectively. Chapter 7 presents the action language $p\mathcal{BC}+$ defined in terms of LP^{MLN}. Chapter 8 presents the decision-theoretic extension of LP^{MLN}. Based on the decision-theoretic extension, Chapter 9 presents an extension of $p\mathcal{BC}+$ where policy optimization problem can be defined, and relates it with Markov Decision Process. We conclude in Chapter 10 with prospective contributions.

Chapter 2

BACKGROUND

Throughout this paper, we assume a first-order signature σ that has finitely many Herbrand interpretations.

2.1 Stable Model Semantics

A *rule* over σ is of the form

$$A_1; \ldots; A_k \ \leftarrow \ A_{k+1}, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n, \mathit{not}\ \mathit{not}\ A_{n+1}, \ldots, \mathit{not}\ \mathit{not}\ A_p \tag{2.1}$$

($0 \leq k \leq m \leq n \leq p$) where all A_i are atoms of σ possibly containing object variables. We write $\{A_1\}^{\mathrm{ch}} \leftarrow \mathit{Body}$ to denote the rule $A_1 \leftarrow \mathit{Body}, \mathit{not}\ \mathit{not}\ A_1$. This expression is called a "choice rule" in ASP.

We will often identify (2.1) with the implication:

$$A_1 \vee \cdots \vee A_k \ \leftarrow \ A_{k+1} \wedge \ldots \wedge A_m \wedge \neg A_{m+1} \wedge \ldots \wedge \neg A_n \wedge \neg\neg A_{n+1} \wedge \ldots \wedge \neg\neg A_p \ . \tag{2.2}$$

A *logic program* is a finite set of rules. A logic program is called *ground* if it contains no variables.

We say that an Herbrand interpretation I is a *model* of a ground program Π if I satisfies all implications (2.2) in Π. Such models can be divided into two groups: "stable" and "non-stable" models, which are distinguished as follows. The *reduct* of Π relative to I, denoted Π^I, consists of "$A_1 \vee \cdots \vee A_k \ \leftarrow \ A_{k+1} \wedge \cdots \wedge A_m$" for all rules (2.2) in Π such that $I \models \neg A_{m+1} \wedge \cdots \wedge \neg A_n \wedge \neg\neg A_{n+1} \wedge \cdots \wedge \neg\neg A_p$. The Herbrand interpretation I is called a *(deterministic) stable model* of Π (denoted by

$I \models_{\text{SM}} \Pi$) if I is a minimal Herbrand model of Π^I. (Minimality is in terms of set inclusion. We identify an Herbrand interpretation with the set of atoms that are true in it.) For example, the stable models of the program

$$P \leftarrow Q \qquad Q \leftarrow P \qquad P \leftarrow not\ R \qquad R \leftarrow not\ P \tag{2.3}$$

are $\{P, Q\}$ and $\{R\}$. The reduct relative to $\{P, Q\}$ is $\{P \leftarrow Q. \quad Q \leftarrow P. \quad P.\}$, for which $\{P, Q\}$ is the minimal model; the reduct relative to $\{R\}$ is $\{P \leftarrow Q. \quad Q \leftarrow P. \quad R.\}$, for which $\{R\}$ is the minimal model.

The definition is extended to any non-ground program Π by identifying it with $gr_\sigma[\Pi]$, the ground program obtained from Π by replacing every variable with every ground term of σ.

The semantics is extended to allow some useful constructs, such as aggregates and abstract constraints (e.g., Niemelä and Simons (2000); Faber *et al.* (2004); Ferraris (2005); Son *et al.* (2006); Pelov *et al.* (2007)), which are proved to be useful in many KR domains.

2.2 Weak Constraints

Weak constraint (Buccafurri *et al.* (2000); Calimeri *et al.* (2012)) is a simple extension of answer set programs for expressing quantitative preference among answer sets. It is a part of ASP Core 2 language and has turned out to be useful in many practical applications. It is implemented in standard ASP solvers such as CLINGO and DLV. In Section 4.1.3, we will show that LP^{MLN} programs can be turned into ASP programs with weak constraints. In this section, we review the syntax and semantics of weak constraints.

A *weak constraint* has the form

$$:\sim F \quad [\mathit{Weight}\ @\ \mathit{Level}]$$

where F is a conjunction of literals, *Weight* is a real number, and *Level* is a nonnegative integer.

Let Π be a program $\Pi_1 \cup \Pi_2$, where Π_1 is an answer set program that does not contain weak constraints, and Π_2 is a set of ground weak constraints. We call I a stable model of Π if it is a stable model of Π_1. For every stable model I of Π and any nonnegative integer l, the *penalty* of I at level l, denoted by $Penalty_\Pi(I, l)$, is defined as

$$\sum_{:\sim\ F[w@l]\in\Pi_2,\ I\models F} w.$$

For any two stable models I and I' of Π, we say I is *dominated* by I' if

- there is some nonnegative integer l such that $Penalty_\Pi(I', l) < Penalty_\Pi(I, l)$ and
- for all integers $k > l$, $Penalty_\Pi(I', k) = Penalty_\Pi(I, k)$.

A stable model of Π is called *optimal* if it is not dominated by another stable model of Π.

The input language of CLINGO allows non-ground weak constraints that contain tuples of terms.

2.3 Markov Logic

The following is a review of Markov Logic from Richardson and Domingos (2006). A *Markov Logic Network (MLN)* $\mathbb{L}$ of signature σ is a finite set of pairs $\langle F, w\rangle$ (also written as a "weighted formula" $w : F$), where F is a first-order formula of σ and w is either a real number or a symbol α denoting the "infinite weight." We say that $\mathbb{L}$ is *ground* if its formulas contain no variables.

We first define the semantics for ground MLNs. For any ground MLN $\mathbb{L}$ of signature σ and any Herbrand interpretation I of σ, we define $\mathbb{L}_I$ to be the set of formulas

in $\mathbb{L}$ that are satisfied by I. The *weight* of an interpretation I under $\mathbb{L}$, denoted $W_{\mathbb{L}}(I)$, is defined as

$$W_{\mathbb{L}}(I) = exp\left(\sum_{\substack{w:F \in \mathbb{L} \\ F \in \mathbb{L}_I}} w \right). \tag{2.4}$$

The probability of I under $\mathbb{L}$, denoted $P_{\mathbb{L}}(I)$, is defined as

$$P_{\mathbb{L}}(I) = \lim_{\alpha \to \infty} \frac{W_{\mathbb{L}}(I)}{\sum_{J \in PW} W_{\mathbb{L}}(J)},$$

where PW ("Possible Worlds") is the set of all Herbrand interpretations of σ. We say that I is a *model* of $\mathbb{L}$ if $P_{\mathbb{L}}(I) \neq 0$.

The basic idea of MLNs is to allow formulas to be soft constrained, where a model does not have to satisfy all formulas, but is associated with the weight that is contributed by the satisfied formulas. For every interpretation (i.e., possible world) I, there is a unique maximal subset of formulas in the MLN that I satisfies, which is $\mathbb{L}_I$, and the weight of I is obtained from the weights of those "contributing" formulas in $\mathbb{L}_I$. An interpretation that does not satisfy certain formulas receives "penalties" because such formulas do not contribute to the weight of that interpretation.

The definition is extended to any non-ground MLN by identifying it with its *ground instance*. Any MLN $\mathbb{L}$ of signature σ can be identified with the ground MLN, denoted $gr_\sigma[\mathbb{L}]$, by turning each formula in $\mathbb{L}$ into a set of ground formulas as described in (Richardson and Domingos, 2006, Table II). The weight of each ground formula in $gr_\sigma[\mathbb{L}]$ is the same as the weight of the formula in $\mathbb{L}$ from which the ground formula is obtained. For non-ground MLN, (2.4) can be written as

$$W_{\mathbb{L}}(I) = exp\left(\sum_{w_i:F_i \in \mathbb{L}} w_i m_i(I) \right)$$

where $m_i(I)$ is the number of ground instances of $w_i : F_i$ in $\mathbb{L}_I$.

2.4 Markov Decision Process

Markov Decision Process (MDP) provides a mathematical framework for modeling sequential decision making in domains where the effects of actions can be stochastic. In Chapter 9, we will relate the probabilistic action language $p\mathcal{BC}+$, which is defined in terms of LP^{MLN}, to MDP, showing that the finite horizon policy optimization problem under $p\mathcal{BC}+$ and MDP coincide, and $p\mathcal{BC}+$ can be used to represent MDP in a succinct and elaboration tolerant way.

A *Markov Decision Process (MDP) M* is a tuple $\langle S, A, T, R\rangle$ where (i) S is a set of states; (ii) A is a set of actions; (iii) $T : S \times A \times S \to [0, 1]$ defines transition probabilities; (iv) $R : S \times A \times S \to \mathbb{R}$ is the reward function.

2.4.1 Finite Horizon Policy Optimization

Given a nonnegative integer m as the maximum timestamp, and a history

$$\langle s_0, a_0, s_1, \ldots, s_{m-1}, a_{m-1}, s_m\rangle$$

such that each $s_i \in S$ ($i \in \{0, \ldots, m\}$) and each $a_i \in A$ ($i \in \{0, \ldots, m-1\}$), the *total reward* of the history under MDP M is defined as

$$R_M(\langle s_0, a_0, s_1, \ldots, s_{m-1}, a_{m-1}, s_m\rangle) = \sum_{i=0}^{m-1} R(s_i, a_i, s_{i+1}).$$

The probability of $\langle s_0, a_0, s_1, \ldots, s_{m-1}, a_{m-1}, s_m\rangle$ under MDP M is defined as

$$P_M(\langle s_0, a_0, s_1, \ldots, s_{m-1}, a_{m-1}, s_m\rangle) = \prod_{i=0}^{m-1} T(s_i, a_i, s_{i+1}).$$

A *non-stationary policy* $\pi : S \times ST \mapsto A$ is a function from $S \times ST$ to A, where $ST = \{0, \ldots, m-1\}$. Given an initial state s_0, the *expected total reward* of a non-