TPLP **21** (5): 557–574, 2021. © The Author(s), 2021. Published by Cambridge University Press
doi:10.1017/S1471068421000417 First published online 27 September 2021

Nonground Abductive Logic Programming with Probabilistic Integrity Constraints

ELENA BELLODI, MARCO GAVANELLI, RICCARDO ZESE and EVELINA LAMMA
Department of Engineering - University of Ferrara, Ferrara, Italy
(*e-mails:* elena.bellodi@unife.it, marco.gavanelli@unife.it, riccardo.zese@unife.it, lme@unife.it)

FABRIZIO RIGUZZI
Department of Mathematics and Computer Science - University of Ferrara, Ferrara, Italy
(*e-mail:* fabrizio.riguzzi@unife.it)

submitted 9 August 2021; revised 30 August 2021; accepted 30 August 2021

Abstract

Uncertain information is being taken into account in an increasing number of application fields. In the meantime, abduction has been proved a powerful tool for handling hypothetical reasoning and incomplete knowledge. Probabilistic logical models are a suitable framework to handle uncertain information, and in the last decade many probabilistic logical languages have been proposed, as well as inference and learning systems for them. In the realm of Abductive Logic Programming (ALP), a variety of proof procedures have been defined as well. In this paper, we consider a richer logic language, coping with probabilistic abduction with variables. In particular, we consider an ALP program enriched with integrity constraints *à la* IFF, possibly annotated with a probability value. We first present the overall abductive language and its semantics according to the Distribution Semantics. We then introduce a proof procedure, obtained by extending one previously presented, and prove its soundness and completeness.

KEYWORDS: abduction, integrity constraints, distribution semantics, probabilistic IFF

1 Introduction

Reasoning in uncertain domains is a common task for humans, and the human brain also has the capability to explore different scenarios by considering a variety of possible hypotheses, in order to take a decision. The ability of the human brain, and human expertise in specific domains, meant that humans were not replaceable by a machine in their reasoning tasks, so far. Nonetheless, in the last decade, the huge increase of available data and knowledge in many domains (e.g. in medicine, science, physics, etc.), often in a form that can be processed automatically, strongly pushes toward forms of automatic reasoning able to cope with uncertainty, probabilities and hypotheses, also in order to have reasoning systems facing humans, and to achieve a more reproducible (and verifiable) behavior. This is definitively a strong commitment for a trustworthy Artificial Intelligence. Logic Programming (LP) is a powerful class of languages to be a candidate for this purpose. The language itself is human-readable, and knowledge expressed in this class of languages can be validated by humans. Standard LP syntax, and LP-based

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

reasoning is the base for a variety of more expressive languages, and proof procedures. In particular, Probabilistic Logic Programming (PLP) (De Raedt and Kersting 2008) languages are simple yet powerful enough to represent different scenarios (Azzolini *et al.* 2019; Nguembang Fadja and Riguzzi 2017). Several of these languages are based on the *distribution semantics* (Sato 1995), such as PRISM (Sato 1995), LPAD (Vennekens *et al.* 2004), and ProbLog (De Raedt *et al.* 2007).

In the meantime, Abductive Logic Programming (ALP) (Kakas *et al.* 1998) has been proven very effective for hypothetical reasoning and for formalizing a variety of domains and applications, ranging from diagnosis to societies of agents and accountable protocols for multiagent systems, commitments and normative systems, web service choreographies. ALP is based on a declarative (model-theoretic) semantics and equipped with an operational semantics in terms of a proof procedure. The IFF proof procedure was proposed by Fung and Kowalski (1997) to support abductive reasoning also in presence of nonground abducible literals.

In this paper, we consider a richer logic language, coping with *probabilistic abduction with variables*. In particular, we consider an ALP program featuring also *Integrity Constraints* (ICs) similar to those offered by IFF, extended by the possibility of annotating them with a probability value, that makes it possible to handle uncertainty of real world domains. Probabilistic integrity constraints were defined by Riguzzi *et al.* (2020): programs containing such constraints are called Probabilistic Constraint Logic Theories (PCLTs) and may be learned directly from data by means of PASCAL ("ProbAbiliStic inductive ConstrAint Logic"), a system that learns both their structure and parameters from interpretations. However, a system able to reason about these integrity constraints is still missing. Consider the following example.

Example 1

Several years ago, a murder in Italy captured the attention of the population: a woman was murdered, and the main indicted person was her husband. The collected evidence included the following facts: the woman was killed in the house where she lived with her husband (`house1`); a pillow stained with the blood of the victim was found in another house (`house2`) some hundreds of km away; the husband had the keys of this second house.

We can represent the facts listed above in the following knowledge base:

```
has_keys(husband,house1).
has_keys(husband,house2).
```

The goal is to find the murderer M, that is, the person who entered both houses and killed the victim:

$G = \mathit{enter}(\mathtt{M},\mathtt{house1}), \mathit{killed}(\mathtt{M},\mathtt{woman}), \mathit{enter}(\mathtt{M},\mathtt{house2}).$

Predicates *killed* and *enter* are not known in the knowledge base, and they must be hypothesized: in ALP they are considered abducibles (in this paper, abducibles are in *italic*). Notice that this problem requires non-\ground abduction, since the murderer M is unknown, and it is not even possible to list all the possible murderers.

The relationship between having the keys and entering a house can be stated through an integrity constraint

$$\mathit{enter}(\mathtt{P},\mathtt{H}) \rightarrow \mathit{has_keys}(P,H). \qquad (ic_1)$$

saying that if a person `P` enters a house `H`, (s)he must have the keys.

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

However, the information encoded in ic_1 is not 100% sure: a person could also enter the house without having the keys, for example, by breaking a window or picking the lock of the door. The encoding would be more faithful to reality if a probability was associated to ic_1. The probability that ic_1 does not hold would be quite low, since entering with the keys is much easier than with unlawful methods, in which the intruder could be noticed and arrested.

Most previous works proposing probabilistic ALP have considered only abduction of ground atoms (Poole 1993; Arvanitis *et al.* 2006; Inoue *et al.* 2009; Kate and Mooney 2009; Raghavan 2011; Turliuc *et al.* 2013; Rotella and Ferilli 2013). To the best of our knowledge, only Christiansen (2008) focused on probabilistic nonground abduction, and this clearly extends the expressiveness of the language, as well as the answer capabilities of the proof-procedure: as a matter of fact, nonground abduction can provide answers in which not all parts of the answer are completely defined, and in which possible hypotheses, for some given evidence, are assumed without complete knowledge. This more closely resembles the capabilities of the human brain, that is able to hypothesize the existence of some action, force or individual causing an effect even without complete knowledge of whom, or which force is responsible. On the other hand, the proof procedure by Christiansen (2008) considers only a limited form of negation and assigns probabilities only to abducibles. In our work, instead, we extend ALP to give the possibility to probabilistically annotate integrity constraints to enrich the standard semantics. This leads to a probabilistic ALP reasoning system that clearly improves on existing systems such as the one by Christiansen, since annotating ICs also allows one to obtain the same effect of adding probabilities to abducibles. Moreover, machine learning systems already exist that learn probabilistic integrity constraints from data in the same form proposed in this paper, such as the aforementioned PASCAL; however, no ALP reasoning system existed before to exploit the learned constraints: in this paper we fill this gap.

The paper is organized as follows. Section 2 introduces information about the IFF semantics necessary for understanding Sections 3 and 4 that discuss language, syntax, and semantics of the proposed probabilistic nonground abduction proof-procedure, called IFF^{Prob}. Secttion 5 proves soundness and completeness of IFF^{Prob}, while Section 6 describes its implementation and Section 7 discusses some real world application examples. Section 8 shows preliminary scalability tests. Section 9 presents related work, and Section 10 concludes the paper.

2 Background

IFF Declarative Semantics. A IFF (Fung and Kowalski 1997) program is a triple $\langle \mathbf{KB}, \mathbf{IC}, \mathcal{A} \rangle$. $\mathcal{A}$ is a set of *abducible predicates*, or simply abducibles. An abducible is a predicate about which it is possible to make assumptions, such as about its truth. An abducible atom is an atom built on an abducible predicate. In this work, abduced literals can contain variables that are implicitly existentially quantified. **KB** is a set of LP clauses of the form

$$h \leftarrow b_1, \ldots, b_m,$$

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

where $m \geq 0$, and each b_i with $1 \leq i \leq m$ is a literal (i.e. an atom or its negation), while h is an atom that cannot be built on a predicate in $\mathcal{A}$. h is called *head*, while $b_1, \ldots, b_m$ is called the *body* of the clause.

IC is a set of implications, called *Integrity Constraints* (ICs). Each $ic \in \mathbf{IC}$ has the form

$$b_1, \ldots, b_n \rightarrow H_1 \vee \cdots \vee H_k,$$

where $b_1, \ldots, b_n$ is a conjunction of atoms called *Body* of the IC ic, while the disjunction $H_1 \vee \cdots \vee H_k$ is called the *Head*. Each H_i with $1 \leq i \leq k$ is a conjunction of literals $h_1, \ldots, h_l$.

A goal G is a conjunction of literals of the form $g_1, \ldots, g_m$ with $m \geq 1$.

To define the abductive semantics, we need to recall some definitions. An IC, clause or abducible is *ground* if it does not contain variables. A *substitution* θ is an assignment of variables to terms: $\theta = \{V_1/t_1, \ldots, V_n/t_n\}$. The application of a substitution $\theta = \{V_1/t_1, \ldots, V_n/t_n\}$ to an IC ic, indicated with $ic\theta$, is the replacement of each variable V_i appearing in ic and in θ with t_i. $ic\theta$ is called an *instance* of ic. θ is *grounding* for ic if $ic\theta$ is ground. The same applies to clauses and abducibles as well.

The IFF abductive semantics defines an abductive answer to a goal G as a pair (Δ, θ), where Δ is a set of abducible atoms and θ is a substitution for the variables contained in G, such that

$$\begin{array}{l} \mathbf{KB} \cup \Delta \models \mathbf{IC} \\ \mathbf{KB} \cup \Delta \models G\theta \end{array} \quad \text{and} \quad , \tag{1}$$

where $\models$ is entailment according to the 3-valued completion semantics (Kunen 1987), that is, we require that every 3-valued models of the completion (Clark 1978) of $\mathbf{KB} \cup \Delta$ is also a 3-valued model of IC. In such a case, we write $ALP \models^{\Delta} G$. Variables in G are free, the remaining variables in Δ are existentially quantified, while variables in **KB** and **IC** are implicitly universally quantified with scope the entire implication.

Riguzzi *et al.* (2020) introduce (although not in the context of ALP) the concept of *Probabilistic Integrity Constraint* (PIC)

$$p_i \ :: \ Body \rightarrow Head, \tag{2}$$

where p_i is a probability $\in [0, 1]$.

3 IFF^{Prob} syntax and declarative semantics

IFF^{Prob} programs define a probability distribution over IFF programs inspired by the distribution semantics proposed in the field of PLP (Sato 1995). A IFF^{Prob} program, similarly to IFF, is a triple $ALP_p = \langle \mathbf{KB}, \mathbf{IC}, \mathcal{A} \rangle$, however, in this case, the set **IC** is defined as $\mathbf{IC}^p \cup \mathbf{IC}^{np}$, where $\mathbf{IC}^p$ is a set of PICs as in Eq. 2 while $\mathbf{IC}^{np}$ is a set of (nonprobabilistic) ICs as in the IFF.

It is worth noting that associating probabilities to ICs makes IFF^{Prob} more general than other frameworks (Christiansen 2008) in which probabilities are attached to abducibles, since the fact that abducible $\mathtt{a}$ has probability p can be simply expressed by the PIC

$$(1 - p) :: \mathtt{a} \rightarrow false.$$

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

A IFF^{Prob} program $T = \langle \mathbf{KB}, \mathbf{IC}, \mathcal{A} \rangle$ defines a probability distribution over IFF programs called *worlds* where the constraints set includes each $ic \in \mathbf{IC}^{np}$, and some instances of any $ic \in \mathbf{IC}^{p}$ included with probability p_i.

An *atomic choice* is a triple (ic_i, θ_j, k) where ic_i is the ith PIC, θ_j is a substitution for the variables in ic_i, and $k \in \{0, 1\}$ indicates whether $ic_i\theta_j$ is chosen to be included in a world ($k = 1$) or not ($k = 0$). A *composite choice* E is a consistent set of atomic choices, where *consistent* means that E does not contain two atomic choices $(ic_i, \theta_j, k), (ic_i, \theta_j, m)$ with $k \neq m$, that is, $(ic_i, \theta_j, k) \in E, (ic_i, \theta_j, m) \in E \Rightarrow k = m$ (only one decision for each instance of each PIC).

The probability of a composite choice E is

$$P(E) = \prod_{(ic_i, \theta_j, 1) \in E} p_i \prod_{(ic_i, \theta_j, 0) \in E} (1 - p_i),$$

where p_i is the probability associated with PIC ic_i.

A *selection* σ is a *total* composite choice, that is, it contains an atomic choice for every instance up to renaming of every PIC of the IFF^{Prob} program. Given a selection σ, the world $\mathcal{W}_\sigma$ is defined as $\mathcal{W}_\sigma = \{ic_i\theta_j | (ic_i, \theta_j, 1) \in \sigma\}$. Let us indicate with $\mathcal{S}_T$ the set of all selections and with $\mathcal{W}_T$ the set of all worlds. The probability of a world $\mathcal{W}_\sigma$ is $P(\mathcal{W}_\sigma) = P(\sigma)$. $P(\mathcal{W}_\sigma)$ is a probability distribution over worlds, that is, $\sum_{\mathcal{W} \in \mathcal{W}_T} P(\mathcal{W}) = 1$.

Given a world $\mathcal{W}$, the conditional probability that a goal G is satisfied in the world is defined as $P(G|\mathcal{W}) = 1$ if $\exists \Delta \langle \mathbf{KB}, \mathcal{W}, \mathcal{A} \rangle \models^{\Delta} G$ and 0 otherwise. Given a goal G, its probability $P(G)$ can be defined by marginalizing the joint probability of the goal and the worlds:

$$P(G) = \sum_{\mathcal{W} \in \mathcal{W}_T} P(G, \mathcal{W}) = \sum_{\mathcal{W} \in \mathcal{W}_T} P(G|\mathcal{W}) P(\mathcal{W}) = \sum_{\mathcal{W} \in \mathcal{W}_T : \mathcal{W} \models G} P(\mathcal{W}).$$

Therefore, the probability of a goal G can be computed by summing the probability of the worlds where the goal is true.

Given a goal G to solve, an *explanation* is a composite choice E for G such that G is entailed by every world of ω_E, where $\omega_E = \{\mathcal{W}_\sigma | \sigma \in \mathcal{S}_T, \sigma \supseteq E\}$ is the set of worlds compatible with E.

We also define the set of worlds identified by a set of explanations $\mathcal{E}$ as $\omega_\mathcal{E} = \bigcup_{E \in \mathcal{E}} \omega_E$. A set of explanations $\mathcal{E}$ is *covering* with respect to G if every world $\mathcal{W} \in \mathcal{W}_T$ in which G is entailed is such that $\mathcal{W} \in \omega_\mathcal{E}$. Two explanations E_1 and E_2 are *incompatible* if their union is inconsistent, that is, given a PIC ic and a substitution θ, the explanations $E_1 = \{(ic, \theta, 1)\}$ and $E_2 = \{(ic, \theta, 0)\}$ are incompatible. A set $\mathcal{E}$ is pairwise incompatible if for all $E_1 \in \mathcal{E}$, $E_2 \in \mathcal{E}$, $E_1 \neq E_2$ implies E_1 and E_2 are incompatible. The probability of a pairwise incompatible set of explanations $\mathcal{E}$ is defined as $P(\mathcal{E}) = \sum_{E \in \mathcal{E}} P(E)$.

Definition 1 (*Probabilistic Abductive Answer*)
Given an ALP_p, (Δ, θ) is a *probabilistic abductive answer* for a goal G in the explanation E if

$$\begin{array}{rrcl} & \mathbf{KB} \cup \Delta & \models & G\theta \\ & \mathbf{KB} \cup \Delta & \models & \mathbf{IC}^{np} \\ \forall (ic_i, \theta_j, 1) \in E, & \mathbf{KB} \cup \Delta & \models & ic_i\theta_j \end{array}.$$

In such a case, we write $ALP_p \models_E^{\Delta} G$.

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

Example 1 (*Continued*)
We are now able to refine the previous example by associating a probability to the IC saying that if a person `P` enters a house `H`, (s)he must have the keys: such statement is uncertain, because it might be the case that an unauthorized person is able to enter a house also without having the keys, although with a lower probability, say 0.3:

$$0.7 :: \texttt{enter(P,H)} \rightarrow has_keys(P, H). \qquad (ic'_1)$$

4 IFF^{Prob} operational semantics

IFF Operational Semantics. Before defining the IFF^{Prob} operational semantics, we recap the IFF operational semantics (Fung and Kowalski 1997).

The following is the subset of IFF transitions that are relevant for this work, in a proof-theory style notation. In the following list, $\mathtt{a}$ is an abducible atom, while p and q represent atoms of either abducible or defined predicates, and N is a set of arguments.

propagation
$$\frac{\mathtt{a}(N) \qquad \mathtt{a}(N'), B \rightarrow H}{N = N', B \rightarrow H}$$

case analysis
$$\frac{(N = N', B) \rightarrow H}{(N = N'), \quad (B \rightarrow H)} \qquad \frac{(N = N', B) \rightarrow H}{N \neq N'}$$

unfolding
$$\frac{p(N) \qquad p(N') \leftarrow B}{N = N', B} \qquad \frac{p(N) \rightarrow H \qquad \begin{array}{l} p(N') \leftarrow B' \\ \ldots \\ p(N^k) \leftarrow B^k \end{array}}{\begin{array}{l} N = N', B' \rightarrow H \\ \ldots \\ N = N^k, B^k \rightarrow H \end{array}}$$

equality reqriting
$$\frac{[\exists E][\forall A] A = E}{\theta = \{A/E\}} \qquad \frac{[\exists E][\forall A] A \neq E}{false}$$

$$\frac{[\exists E_1][\exists E_2] E_1 = E_2}{\theta = \{E_1/E_2\}} \qquad \frac{X = t \qquad t \text{ does not contain } X}{\theta = \{X/t\}}$$

$$\frac{X = t \qquad t \text{ contains } X}{false} \qquad \frac{X \neq t \qquad t \text{ contains } X}{true}$$

$$\frac{p(t_1, \ldots, t_n) = p(s_1, \ldots, s_n)}{t_1 = s1, \ldots, t_n = s_n} \qquad \frac{p(t_1, \ldots, t_n) \neq p(s_1, \ldots, s_n)}{t_1 \neq s1 \vee \cdots \vee t_n \neq s_n}$$

$$\frac{p(t_1, \ldots, t_n) = q(s_1, \ldots, s_m) \text{ where } p \neq q \vee n \neq m}{false}$$

$$\frac{p(t_1, \ldots, t_n) \neq q(s_1, \ldots, s_m) \text{ where } p \neq q \vee n \neq m}{true}$$

where the substitution θ is added to the child node. In case two variables with different quantifiers are unified, the new variable is existentially quantified.

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

logical simplifications	$\dfrac{true \rightarrow A}{A}$	(3)
	$\dfrac{false \rightarrow A}{true}$ $\quad$ $\dfrac{true \wedge A}{A}$	
	$\dfrac{false \wedge A}{false}$ $\quad$ $\dfrac{true \vee A}{true}$	
	$\dfrac{false \vee A}{A}$	
factoring	$\dfrac{\mathtt{a}(X) \quad \mathtt{a}(Y)}{X = Y}$ $\quad$ $\dfrac{\mathtt{a}(X) \quad \mathtt{a}(Y)}{X \neq Y}$	

IFF^{Prob} operational semantics. We provide an extended version of the IFF operational semantics capable to deal with probabilistic integrity constraints. We decorate each probabilistic integrity constraint ic with its original version and a substitution, initially empty, $\theta_\emptyset$:

$$ic^{ic\theta_\emptyset}.$$

The main transition that needs to be updated is the logical equivalence $\frac{\text{true} \rightarrow A}{A}$ (Eq 3). Such a rule continues to exist for nonprobabilistic ICs, while for probabilistic ones it is replaced by the following two rules:

$$\frac{(p :: true \rightarrow A)^{ic_i\theta_j} \qquad E \qquad (ic_i, \theta_j, 0) \notin E}{A, \qquad E \cup \{(ic_i, \theta_j, 1)\}}, \tag{4}$$

$$\frac{(p :: true \rightarrow A)^{ic_i\theta_j} \qquad E \qquad (ic_i, \theta_j, 0) \notin E}{E \cup \{(ic_i, \theta_j, 0)\})}. \tag{5}$$

Intuitively, when the body of a PIC is proven true, its consequences are not propagated in all possible explanations, but, instead, two alternative explanations are generated: one in which the PIC is assumed to hold and its consequences are propagated, while in the other the PIC is not assumed to hold and the consequence A is not derived.

The other transitions are unmodified, except for the decoration explained earlier, for example, probabilistic propagation becomes

$$\frac{a(X) \qquad (p :: a(Y), B \rightarrow H)^{ic\theta}}{(p :: X = Y, B \rightarrow H)^{ic\theta}}.$$

Definition 2 (*Successful derivation*)
Each node of a derivation has the form

$$\langle R, \Delta, PSIC, E, \theta \rangle,$$

where the goal still to be proven is partitioned into the set R (Resolvent) of nonabducible literals and the set Δ of abducible ones, $PSIC$ is a set of (probabilistic and nonprobabilistic) ICs that must hold, E is the current explanation, and θ is the current substitution.

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

A *derivation is successful* for a goal G in an $ALP = \langle \mathbf{KB}, \mathbf{IC}, \mathcal{A} \rangle$ with explanation E if it starts from a node

$$\langle G, \emptyset, IC, \emptyset, \emptyset \rangle,$$

and terminates in a node

$$\langle \emptyset, \Delta, PSIC, E, \theta \rangle,$$

where no transition is applicable. If such a derivation exists, we write

$$ALP \vdash_E^{\Delta} G\theta.$$

Definition 3 (*Computed Probability of a goal*)
Let $\mathcal{E} = \{E | ALP \vdash_E^{\Delta} G\}$ be the *covering* pairwise incompatible set of explanations for the goal G, the computed probability of G is $P(G) = P(\mathcal{E}) = \sum_{E \in \mathcal{E}} P(E)$.

5 Soundness and completeness

Theorem 1 (*Soundness*)
If there exists a successful derivation $ALP \vdash_E^{\Delta} G\theta$, then $ALP \models_E^{\Delta} G\theta$.

Proof
Note that, with respect to the original IFF transitions, the new transitions of IFF^{Prob} only change the explanations, while all remaining elements of each node of the proof tree remain as in IFF. In particular, all nonprobabilistic integrity constraints are handled as in IFF, that is sound (Fung and Kowalski 1997), so for each successful derivation $ALP \vdash_E^{\Delta} G$, $KB \cup \Delta \models \mathbf{IC}^{np}$ holds.

The only modification is probabilistic logical equivalence (Eq 4 and Eq 5), in which a new branch is added: while in the original IFF proof-procedure, when the condition of an IC is satisfied, the consequence is always added to the node, for probabilistic ICs two mutually-exclusive branches are added to the proof-tree.

In one branch, Eq 4 is applied and adds the atomic choice $(ic, \theta, 1)$ to E. Note that no transition removes elements from E. In such a branch, the consequent of $ic\theta$ is added to the resolvent and all the following computation treats $ic\theta$ as a nonprobabilistic integrity constraint in IFF; this proves that for each $(ic, \theta, 1)$ that is added to E, $KB \cup \Delta \models ic\theta$.

In the other branch, Eq 5 is applied and the atomic choice $(ic, \theta, 0)$ is added to E. Notice that no transition can add an atomic choice (ic, θ, k) to E if $(ic, \theta, 0)$ was already in E (the set E is always consistent). This proves that $(ic, \theta, 1)$ will not be added in the following part of the derivation to E. □

Theorem 2 (*Completeness*)
If $ALP \models_E^{\Delta} G\theta$ then there exists a successful derivation $ALP \vdash_{E'}^{\Delta'} G\theta$, where $\Delta' \subseteq \Delta$ and $\forall (ic_i, \theta_j, 1) \in E'$, $(ic_i, \theta_j, 1) \in E$.

Proof
Suppose that $ALP \models_{E_*}^{\Delta} G$, that is, there is a probabilistic abductive answer for a goal G, in an explanation E_*. We leverage on the completeness theorem of the IFF proof procedure (Fung and Kowalski 1997), crafting an IFF program in such a way that a

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

IFF^{Prob} derivation can be built with simple replacements of transitions from the IFF derivation.

Let $IC(E^+_*) = \{ic_i\theta_j | (ic_i, \theta_j, 1) \in E_*\}$; consider the (nonprobabilistic)

$$ALP_{\not p} = \langle \mathbf{KB}, (IC(E^+_*) \cup \top(\mathbf{IC}^p \setminus IC(E^+_*)))|_{\not p} \cup \mathbf{IC}^{np}, \mathcal{A} \rangle,$$

where $X|_{\not p} = \{ic | p :: ic \in X\}$ strips the probability annotation from a set of probabilistic ICs X and $\top(X) = \{Body \rightarrow true | Body \rightarrow Head \in X\}$ replaces the $Head$ of a set of ICs with $true$; clearly the last operation produces implications that are tautologies, and are useless from a logical viewpoint, but that help us build the IFF^{Prob} derivation.

From the completeness theorem of the IFF (Fung and Kowalski 1997), we have that there exists an IFF derivation $ALP_{\not p} \vdash^{\Delta'} G$ where $\Delta' \subseteq \Delta$. We build a successful IFF^{Prob} derivation that mimics the IFF derivation, where Eq 4 is applied to ICs in $IC(E^+_*)$, and in which Eq 5 is applied to ICs in its complement, $\mathbf{IC}^p \setminus IC(E^+_*)$.

In the IFF derivation, the logical equivalence Eq 3 is possibly applied to $\top(\mathbf{IC}^p \setminus IC(E^+_*))|_{\not p}$; we substitute each application with the application of Eq 5. Such applications have no other consequences beside adding elements $(ic_i, \theta_j, 0)$ to the E set, where $ic_i\theta_j \in \mathbf{IC}^p \setminus IC(E^+_*)$.

Since IFF^{Prob} may also have to apply propagation, unfolding and case analysis to the ICs in $\mathbf{IC}^p \setminus IC(E^+_*)$, adding the set $\top(\mathbf{IC}^p \setminus IC(E^+_*))|_{\not p}$ to the integrity constraints of the IFF program ensures that also the IFF derivation applies such transitions when necessary.

In the IFF derivation, the logical equivalence $\frac{true \rightarrow A}{A}$ is (possibly) applied to ICs in $IC(E^+_*)|_{\not p}$. We substitute each application of Eq 3 with the application of Eq 4. Note that Eq 4 is applicable, because the set E is empty in the initial node, and the only transition that adds atomic choices of the type $(ic_i, \theta_j, 0)$ to E is Eq 5, that adds only elements of $\mathbf{IC}^p \setminus IC(E^+_*)$. Clearly the result in the resolvent is the same as for the original IFF derivation, since both Eq 3 and Eq 4 add the consequence A of the implication to the resolvent. By construction, the set of atomic choices $(ic_i, \theta_j, 1)$ added in this way to E is always a subset of $IC(E^+_*)$. □

6 Implementation

6.1 Recap: CHR implementation of the SCIFF proof-procedure

We implemented IFF^{Prob} leveraging on the implementation of the $\mathcal{S}$CIFF proof procedure. $\mathcal{S}$CIFF (Alberti *et al.* 2008) is an extension of the IFF proof procedure (Fung and Kowalski 1997) that also features constraints (*à la* Constraint Logic Programming (CLP)) and universally quantified abducibles. In this work, we extend to the probabilistic case only the IFF sub-language and leave for future work the probabilistic extension of the other parts of $\mathcal{S}$CIFF.

The $\mathcal{S}$CIFF proof-procedure was implemented (Alberti *et al.* 2013) in Constraint Handling Rules (CHR) (Frühwirth 2020). CHR is a rewriting system originally developed to implement new CLP constraint solvers, and then employed also as a language for a wide variety of applications. A set of atoms are declared as CHR constraints; according to the CLP operational semantics, when a CHR constraint is selected (e.g. during SLD-resolution), it is moved to a constraint store. CHR rules transform the constraint

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

store, hopefully simplifying it. There exist two main types of CHR rules: *propagation* and *simplification* rules.

A *Propagation* rule has the form

$$c_1, c_2, \ldots, c_n \Rightarrow Guard | Body,$$

where $c_1, \ldots, c_n$ are CHR constraints, *Guard* and *Body* are Prolog goals. The meaning is that *Body* is a logical consequence of the conjunction $c_1 \wedge \cdots \wedge c_n$ provided that *Guard* is true. Operationally, when the set of constraints $c_1, \ldots, c_n$ are in the constraint store, the guard is evaluated; in case it is true, the *Body* is executed.

A *Simplification* rule has the form

$$c_1, c_2, \ldots, c_n \Leftrightarrow Guard | Body.$$

Its declarative reading is that *Body* is equivalent to the conjunction $c_1 \wedge \cdots \wedge c_n$ provided that *Guard* is true. Operationally, when the set of constraints $c_1, \ldots, c_n$ is in the store, and the guard is true, the constraints $c_1, \ldots, c_n$ are removed from the store and the *Body* is executed.

In the $\mathcal{S}$CIFF implementation, each abducible atom `a(X)` is mapped to a CHR constraint `abd(a(X))`, and each IC $Body \rightarrow Head$ is mapped to a CHR constraint `ic(Body,Head)`. Each transition in the operational semantics is mapped to a CHR rule. For example, transition *propagation* is mapped to the CHR simplification rule (CHR expert readers will actually recognize it as a simpagation rule)

$$abd(A),\ ic([abd(B)|T], H)\ \Leftrightarrow\ abd(A),\ ic([A = B|T], H),$$

and transition *case analysis* is mapped to

$$ic([A = B|T], H)\ \Leftrightarrow\ A = B,\ ic(T, H)\ ;\ A \neq B,$$

where the semicolon is Prolog's OR and $\neq$ is a dis-unification constraint. Logical equivalence $\frac{true \rightarrow G}{G}$ is mapped to the simplification rule

$$ic([], Head) \Leftrightarrow Head. \tag{6}$$

6.2 Implementation of IFF^{Prob}

In order to implement probabilistic reasoning, we add a new CHR constraint that represents the current explanation:

$$expl(E, P),$$

means that, in the current derivation branch, the explanation is E, and has probability P. The E parameter is a collection (e.g. list) of triples (ic_i, θ_j, k), holding the integrity constraint ic_i, the substitution that made its body true, θ_j, and the integer k representing a Boolean value explaining whether $ic_i\theta_j$ belongs to E or not.

The CHR constraint representing ICs now requires additional information for probabilistic ICs: unsurprisingly the probability P needs to be added to the parameters, and also the original version ic_i of the integrity constraint together with the θ_j substitution that binds the variables in the body is stored. The new CHR constraint representing a probabilistic implication is

$$ic(Body, Head, ic_i\theta_j, P).$$

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

```
1: ic([],Head,ic_i θ_j,P_ic), expl(E,P_E) ⇔
2: (   (ic_i,θ_j,_) ∈ E,
3:     expl(E,P_E)
4: ;   (ic_i,θ_j,_) ∉ E,
5:     (   expl(E ∪ {(ic_i,θ_j,1)},P_E P_ic),
6:         Head
7:     ;   expl(E ∪ {(ic_i,θ_j,0)},P_E(1−P_ic))
8:     )
9: ).
```

Fig. 1. Implementation of the probabilistic logical equivalence rule in CHR.

We extend the logical equivalence rule $\frac{true \rightarrow A}{A}$ according to the new operational semantics (Figure 1). Such logical equivalence is applied when (due to successive applications of other transitions) the body of an integrity constraint ic_i is proven true, for a given substitution θ_j of the variables in the body. Simplification rule 6 is extended to consider three cases in disjunction. In the first (line 2), it is imposed that $ic_i\theta_j$ belongs to the current explanation (i.e. a unification is imposed such that $ic_i\theta_j$ unifies with at least one member of the explanation, similarly to the `member` predicate in standard Prolog); in such a case, the ic has already been propagated with exactly the same substitution, so there is no need to re-propagate its consequences, nor to change the current explanation and its probability. Otherwise (line 4), it is imposed that $ic_i\theta_j$ does not belong to the current explanation (i.e. a dis-unification constraint is imposed between $ic_i\theta_j$ and all members of the explanation). In such a case, two alternative branches are opened, that is, one in which we consider an explanation that includes the integrity constraint $ic_i\theta_j$ (line 5) and one in which $ic_i\theta_j$ is considered removed (line 7).

6.3 Computation of the probability of goals

The IFF^{Prob} proof procedure returns the covering set $\mathcal{E}$ of all the explanations $expl/2$ for a goal G. As seen in Definition 3, if $\mathcal{E}$ is also pairwise incompatible, then $P(G) = P(\mathcal{E})$. However, a covering set of explanations $\mathcal{E}$ for a goal G is not guaranteed to be pairwise incompatible.

Thus, we associate a Boolean random variable X_{ij} to each instance of PIC $ic_i\theta_j$. In this way, an atomic choice $(ic_i, \theta_j, 1)$ corresponds to X_{ij} assuming value $true$. The variables $\mathbf{X} = \{X_{ij} | (ic_i, \theta_j, k) \in E, E \in \mathcal{E}\}$ are pairwise independent and the probability that X_{ij} takes value 1 is p_i, the probability associated with the ith PIC.

Given a covering set of explanations $\mathcal{E}$ for a query G, each world where the query is true corresponds to an assignment of $\mathbf{X}$ for which the following Boolean function takes value 1:

$$f_{\mathcal{E}}(\mathbf{X}) = \bigvee_{E \in \mathcal{E}} \bigwedge_{(ic_i,\theta_j,1) \in E} X_{ij} \bigwedge_{(ic_i,\theta_j,0) \in E} \overline{X_{ij}}. \tag{7}$$

Thus, we can compute the probability of G by computing the probability that $f_{\mathcal{E}}(\mathbf{X})$ takes value 1. This formula is in Disjunctive Normal Form (DNF) but we cannot compute $P(f_{\mathcal{E}}(\mathbf{X}))$ by summing the probability of each individual explanation because the different explanations may not be mutually disjoint. To solve the problem, we can apply knowledge compilation to the propositional formula $f_{\mathcal{E}}(\mathbf{X})$ (Darwiche and Marquis 2002) in order

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

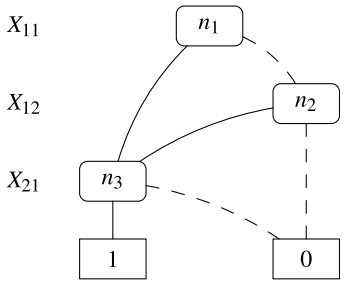


Fig. 2. BDD for function $f(\mathbf{X})$.

```
1: function PROB(node, pMap)
2:     Input: a BDD node node and a (node,prob) map pMap
3:     Output: the probability of the formula associated to node
4:     if node is a terminal then
5:         return value(node)                    ▷ value(node) is 0 or 1
6:     else if node is in pMap then
7:         return pMap[node]                     ▷ The probability of node
8:     else
9:         let X be v(node)    ▷ v(node) is the variable associated to node
10:        P1 ←PROB(child1(node))
11:        P0 ←PROB(child0(node))
12:        add (node, P(X)·P1 + (1−P(X))·P0) to pMap
13:        return P(X)·P1 + (1−P(X))·P0
14:    end if
15: end function
```

Fig. 3. Function that computes the probability of a BDD.

to translate it into a target language that allows the computation of the probability in polynomial time. A target language that was found to give good performances is the one of Boolean Decision Diagrams (BDDs).

A BDD for a function of Boolean variables is a rooted graph that has one level for each Boolean variable. A node n in a BDD has two children: one corresponding to the 1 value of the variable associated with n, indicated with $child_1(n)$, and one corresponding to the 0 value of the variable, indicated with $child_0(n)$. The leaves store either 0 or 1. Given values for all the variables, a BDD can be used to compute the value of the formula by traversing the graph starting from the root, following the edges corresponding to the variables values and returning the value associated to the leaf that is reached. For instance, Figure 2 shows a BDD for the function $f(\mathbf{X}) = (X_{11} \wedge X_{21}) \vee (X_{12} \wedge X_{21})$. A BDD performs a Shannon expansion of the Boolean formula $f_{\mathcal{E}}(\mathbf{X})$, so that if X is the variable associated to the root level of a BDD, the formula $f_{\mathcal{E}}(\mathbf{X})$ can be represented as $f_{\mathcal{E}}(\mathbf{X}) = X \wedge f_{\mathcal{E}}^{X}(\mathbf{X}) \vee \overline{X} \wedge f_{\mathcal{E}}^{\overline{X}}(\mathbf{X})$ where $f_{\mathcal{E}}^{X}(\mathbf{X})$ (respectively, $f_{\mathcal{E}}^{\overline{X}}(\mathbf{X})$) is the formula obtained by $f_{\mathcal{E}}(\mathbf{X})$ by setting X to 1 (resp., 0). Now the two disjuncts are mutually exclusive and the probability of $f_{\mathcal{E}}(\mathbf{X})$ can be computed as $P(f_{\mathcal{E}}(\mathbf{X})) = P(X)P(f_{\mathcal{E}}^{X}(\mathbf{X})) + (1 - P(X))P(f_{\mathcal{E}}^{\overline{X}}(\mathbf{X}))$. Figure 3 shows the function PROB that implements the dynamic programming algorithm of De Raedt *et al.* (2007) for computing the probability of a formula encoded as a BDD.

7 Application examples

We can now show how probabilities are computed in the running example; we then show a second example taken from the literature, to show the versatility of our approach.

Example 1 (*Continued*)
Here $\mathbf{IC}^{p} = \{ic_1'\}$, $\mathbf{IC}^{np} = \emptyset$ and the set of abducible predicates is $\mathcal{A} = \{\texttt{enter/2}, \texttt{killed/2}\}$, since they are not known.

Two atoms may make the body of ic_1' true, one stating that `husband` has the keys of `house1` and the other stating that he holds the keys of `house2`; these will correspond to the substitutions $\theta_1 = \{\texttt{P/husband}, \texttt{H/house1}\}$ and $\theta_2 = \{\texttt{P/husband}, \texttt{H/house2}\}$. For each instantiation that makes the body true, exactly one of the two transitions in Eq. 4 and 5 is applicable. In this way, four explanations are generated in alternative branches: $E_1 = \{(ic_1', \theta_1, 1), (ic_1', \theta_2, 1)\}$ with $P(E_1) = 0.7^2 = 0.49$, $E_2 = \{(ic_1', \theta_1, 1), (ic_1', \theta_2, 0)\}$

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

with $P(E_2) = 0.7 \cdot (1 - 0.7) = 0.21$, $E_3 = \{(ic'_1, \theta_1, 0), (ic'_1, \theta_2, 1)\}$ with $P(E_3) = (1 - 0.7) \cdot 0.7 = 0.21$ and $E_4 = \{(ic'_1, \{\texttt{P/X,H/house1}\}, 0), (ic'_1, \{\texttt{P/X,H/house2}\}, 0)\}$ with $P(E_4) = (1 - 0.7)^2 = 0.09$.

The sets of explanations where the goal is true are

$\mathcal{E}_1 = \{E_1, E_2, E_3\}$ with probability $P(\mathcal{E}_1) = \sum_{i=1}^{3} P(E_i) = 0.91$, probabilistic abductive answer $\Delta = \{\textit{enter}\texttt{(husband,house1)}, \textit{enter}\texttt{(husband,house2)}, \textit{killed}\texttt{(husband,woman)}\}$ and $\theta = \{\texttt{M/husband}\}$. This solution (the most likely) states that the husband was the killer with a chance of 91%. Explanation E_1 indicates that the goal is true (the husband was the killer) if the husband had entered both houses since he had the keys for both. The second and third explanations represent the fact that one instantiation of the PIC is not considered, so with probability 0.3 one can enter a house even if (s)he does not have the keys.

$\mathcal{E}_2 = \{E_4\}$ with $P(\mathcal{E}_2) = P(E_4) = (1 - 0.7)^2 = 0.09$; note that both instances of ic'_1 are relaxed ($k = 0$) so the complement of 0.7 must be used. In this case, the probabilistic abductive answer is $(\Delta, \theta) = (\{\textit{enter}\texttt{(M,house1)}, \textit{enter}\texttt{(M,house2)}, \textit{killed}\texttt{(M,woman)}\}, \emptyset)$. This solution (much less probable) states that some unknown person entered the two houses and committed the murder with a chance of 9%.

IFF^{Prob} provided two reasonable explanations. In particular, the second abductive explanation, obtained through nonground abductive reasoning, is that some person, unknown to the knowledge base, could have perpetrated the murder. It is interesting to know that this is exactly the explanation suggested by the husband during the trial: he pleaded not guilty and suggested that some other person could have entered his house, killed the woman, and carried the pillow to the second house. Although logically possible, such explanation was considered unlikely by the judges, that sentenced the husband guilty. Nevertheless, as a human brain was able to hypothesize the existence of an external person entering both houses and killing the victim, also the IFF^{Prob} proof-procedure was able to produce such hypothesis. And as a human brain judged such hypothesis unrealistic, also the proof-procedure assigned it a low probability. Of course, this is a simplification of all the evidence collected during the trial, that we cannot report here due to lack of space.

It is also interesting to vary the probability q associated with ic'_1 to see when $\mathcal{E}_2$ becomes the most probable explanation. Now the probabilities are: $P(E_1) = q^2$, $P(E_2) = q \cdot (1-q)$, $P(E_3) = q \cdot (1-q)$ and $P(E_4) = (1-q)^2$. In order to have $P(E_1) + P(E_2) + P(E_3) < P(E_4)$, it must be that $q < 1 - \frac{\sqrt{2}}{2} \simeq 0.293$.

Example 2

Christiansen (2008) proposes an example of power supply network diagnosis (here adapted to IFF^{Prob} syntax); a power plant `pp` provides electricity to a few villages $\texttt{v}_i$ through directed wires $\texttt{w}_i$. The network structure is described by a set of `edge/3` facts (please refer to the figure of the network in Section 7.1 in Christiansen (2008)):

```
edge(w1, pp, n1).    edge(w4, n3, v3).    edge(w7, n3, v2).
edge(w2, n1, n2).    edge(w5, n1, n4).    edge(w8, n4, v4).
edge(w3, n2, n3).    edge(w6, n2, v1).    edge(w9, n4, v5).
```

The fact that a given point (the power plant, a node or a village) in the network has no electricity is described in Christiansen (2008) by means of the `hasnopower/1` predicate.

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

```
hasnopower(pp) ← down(pp).
hasnopower(N2) ← edge(W,_,N2), down(W).
hasnopower(N2) ← edge(_,N1,N2), hasnopower(N1).
```

while the opposite situation is described by:

```
haspower(pp) ← up(pp).
haspower(N2) ← edge(W,N1,N2), up(W), haspower(N1).
```

The clauses and facts mentioned above represent the **KB** of the IFF^{Prob} program. In the proof-procedure by Christiansen (2008), probabilities are associated to abducible atoms, and he defines the abducibles *up*/1 with probabilities 0.9 (every instance of *up*(X) has the same probability) and *down*/1 with 0.1. As noted earlier, in IFF^{Prob} we can use PICs to associate probabilities to single abducibles, using the complementary probability:

ic_1 = `0.1::up(X) → false.`
ic_2 = `0.9::down(X) → false.`

Finally, the program in Christiansen (2008) also includes one IC to state that no node may be up and down at the same time:

ic_3 = `up(X) ∧ down(X) → false.`

Here $\mathbf{IC}^p = \{ic_1, ic_2\}$, $\mathbf{IC}^{np} = \{ic_3\}$ and the set of abducible predicates is $\mathcal{A} = \{$*up*/1, *down*/1$\}$.

Given the goal:

G = `hasnopower(v1),hasnopower(v2),hasnopower(v3),`
`hasnopower(v4),hasnopower(v5)`

$P(G) = 0.199695$ is returned as the probability that no village has electricity, as the sum of the probability of the worlds where the goal is true, throughout the application of function Prob of Fig. 3. In total, 1600 worlds are found, corresponding to the different combinations of failures of wires/power plant. As also computed by Christiansen (2008), the two most probable worlds have both probability 0.1 and are, respectively, identified by the explanations:

$E_1 = \{(ic_2, \texttt{X/pp}, 0)\}$ having probability $P(E_1) = (1 - 0.9) = 0.1$; the complement of 0.9 is taken as ic_2 with substitution $\theta_1 = \{\texttt{X/pp}\}$ (i.e. *down*`(pp) → false.`) is not included ($k = 0$). The probabilistic abductive answer is $(\Delta, \theta) = (\{$*down*`(pp)`$\}, \theta_1)$;
$E_2 = \{(ic_2, \texttt{X/w1}, 0)\}$ having probability $P(E_2) = (1 - 0.9) = 0.1$; here ic_2 with substitution $\theta_2 = \{\texttt{X/w1}\}$ is not included. The probabilistic abductive answer is $(\Delta, \theta) = (\{$*down*`(w1)`$\}, \theta_2)$.

E_1 indicates that the goal is true (no village receives power) if the power plant is down with probability 0.1, while E_2 if the main wire (`w1`) is down with probability 0.1.

8 Experiments

To test how our approach reacts with an increasing number of worlds, given an integer $n \geq 1$, we considered the ALP containing the following ICs for $1 \leq i \leq n$:

$0.6 :: b_{i-1}(X) \rightarrow p_i(X) \wedge q_i(X).$
$0.6 :: p_i(X) \rightarrow b_i(X).$
$0.6 :: q_i(X) \rightarrow b_i(X).$

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

Table 1. *Average running time (in seconds) on 5 executions of the IFF^{Prob} proof-procedure and the ± the standard deviation. ALP size indicates the value n takes in each ALP, that is, it defines the range $1 \leq i \leq n$. OM means Out of Memory.*

ALP size (n)	N. Worlds	Time (s) ± Std. dev.	ALP size (n)	N. Worlds	Time (s) ± Std. dev.
1	5	0.00133 ± 0.00003	7	4 373	1.91763 ± 0.01396
2	17	0.00387 ± 0.00001	8	13 121	6.32130 ± 0.02730
3	53	0.01228 ± 0.00003	9	39 365	21.51733 ± 0.06072
4	161	0.04042 ± 0.00013	10	118 097	73.78656 ± 0.29424
5	485	0.13582 ± 0.00059	11	354 293	333.90058 ± 0.78041
6	1 457	0.45903 ± 0.45903	12	1 062 881	OM

where $\mathtt{b}_i\mathtt{(X)}$, $\mathtt{p}_i\mathtt{(X)}$, and $\mathtt{q}_i\mathtt{(X)}$ are abducibles for all i. For the test purpose, we built 12 ALPs of increasing size containing the above ICs for i varying from 1 to n, with $n \in \{1, 2, 3, \ldots, 11, 12\}$.

These ALPs present a number of worlds that grows exponentially with n. This is due to the fact that $\mathtt{p}_{i+1}\mathtt{(X)}$ and $\mathtt{q}_{i+1}\mathtt{(X)}$ are needed to abduce $\mathtt{b}_i\mathtt{(X)}$. $\mathtt{p}_{i+1}\mathtt{(X)}$ and $\mathtt{q}_{i+1}\mathtt{(X)}$ can either both be true or one true while the other must be abduced or both must be abduced. In turn, to abduce $\mathtt{p}_{i+1}\mathtt{(X)}$ or $\mathtt{q}_{i+1}\mathtt{(X)}$, one must use $\mathtt{b}_{i+1}\mathtt{(X)}$, which may be already abduced or may possibly have to be abduced. These possible cases must be considered for all values assumed by i, creating an exponential number of possible ways to prove $\mathtt{b}_i\mathtt{(X)}$.

For the test, we compute the probability of the goal $G = \mathtt{b}_0\mathtt{(X)}$. To compute the running time, we ran the IFF^{Prob} proof procedure 5 times w.r.t. each ALP built and we computed the average running time with its standard deviation. All the tests have been performed on a Linux machine equipped with IBM© POWER9™ AC922 at 2.6(3.1) GHz, with 256 GB of RAM. We allowed a maximum of 100 GB for the Prolog stack, necessary to hold the worlds and the information about the choice points for the ALPs with $i \geq 11$.

Table 1 shows, for each ALP, the number of found worlds and the average running time in seconds to compute the probability of the goal ± the standard deviation. The value OM represents the fact that the proof-procedure went out of the available stack due to the high number of worlds and choice points to maintain in memory. As one can see, our approach is able to manage very high number of worlds within 1 minute.

9 Related work

Some works explicitly addressed probabilistic abductive reasoning: Turliuc *et al.* (2013) rank explanations in terms of probabilities and investigate the role of integrity constraints. They define a probability distribution over the truth values of each (ground) abducible, while we set probabilities on integrity constraints. SOLAR (Inoue *et al.* 2009) is a system for abductive inference that applies an expectation maximization (EM) algorithm for evaluating hypotheses obtained from the process of hypothesis generation. After generating all minimal explanations, the EM algorithm, working on BDDs, is used to assign probabilities to atoms in explanations. Finally, SOLAR computes the probability of each hypothesis to find the most probable one. EM is also used by Raghavan (2011), who considers Bayesian Logic Programs (BLPs), and by Kate and Mooney (2009),

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

who consider Markov Logic Networks (MLNs). EM is exploited to learn the parameters associated with the model.

Differently, Poole (1993) considers Bayesian networks but focuses on the definition of the language instead of the combination of abductive proof-procedures and statistical learning. Moreover, it imposes assumptions on the type of constraints in order to simplify the procedure.

Arvanitis *et al.* (2006) consider Stochastic Logic Programs (SLPs), where abductive reasoning is done by reversing deduction, that is, reversing the flow of the proof procedure. However, this may return wrong conclusions without imposing *ad hoc* constraints in the program.

Rotella and Ferilli (2013) define new types of probabilistic constraints to guide the search of explanations that are consistent with the constraints, by giving priority to explanations having higher probability to be true. However, all these approaches do not allow nonground abduction. We refer to De Raedt and Kersting (2008) for a description of BLPs, MLNs, and SLPs.

Abduction is also used in machine learning. For example in Muggleton *et al.* (2015), it is used to perform predicate invention and recursive generalizations with respect to a meta-interpreter. In this case, however, values for predicate variables rather than values for first order variables are abduced. More recently, Dai and Muggleton (2021) proposed to use abduction to infer constraints for learning problems but do not consider existential variables.

Christiansen (2008) also implements probabilistic nonground abduction in CHR; the main difference with our work is that in his work probabilities are associated with abducibles, while in our work they are associated with integrity constraints. The integrity constraints in his proof-procedure are more limited in syntax, since they can only accommodate abducible predicates and are in the form of denials, while in IFF they can include all types of atoms, and are in the form of implications, with disjunctions in the head. Due to the syntactic restrictions on integrity constraints, only a limited form of negation is possible, while in IFF sound negation can be applied to both abducible and defined atoms. Differently from our solution, Christiansen adopts a best-first search scheme, in which branches with higher probability are explored before the branches with lower probability; while the exact probability is only known at the end of the whole search, at each found solution a lower bound is obtained, and it gets more precise as new solutions are found. On the other hand, a best-first search has higher memory requirements than a depth-first search.

10 Conclusions

We presented a probabilistic ALP language able to perform abductive reasoning with variables, and probabilities attached to constraints. The need to have probabilistic integrity constraints comes from probabilistic reasoning with many real-life applications, and such integrity constraints may be learned from available data (Riguzzi *et al.* 2020). We showed two examples on different domains of abductive reasoning with probabilities in action, showing the usefulness of nonground abduction and that our language can also tackle problems with probabilities attached to abductibles. Soundness and completeness of the devised proof-procedure have been shown.

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

Future work concerns considering nonground probability, that is, variable probabilities attached to integrity constraints and its CHR implementation, as well as probabilistic clauses in the **KB**, and CLP constraints.

References

Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P. and Torroni, P. 2008. Verifiable agent interaction in ALP: The SCIFF framework. *ACM Transactions on Computational Logic 9,* 4, 29:1–29:43.

Alberti, M., Gavanelli, M. and Lamma, E. 2013. The CHR-based implementation of the SCIFF abductive system. *Fundamenta Informaticae 124,* 4, 365–381.

Arvanitis, A., Muggleton, S. H., Chen, J. and Watanabe, H. 2006. Abduction with stochastic logic programs based on a possible worlds semantics. In *Short Paper Proceedings of the 16th International Conference on Inductive Logic Programming (ILP 2006).*

Azzolini, D., Riguzzi, F. and Lamma, E. 2019. Studying transaction fees in the Bitcoin blockchain with probabilistic logic programming. *Information 10,* 11, 335.

Christiansen, H. 2008. Implementing probabilistic abductive logic programming with constraint handling rules. In *Constraint Handling Rules, Current Research Topics.* Lecture Notes in Computer Science, vol. 5388. Springer, 85–118.

Clark, K. L. 1978. Negation as failure. In *Logic and Data Bases.* Springer, 293–322.

Dai, W.-Z. and Muggleton, S. H. 2021. Abductive knowledge induction from raw data. In *Proceedings of the 35th Conference on Artificial Intelligence (IJCAI 2021).*

Darwiche, A. and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research 17*, 229–264.

De Raedt, L. and Kersting, K. 2008. Probabilistic inductive logic programming. In *Probabilistic Inductive Logic Programming - Theory and Applications.* Lecture Notes in Artificial Intelligence, vol. 4911. Springer, 1–27.

De Raedt, L., Kimmig, A. and Toivonen, H. 2007. ProbLog: A probabilistic Prolog and its application in link discovery. In *20th International Joint Conference on Artificial Intelligence (IJCAI 2007).* Vol. 7. AAAI Press/IJCAI, 2462–2467.

Frühwirth, T. W. 2020. Justifications in constraint handling rules for logical retraction in dynamic algorithms: Theory, implementations, and complexity. *Fundamenta Informaticae 173,* 4, 253–283.

Fung, T. H. and Kowalski, R. A. 1997. The IFF proof procedure for abductive logic programming. *Journal of Logic Programming 33,* 2, 151–165.

Inoue, K., Sato, T., Ishihata, M., Kameya, Y. and Nabeshima, H. 2009. Evaluating abductive hypotheses using an EM algorithm on BDDs. In *21st International Joint Conference on Artificial Intelligence (IJCAI 2009).* Morgan Kaufmann Publishers Inc., 810–815.

Kakas, A., Kowalski, R. and Toni, F. 1998. The role of abduction in logic programming. In *Handbook of Logic in Artificial Intelligence and Logic Programming.* Vol 5. Oxford University Press, 235–324.

Kate, R. J. and Mooney, R. J. 2009. Probabilistic abduction using Markov logic networks. In *IJCAI-09 Workshop on Plan, Activity, and Intent Recognition (PAIR 2009).*

Kunen, K. 1987. Negation in logic programming. *The Journal of Logic Programming 4,* 4, 289–308.

Muggleton, S. H., Lin, D. and Tamaddoni-Nezhad, A. 2015. Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. *Machine Learning 100,* 1, 49–73.

Nguembang Fadja, A. and Riguzzi, F. 2017. Probabilistic logic programming in action. In *Towards Integrative Machine Learning and Knowledge Extraction.* Lecture Notes in Computer Science, vol. 10344. Springer.

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press

Poole, D. 1993. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence 64,* 1, 81–129.

Raghavan, S. V. 2011. Bayesian abductive logic programs: A probabilistic logic for abductive reasoning. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*. IJCAI/AAAI, 2840–2841.

Riguzzi, F., Bellodi, E., Zese, R., Alberti, M. and Lamma, E. 2020. Probabilistic inductive constraint logic. *Machine Learning*, 1–32.

Rotella, F. and Ferilli, S. 2013. Probabilistic abductive logic programming using possible worlds. In *Proceedings of the 28th Italian Conference on Computational Logic, Catania, Italy, September 25-27, 2013*. CEUR Workshop Proceedings, vol. 1068. CEUR-WS.org, 131–145.

Sato, T. 1995. A statistical learning method for logic programs with distribution semantics. In *Proceedings of the Twelfth International Conference on Logic Programming, (ICLP 1995)*. MIT Press, 715–729.

Turliuc, C.-R., Maimari, N., Russo, A. and Broda, K. 2013. On minimality and integrity constraints in probabilistic abduction. In *Logic for Programming, Artificial Intelligence, and Reasoning*. Springer, 759–775.

Vennekens, J., Verbaeten, S. and Bruynooghe, M. 2004. Logic programs with annotated disjunctions. In *24th International Conference on Logic Programming (ICLP 2004)*. Lecture Notes in Computer Science, vol. 3131. Springer, 431–445.

https://doi.org/10.1017/S1471068421000417 Published online by Cambridge University Press