

Relational Probabilistic Inference Over Unknown Objects

Nicholas L. Cassimatis

Arthi Murugesan

Perrin G. Bignoli

Rensselaer Polytechnic Institute

110 8th Street

Troy, NY 12180, USA

CASSIN@RPI.EDU

MURUGA@RPI.EDU

BIGNOP@RPI.EDU

Abstract

Combinations of logical and probabilistic methods have proven in many cases to enable compact representations and efficient inference with relational probabilistic theories. Theories involving objects that are not known before inference often entail infinite models and thus lead to difficulties in finding and defining a probability distribution over those models. We propose an approach for addressing these problems using the GenProb language for expressing relational probabilistic theories that can license the inference of initially unknown objects. A semantics for GenProb theories is defined by mapping them to probabilistic graphical models. Even in many cases where these models contain infinite nodes, it is possible to define a probability distribution over them. In these cases, a translation from GenProb theories to weighted generative probabilistic inference (GenSAT) problems enables correct maximum a posteriori inference.

Keywords: relational probabilistic inference, infinite domains

1. Introduction

An important aspect of problems generally intelligent systems will need to be able to solve is inference in domains where not all objects are known in advance. Autonomous physically embodied systems, for example, generally do not have complete sensory access to their environment and thus cannot often know all the relevant objects before they begin reasoning about their surroundings. As another example, grammatical formalisms intended to capture natural language such as context-free grammars and unification grammars generate an infinite number of possible phrases and thus these cannot all be anticipated in advance of inference. Unknown objects, however, pose many difficulties for many modern inference methods because these require all objects to be known in advance.

In this paper, we describe an approach to dealing with these problems in the context of methods that combine first-order logic and probabilistic graphical models by propositionalizing relational theories. We adopt this approach in this paper because it has proven promising for compactly representing and making inferences in many domains (Pedro Domingos & Richardson, 2006).

Theories over potentially unknown objects pose two problems for approaches based on propositionalization. Since all objects cannot be known in advance a theory cannot be fully propositionalized before inference. Second, languages that refer to unknown objects often require models of infinite size. For example, function symbols with more than one argument in

(unsorted) first-order logic can require infinite models. Even in cases where propositionalizations are finite, they can be very large and make efficient inference difficult to achieve.

In this paper, we propose an approach to inference with relational theories over known and unknown objects. Like other approaches (P. Domingos, Kok, Poon, Richardson, & Singla, 2006), we exploit the fact that probabilistic inference can be framed as a weighted satisfiability problem. However, our approach does not convert relational theories into *propositional* satisfiability problems for the reasons mentioned. Instead, we define a language for expressing generative relational probabilistic (GenProb) theories over unknown objects and provide a translation from these theories to first-order weighted generative satisfiability (GenSAT) (Cassimatis, Murugesan, & Bignoli, 2009) theories. Further, by associating a GenProb theory with a potentially infinite probabilistic graphical model, we define a probability distribution over models of GenProb theories. The cost of the best model of a GenSAT translation corresponds to the cost of the most likely model of the GenProb problem. This equivalence permits finite maximum a posteriori inference in many cases for GenProb theories with infinite models.

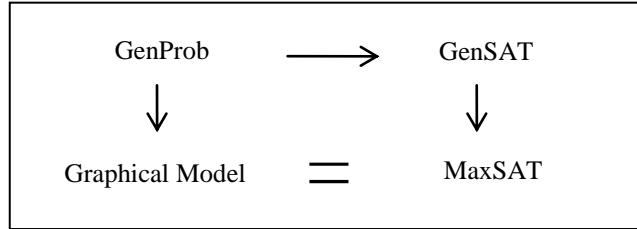


Figure 1. The relation between GenSAT theories, their translation into GenSAT and models of both.

2. Generative Probabilistic Theories

GenProb is a language for expressing probabilistic relational theories. The following highly simplified example theory of an airplane radar detector illustrates GenProb.

“Any particular plane has a 1% chance of being within range of a radar station. The radar display generates blips that indicate a strong possibility of a plane being detected and blips that indicate a weak possibility. Strong blips are only caused by planes, whereas weak blips can be caused by noise .1% of the time. Planes being tracked are fitted with collision warning systems that, in the presence of other planes in range, have a 90% chance of sounding an alarm that is transmitted to the radar station.”

The following formulae indicate the priors on a particular plane being in range and on noise:

$$\begin{aligned} True() &\rightarrow (.01) InRange(?p) \wedge Plane(?p) \\ True() &\rightarrow (.001) WeakBlip(?b) \end{aligned}$$

The causal aspects of the theory are indicated with clauses:

$Plane(?p) \wedge InRange(?p) \rightarrow$

$(.3)StrongBlip(?b), (.5)WeakBlip(?b), (.2) NoBlip(?b), ?p$

“Of the cases where a plane is range, 30% of them will generate a strong blip, 50% of them will generate a weak blip and in 20% no blip will be generated.”

$Detect(?p1, ?p2) \wedge Plane(?p1) \wedge Plane(?p2) \rightarrow (.9) TransmitAlarm(?p1), ?p1, ?p2$

“In 90% of the cases where one plane detects another, an alarm will be transmitted.”

The occurrence of $?p$, $?p1$ and $?p2$ after the final comma indicated that a blip licenses the existence of a plane to be inferred. The other variables are implicitly universally quantified. This will be made more precise below. The alarm detection system can be indicated thus:

$$TransmitAlarm(?p) \rightarrow AlarmSound()$$

Clauses with numbers are called causal clauses and those without number are called logical clauses. Logical clauses are hard constraints. Since blips occur in the consequent of causal clauses, they must be the effect of one of these clauses. In this case, strong blips can only be caused by planes, while weak blips can be caused by planes and by noise. Such mandatory causation is not implied by logical clauses. Mandatory causation for literals can be neutralized with a causal clause whose antecedent is $True()$, which (see below) is always true.

More formally, a GenProb theory is a set of causal and logical clauses. Causal clauses are of the form $C_1 \wedge \dots \wedge C_n \rightarrow (p_1)E_1 \dots (p_m)E_m, \dots ?v_i, \dots$, where $0 \leq p_i \leq 1$ and where the p_i sum to 1, each of the C_i are either literals or negations thereof, and the E_i conjunctions of literals. Each E_i conjunction is called an *effect* of the clause and each v_i is called a *posited variable*. Non-posited variables are implicitly universally quantified. Literals are predicates with arguments that are terms. Terms that are not variables are called “objects”. Logical clauses are of the form $A1 \wedge \dots \wedge A_m \rightarrow B1 \wedge \dots \wedge B_n$, where each conjunct is either a literal or a negation thereof. Literal a is a grounding of literal b if it is a ground literal and equivalent under an assignment of variables in b to objects.

3. Semantics

There are two components to defining a semantics for a GenProb theory. First, we propose a translation of a GenProb theory into a set of clauses with only propositional literals. Identifying these literals is complicated by the fact that these theories can range over objects, potentially infinite in number, for which there is no constant in the theory. A model of a theory is an assignment of truth values to these literals. Second, in order to define a probability distribution over models of a theory, we associate a dual graph with each theory. The nodes of the dual graph correspond to literals in the models of the theory. The graph is directed and its edges encode probabilistic independence in a manner similar to Bayesian Networks. However, a dual graph can have infinite numbers of nodes and each node can be involved in infinite numbers of edges.

3.1 Propositional translation of a GenProb theory

First, we define the set of *relevant* propositional literals for a theory. A literal is relevant for a theory if it occurs in a clause in the grounding of a theory. A clause is in the grounding of a

theory if it can be derived by the repeated grounding of objects in the theory and of objects generated in previous grounding steps.

The formal definition of the grounding of a GenProb theory depends on a skolem function s that uniquely maps objects (that are bound to variables) to objects generated by this binding. In the example, from the last section, blip b , will bind to $?b$ and license the inference of the existence of a plane corresponding to b . Technically, a skolem function maps a (potentially partial) assignment of objects to variables in a GenProb theory, to an object. In our example, for any assignment a , where $?b$ is assigned to b , $s(a, ?p) = p$. It will be convenient to abbreviate this $s_{(?b,b)}(?p) = p$. This can be read, “when b is assigned to $?b$, p is assigned to $?p$ ”. Even more colloquially and specifically to this example, we can say that, “ p is the plane posited to explain b ”. As this example illustrates, the purpose of this function is to produce objects that are inferred during inference and that thus do not occur in P.

We illustrate translation with the GenSAT theory with the following four clauses

$$\begin{aligned} & \text{Hammer}(?h) \wedge \text{Bell}(?b) \wedge \text{Hit}(?h, ?b) \rightarrow (.99)\text{Ring}(?b) \\ & \text{Hammer}(h) \\ & \text{Bell}(b) \\ & \text{Hit}(h, b) \end{aligned}$$

The translation of this theory grounds variables with constants occurring as arguments of literals in the theory.

$$\begin{aligned} & \text{Hammer}(h) \wedge \text{Bell}(b) \wedge \text{Hit}(h, b) \rightarrow (.99) \text{Ring}(b) \\ & \text{Hammer}(b) \wedge \text{Bell}(h) \wedge \text{Hit}(b, h) \rightarrow (.99) \text{Ring}(h) \\ & \text{Hit}(h, b) \\ & \text{Hammer}(h) \\ & \text{Bell}(b) \end{aligned}$$

Note that the second pair of clauses above are in some sense spurious since $\text{Hammer}(b)$ should intuitively be false given that $\text{Bell}(b)$ and that bells are not hammers. A clause indicating that hammers are not bells could prevent models with $\text{Hammer}(b)$ from being true. It is also possible, though beyond the scope of this article, to eliminate such spurious clauses by extending GenProb to be a sorted logic.

To illustrate a theory over unknown objects, consider adding a clause to the effect that ringing bells generates sounds:

$$\text{Ring}(?b) \rightarrow \text{GenerateSound}(?b, ?s), ?s$$

Adding this clause to the GenSAT theory above would add the following two clauses to its translation:

$$\begin{aligned} & \text{Ring}(b) \rightarrow \text{GenerateSound}(b, b\text{Sound}) \\ & \text{Ring}(h) \rightarrow \text{GenerateSound}(h, h\text{Sound}) \end{aligned}$$

These clauses include terms $b\text{Sound}$ and $h\text{Sound}$ which, informally, are the sounds generated by the ringing of b and of h : $s_{(?b,b)}(?s) = b\text{Sound}$, $s_{(?b,b)}(?s) = h\text{Sound}$. The following theory illustrates the possibility of infinite translations.

$$\begin{aligned}
&Mammal(a) \\
&Mammal(?m) \wedge GiveBirth(?m, ?x) \rightarrow Mammal(?x), ?m
\end{aligned}$$

Its translation includes infinitely many clauses:

$$\begin{aligned}
&Mammal(a) \\
&Mammal(aMother) \wedge GiveBrith(aMother, a) \rightarrow Mammal(a) \\
&Mammal(aMotherMother) \wedge GiveBirth(aMotherMother, aMother) \\
&\quad \rightarrow Mammal(aMother) \\
&\dots
\end{aligned}$$

We now describe the translation more precisely in terms of a function that translates a GenProb theory onto a set of grounded propositional clauses. It is quite similar to the propositional translation (Cassimatis et al., 2009) of GenSAT, a language for expressing relational constraints over unknown objects. The translation depends on an appropriate skolem function, s , as described above.

A clause is in $Translation(P, s)$ if it can be generated by successive steps of grounding:

$$\begin{aligned}
Translation(P, s) = \\
&OneStepGrounding(P, Objects(P), s) \cup \\
&OneStepGrounding(P, Objects(Translation(P, s)), s)
\end{aligned}$$

A clause is produced in a step of grounding if it can be produced by assigning to variables objects that are either generated by a skolem function or are already in the translation.

$$OneStepGrounding(P, objects, s) = \cup_c ClauseGrounding(c, objects, s).$$

The grounding of a clause with respect to a set of objects is the set of clauses that can be formed by all the possible assignments of those objects to variables in the clause. For example, the grounding of $A(?x) \rightarrow B(?y), ?x$, with respect to s and objects $\{a, b\}$ includes (only) the following two clauses:

$$\begin{aligned}
&A(s_{(?y,a)}(?x)) \rightarrow B(a) \\
&A(s_{(?y,b)}(b?x)) \rightarrow B(b).
\end{aligned}$$

Having defined the translation of a theory, we can use the propositional literals occurring in it to define a model for that theory. Specifically, a **model** of a GenProb theory, T , is an assignment of propositional literals in $Translation(T)$ to truth values that is consistent with the logical constraints in that translation. A model is consistent with the logical constraints in a translation if none of the logical clauses in the translation are violated by the standard interpretation of ordinary propositional material implication.

3.2 Dual Graphs

In order to define a probability distribution over models of a GenProb theory, we introduce the notion of a dual graph (DG). Dual graphs are directed graphical models whose nodes include the propositional literals in the translation of a theory and additional literals that will be described

below. The edges are related (as will be described in detail below) to clauses in the translation. Each node has a standard conditional probability distribution representing the probability of the literal being true given its parents. The graph encodes probabilistic independence in the standard manner. Since a translation can have an infinite number of literals and since each literal can potentially participate in an infinite number of clauses, the graph can have infinite numbers of nodes, with nodes potentially participating in an infinite number of edges.

DGs are defined by the merger of several subnetworks associated with the relevant literals in the translation of a theory. Some definitions will help characterize the DG. **Caused literals** are groundings of literals in the consequent of a ground causal clause. For literals $P_1(a_1, \dots), P_2(a_m, \dots), \dots$, the literal $And(P_1, a_1, \dots, P_2, a_m, \dots, \dots)$ is their conjunction literal. An **antecedent conjunction** literal is the conjunction literal formed from the antecedents of a clause. The **consequent conjunction** literals are those conjunction literals formed from each effect in the literal. For every possible effect in a ground clause, we create a literal that represents the conjunction of the antecedent. An **effect literal** has predicate *Cause* and arguments: *AND* followed by the arguments of the conjunction literal, followed by the predicate, followed by *AND*, followed by the arguments in the consequent conjunction literal. For example, the effect literal for $P(x) \rightarrow .6 Q(x)$ is $Cause(AND, P, x, AND, Q, x)$.

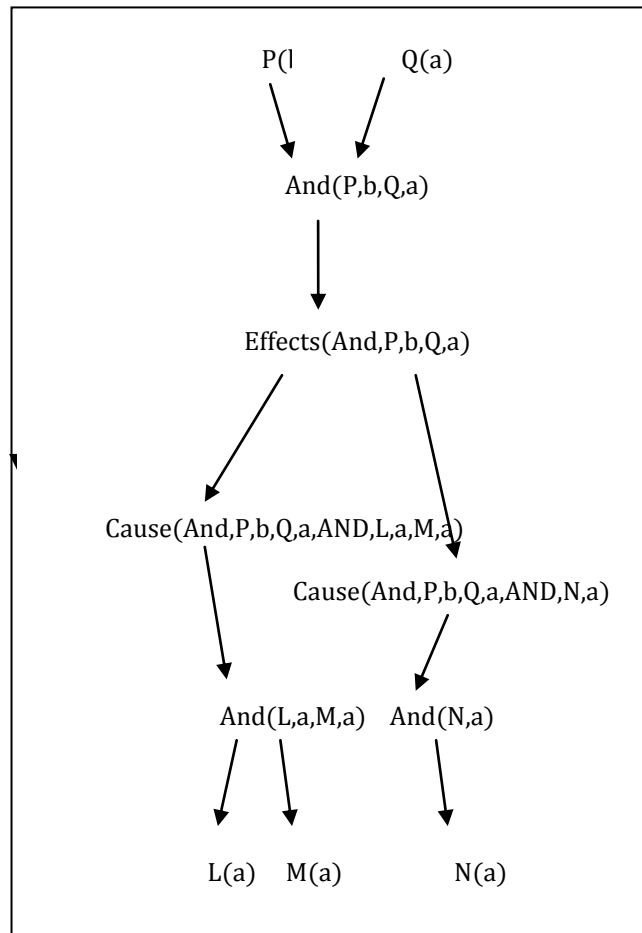


Figure 2. Subgraph of a dual graph associated with: $P(b) \wedge Q(a) \rightarrow (.7) L(a) \wedge M(a), (.3) N(a)$

Several kinds of subnetworks comprise a DG. For every clause with probabilities for effect p_i , there is a **ramification network**. This network includes an edge from the conjunction literal of the clause to a multi-valued *effects literal* node – this is the only kind of non-binary node in a DG – that indicates which effect results when the left-hand-side is true. The conditional probability table is $CPT(1) = p_i$ and $CPT(0) = 0$. An edge from this node is directed outward towards each effect literal for the clause. The CPT is 1 when the value corresponding to an effect is true and 0 otherwise. Figure 1 illustrates the ramification network and conjunction literals for:

$$P(b) \wedge Q(a) \rightarrow (.7) L(a) \wedge M(a), (.3) N(a)$$

Antecedent conjunction literals receive inputs from the individual literals in the conjunction. The CPT is 1 when all the literals are true and 0 otherwise. The consequent conjunction literals have edges directed outwards to the literals in the conjunction. The CPT for each of these is 1 when the consequent conjunction is true and 0 otherwise.

In the example, the CPT for $Effects(And,P,b,Q,a)$ is:

$And(P,b,Q,q)$	Effect 1	Effect 2
0	0	0
1	.7	.3

The CPT for $Cause(And,P,b,Q,a,N,a)$ is:

$Effects(And,P,b,Q,a)$	0	1
Effect 1	1	0
Effect 2	0	1

The DG of a theory P is the merger of the all the ramification networks for each literal relevant to P . The only shared literals among these graphs are the conjunction literals and the relevant literals to P . The antecedent conjunction literal CPTs are identical in each ramification network and in the merged graph. CPTs for the consequent ramification literals in the merged graph have entries of 1 when at least one of the effect literal inputs is 1, and 0 otherwise. CPTs for literals only have edges entering from conjunction literals and are 1 when one of these conjunction literals is true and 0 otherwise.

A model for a GenProb problem P is an assignment of truth values to the values of the literals in the DG of P that is consistent with the logical constraints in P . We define a probability distribution over the models of P by computing a sum over the probabilities of the individual models of P . The probability of a model of a GenProb problem involves the product of the probabilities of the nodes in P given their parents. As in Bayes Networks and as will be discussed below, this distribution requires that the DG of P contains no cycles for that probability to be well-defined. Since the language has no explicit provision for priors, a further requirement for being well-defined is that parentless edges have only one possible value.

A probability distribution over GenProb problems with models M that is consistent with logical constraints in M can be defined thus:

$$P(m) = \frac{f(m)}{\sum_{m_i \in M} f(m_i)}$$

where, for literals L in m and where m_l is a model m without an assignment for l :

$$f(m) = \begin{cases} \prod_{l_i \in L} P(l_i | m_l), & \text{when finitely many literals are true in } L \\ 0, & \text{otherwise} \end{cases}$$

For many infinite models, the product in f converges on zero in the limit. One class of cases where this is guaranteed to happen involves *decreasing probability theories*. In a decreasing probability theory, each object that is the argument of a literal of the DG is an argument of a literal whose probability given the rest of the DG is less than 1. Although some problems are not naturally cast as decreasing probability theories, e.g., arithmetic, there are many cases where each potential new object does lead to increased uncertainty. For example, in radar interpretation, each blip can be caused by a plane or by noise; in most gesture recognition situations, a person is never certain to produce a specific gesture; any probabilistic context-free grammar whose phrases can all be generated using more than a production rule is a decreasing probability theory.

In decreasing probability theories, therefore, it is natural to define $f(model) = 0$ when there are infinite numbers of literals assigned to true. $f(model)$ has finite nonzero addends because every CPT in the model is zero when all the parent nodes are zero. Since only finite numbers of nodes can have nonzero parents, there are only finite nonzero addends.

The resulting probability distribution is well defined in many, but not all, cases. As in Bayesian Networks, in order for f to be well-defined, the DG cannot contain directed cycles. When involved in such a cycle, the probability of a particular node value would be defined in terms of itself. DGs of GenProb theories contain no directed cycles in (but not only in) the following circumstance: no predicate appears in its own antecedent predicate closure. The antecedent closure for a predicate, A , contains the union of all predicates in the antecedent of clauses that have A in the consequent together with the predicate closure of those predicates. In other words, the antecedent closure for A contains all the predicates that can be involved in causal chains supporting A . If a literal is involved in a cycle, then its predicate will clearly be in its own antecedent predicate closure. Thus, when a predicate does not appear in its antecedent closure, there can be no cycles in the DG.

As mentioned earlier, the parentless nodes in the DG must have only one possible value. One way to accomplish this that proves useful in practice is to include a causal clause whose antecedent is $True()$ for each type of literal that is not in an effect of a causal clause.

Proposition 1. GenProb theories have a well-defined probability distribution if no predicate appears in its own antecedent closure and when for each type of literal l not in the effect of another clause there is a clause, $True() \rightarrow (p) l$, where $p > 0$.

Although we would like to find stronger conditions for a well-defined probability distribution – the conditions in Proposition 1 exclude some of our examples, for instance – some important classes of theories do meet Proposition 1’s conditions, however. In that absence of such conditions, however, we must rely on the fact that theories without directed cycles in their DG will admit well-defined probability distributions.

Proposition 2. GenProb theories have well-defined probability distributions if their DGs do not contain directed cycles and if for each type of literal l , not in the effect of another clause, there is a clause, $True() \rightarrow (p) l$, where $p > 0$.

This proposition will be useful in some specific, but important cases. Although we do not prove it here, for example, the GenProb translation (Murugesan & Cassimatis, 2009) of a probabilistic context-free grammar intuitively will have a DG without cycles because no specific phrase can generate itself. (Of course, categories of phrases can generate phrases with the same category as themselves, e.g., $NP \rightarrow NP + PP$).

Note that the above conditions on well-definedness pertain only to causal clauses. Logical clauses only serve to rule out models. For this reason and because problems with cyclicity generally do not obtain when all probabilities are zero or one, there is no need to impose constraints on the cyclicity of logical constraints in a GenProb theory in order for there to be a well-defined probability distribution over its models.

Approaches that involve probabilistic inference over infinite numbers of objects (Kersting & De Raedt, 2001; Milch, Marthi, Sontag, Russell, & Ong, 2005; Brian Milch et al., 2005; Muggleton, 1996; Richardson & Domingos, 2007), typically require that a node be directly connected with only finitely many other nodes in a graph. Infinite parents in GenProb DGs only occur for conjunction and literal nodes since all other nodes are defined to have only a fixed number of parents. These nodes have CPTs that are nonzero only when one of the parents is true and are thus well-defined even in the infinite case. In models where there are infinite parents that are true for a node, the probability of the model, by the definition of f , is zero.

The following is simple example of a theory where infinite parents for a node arise:

A parent with a particular gene, X, is likely to pass it on to its offspring and that some, but many fewer, instances of a gene occur by chance. Genes generate a symptom Y, which can also occur through contagion from others with that symptom. Finally, individual a , has the Y phenotype.

The following clauses formalize this theory in GenProb:

$Parent(?p, ?c) \wedge GeneX(?p) \rightarrow (.2) GeneX(?c), ?p$

“If $?p$ is the parent of $?c$ and $?p$ has Gene X, then there is a 20% chance $?p$ will have passed X to $?c$.”

$True() \rightarrow (.000001) GenX(?c)$

“In one out of a million cases, a person can have Gene X without any other cause, i.e., through mutation.”

$Y(?a) \rightarrow (.1) SpreadYTo(?a, ?p) \wedge Y(?p)$

“Someone with Y has a 10% chance of spreading it to someone else.”

$GeneX(?p) \rightarrow Y(?p)$
 “Gene X always generates Y”

$Y(a)$
 “a has Y”

The DG for this theory is infinite because one can acquire Y through one’s parent, who can acquire it from his parent and so on. However, the probability of each inheritance is less than one and thus the probability of a model with an infinite number of ancestors with the gene is zero. Models with nonzero probability are possible. For example, the model where $Y(a)$ and $GeneX(a)$ are true, and all other literals are false, has finite probability.

4. Inference

As in the case of some other probabilistic frameworks (P. Domingos et al., 2006), inference in GenProb theories can be formulated as a weighted satisfiability problem. However, since GenProb theories can have infinite models, they cannot be straightforwardly translated to ordinary MaxSAT problems. Instead, we must translate them to a weighted satisfiability framework that can deal with unknown objects. Specifically, in this section we show that a GenProb theory can be translated into a generative weighted satisfiability (GenSAT) problem such that cost of a model, m , for the GenSAT problem corresponds to $f(m)$.

GenSAT theories are relational weighted satisfiability problems. They are expressed in a language for encoding weighted and logical constraints that license the positing of new objects. Their syntax has many similarities with that of GenProb theories. The following simple example illustrates the main aspects of the language. It involves constraints that state that both a telephone (whose ringer is on) receiving a call and the striking of a bell lead to a ringing sound and that a phone ringer is only on when the phone’s battery is not empty.

$BellStrike(?x, ?t) \rightarrow (10) RingSound(?t), ?x$

$PhoneGetsCall(?p, ?t) \wedge RingerOn(?p, ?t) \rightarrow (7) RingSound(?t), ?p$

$RingerOn(?p, ?t) \rightarrow EmptyBattery(?p, ?t)$

The numbers “10” and “7” are the weights on the constraints represented by the clauses. Terms beginning with ‘?’ are variables. Variables interpreted after the comma are called *posited* variables and indicate the potential of an object to be posited. For example, ?x in the second constraint indicates that if a ring sound occurs, then a bell (as of yet unknown) being struck may have caused it. Non-posited variables are implicitly universally quantified. Clauses with weights are causal clauses and those without are hard constraints.

Each GenSAT theory can be translated into a potentially infinite set of weighted MaxSAT constraints. The atoms over which these constraints hold includes the set of relevant literals for the GenProb. A GenSAT model is an assignment of truth values to these literals. Each model has a cost, which is the sum of the cost of all the ground constraints that are broken by it.

The MaxSAT translation of GenSAT models imposes a “mandatory causation” constraint on literals. Literals that occur in the consequent of a causal clause must be “caused” by some causal clause constrained to be true. Thus, if $P(a) \rightarrow 10 R(a)$ and $Q(a) \rightarrow 15 R(a)$ are the only two causal clauses with $R(a)$ in the antecedent, any model where $R(a)$ is true must have one of $P(a)$ or $Q(a)$ being true.

Our approach is to translate a GenProb problem into a GenSAT problem such that probability of a GenProb model can be discerned from the cost of the corresponding GenSAT model. The minimal-cost GenSAT model corresponds to the most likely model of the GenProb problem. Since at least one complete algorithm (Cassimatis et al., 2009) exists for GenSAT problems, this mapping therefore provides a method of inference for GenProb theories. We associate a GenProb theory P with a GenSAT theory S so that the total cost of the constraints broken in the best model of S is equivalent to the negative log of the probability of the best model of P . Since the equation for the probability of the best model of P is a product, we use its log so that the resulting sums can be more easily associated the addends making up the sum of the broken constraints in models of S . Specifically:

$$\ln f(m) = \sum_{l \in L} \ln P(l|m_l)$$

Maximizing this quantity is equivalent to minimizing the following quantity:

$$-\ln f(m) = -\sum_{l \in L} \ln(P(l|m_l))$$

Note that each addend corresponds to a literal in the DG. The translation creates GenSAT constraints whose cost equals each addend. For example, if $P(l|m_l) = .75$ in a DG, then a constraint is broken in the corresponding model for the GenSAT translation whose cost is $-\ln .75$. Let $C(i,o)$ be the cost of the constraints broken in a model given the values of i and o , where there is only one edge leading into o , that edge is i and whose CPT is p_v for possible value v if i is true and 0 otherwise. This is the kind of CPT for the effect literal nodes in a ramification network. C must generate the following costs:

$$\begin{aligned} C(1,0) &= -\ln P(i = 1|o = 1) = -\ln p_v \\ C(1,0) &= -\ln P(i = 0|o = 1) = -\ln(1 - p_v) \\ C(0,1) &= -\ln P(i = 1|o = 0) = -\ln 0 = \infty \\ C(0,0) &= -\ln P(i = 0|o = 0) = -\ln 1 = 0 \end{aligned}$$

For clauses with antecedent A , consequent effects, C_i and a list of posited variables V , the following GenSAT constraints, where $\text{conj}(X)$ is the conjunction literal for literals X and the $\text{effect}(A,C_i)$ are the effect literals for the clause, will lead to these costs:

$$\text{conj}(A) \rightarrow (\ln p) \neg \text{effect}(A, C_i), V$$

$$\text{conj}(A) \rightarrow (\ln p) \neg \text{effect}(A, C_i), V$$

$$\neg conj(A) \rightarrow \neg effect(A, C_i)$$

$$\neg effect(C_1) \wedge \neg effect(C_n) \rightarrow \neg conj(A)$$

Since the entries in CPTs for all other nodes are 1 or 0, they can be captured with straightforward logical constraints.

The translation of a GenProb theory into a GenSAT theory is the union of all the constraints capturing the conditional probability table of the nodes in the DG together with the constraints defining the conjunction and effect literals.

As in Bayes Networks, implicit in DGs is that an inner node’s value only occurs under or is “caused by” certain configurations of the values of the nodes directed towards it. This is automatically captured by GenSAT “mandatory causation”. For example, if A and B are the only two inputs to node C , there will be two causal clauses in the translation with C in the consequent. As in the DG, if A and B are false in a model of the GenSAT translation of a GenProb theory, then so must be C .

We have shown that for each literal l in the joint probability equation f , there will be a cost broken in the GenSAT translation of it that corresponds to the probability of l given its parents. Since these are the only constraints broken, the cost of a model, m_S , for a GenSAT translation of a GenSAT problem P , corresponds to $f(m)$, where m_P is a model of P .

Proposition 3. For model m_P of a GenSAT theory P and model m_S of its GenSAT translation S :
 $-\ln f(m_P) = cost(m_S)$.

This fact enables algorithms for finding GenSAT models to make inferences about theories expressed in GenProb. The GenDPLL algorithm (Cassimatis et al., 2009) finds minimal cost models for “increasing cost theories”. Increasing cost theories have models whose posited objects are guaranteed to participate in broken constraints that have net positive costs. Since each causal clause in a decreasing probability GenProb theory is translated into GenSAT constraints that are guaranteed to generate positive costs, GenDPLL will therefore identify solutions to translations of decreasing probability GenSAT problems.

5. Related Work

Several approaches to probabilistic inference deal with unknown objects and infinite domains. Infinite Markov Logic Networks (Richardson & Domingos, 2007) with well-defined probability distributions are possible and have fewer restrictions than GenProb models, though to our knowledge, no algorithm for inference over infinite MLNs has been published. Other approaches impose restrictions similar to those in GenProb theories. Bayesian Logic Programs (Kersting & De Raedt, 2001) require a finite number of ancestors. Stochastic logic programs (Muggleton, 1996) assign some infinite models zero probability. Contingent Bayes Networks (B. Milch et al., 2005) impose conditions about finite sets of ancestors being true in any given model. BLOG (Brian Milch et al., 2005) allows inference over unknown objects but does not permit cyclic logical constraints and to our knowledge there are no MAP algorithms that have been proposed for it. GenProb has some similarities to CP-logic (CITE), but that latter does not deal with infinity or unknown objects. To our knowledge, GenProb is the only approach with an inference

algorithm for correct MAP inference for theories whose models have infinite domains and incorporates potentially cyclic logical constraints.

6. Conclusions

Probabilistic and logical reasoning over (a potentially infinite number of) unknown objects is an important aspect of general intelligence. Relational probabilistic theories are an effective means of compactly representing and efficiently making inferences in many situations. However, unknown objects and infinite domains can lead to several formal and computational complications using an approach that propositionalizes a relational theory before inference. By expressing theories when using the GenSAT language and translating them into graphical models, a probability distribution over models of these theories can be well-defined. Translations of GenSAT theories to GenSAT models enable correct inference, even in many cases that involve models of infinite size. In other work (Murugesan & Cassimatis, 2009), we have shown that GenProb can be used to enable for the first time model-based probabilistic reasoning that combines language grammars with general knowledge about the world in order to find correct interpretations. Such work and the formal results proposed in this paper demonstrate that probabilistic reasoning in a weighted generative satisfiability framework can be an effective means of addressing some difficult problems in achieving artificial general intelligence.

7. References

- Cassimatis, N. L., Murugesan, A., & Bignoli, P. (2009). *Inference with Relational Theories over Infinite Domains*. Paper presented at the FLAIRS-22.
- Domingos, P., Kok, S., Poon, H., Richardson, M., & Singla, P. (2006). *Unifying Logical and Statistical AI*. Paper presented at the AAI-06.
- Domingos, P., & Richardson, M. (2006). Markov Logic Networks. *Machine Learning*, 62, 107-136.
- Kersting, K., & De Raedt, L. (2001). *Towards combining inductive logic programming with Bayesian networks*. Paper presented at the ILP-01.
- Milch, B., Marthi, B., Sontag, D., Russell, S., & Ong, D. L. (2005). *Approximate inference for infinite contingent Bayesian networks*. Paper presented at the AISTATS-05.
- Milch, B., Marthi, B., Sontag, D., Russell, S., Ong, D. L., & Kolobov, A. (2005). *BLOG: Probabilistic Models with Unknown Objects*. Paper presented at the IJCAI-05, Edinburgh, Scotland.
- Muggleton, S. (1996). Stochastic logic programs. In L. D. Raedt (Ed.), *Advances in inductive logic programming* (pp. 254–264): IOS Press.
- Murugesan, A., & Cassimatis, N. L. (2009). *Parsing PCFG within a General Probabilistic Inference Framework*. Paper presented at the AGI-09.
- Richardson, M., & Domingos, P. (2007). *Markov Logic in Infinite Domains*. Paper presented at the UAI-07.