Identity Uncertainty

by

Hanna Maria Pasula

B.A. (Harvard University) 1996

A dissertation submitted in partial satisfaction of the
requirements for the degree of

Doctor of Philosophy
in
Computer Science

in the
GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA, BERKELEY

Committee in charge:

Professor Stuart J. Russell, Chair
Professor Peter Bickel
Professor Alistair Sinclair

Fall 2003

The dissertation of Hanna Maria Pasula is approved.

Professor Peter Bickel

Professor Alistair Sinclair

Professor Stuart J. Russell, Committee Chair

August Fall

Identity Uncertainty

Abstract

Identity Uncertainty

by

Hanna Maria Pasula

Doctor of Philosophy in Computer Science

University of California at Berkeley

Professor Stuart J. Russell, Chair

Recent work in AI has made clear the advantages to be derived from combining probability theory with the expressive power of first-order logic. One promising approach is based on the concept of possible worlds, where a probability measure is defined over the interpretations defined by a logical knowledge base. This approach has been successfully used to add probabilistic elements to representations based on semantic networks and logic programming. However, all of the representations developed to date have made the unique names assumption; they have assumed that the constants of a language uniquely identify each such object. This is not always reasonable, since objects in the real world are not usually labeled with easily observable unique identifiers. Often, there exists a great deal of uncertainty over the identity mappings of observed objects. This is what we term identity uncertainty, and it is a pervasive problem of real-world data analysis, occurring in numerous settings such as database merging, feature correspondence, and object tracking. We propose an extension to the possible world approaches, one where the uncertainty over the mapping from terms to objects is represented explicitly, by extending the language used to define the probability distribution over possible worlds. We show that this extended language does define a unique and consistent distribution. We also suggest an approximate inference method for use in this scenario. This method is based on Markov chain Monte

Carlo, and we have applied it to several domains, including vehicle matching and citation clustering, with promising results.

Professor Stuart J. Russell
Dissertation Committee Chair

Contents

List of Figures

List of Tables

Chapter 1

Introduction

First-order representations have long been used to represent structured domains, because their ability to handle objects, relations, and quantification lets them represent structure efficiently. However, the traditional deterministic first-order representations, such as first-order logic, do have one major drawback: the constraint that statements must be either true of false makes them ill-suited to representing many real-world domains, because the real world is rife with uncertainty. On the other hand, probability theory provides us with a sound framework for reasoning about uncertainty, but well-known probabilistic representations, such as Bayesian networks, tend to be propositional in nature, and therefore unable to capture much structure and inefficient. Thus, there are clear advantages to be derived from combining probability theory with (at least some of) the expressive power of first-order logic (Wellman, Breese, & Goldman, 1992). We call languages that exhibit such a combination *first-order probabilistic languages* (FOPLs). In the recent past, first-order probabilistic languages have been applied to diverse domains, such as battle-space management (Pfeffer, 2000), hypertext classification (Getoor, Segal, Taskar, & Koller, 2001b), and movies and people's rankings of movies (Getoor, Friedman, Koller, & Taskar, 2001a). The semantics used in all these applications is based on the idea that each model of a FOPL knowledge base should be viewed as a probability measure over the possible worlds (logical interpretations) defined by the constant, function, and predicate symbols of the knowledge base (Gaifman, 1964). Although "wildly undecidable" in full generality, highly restricted FOPLs appear to be practical, especially with finite models. Two threads have arisen, based on semantic networks (e.g., (Koller & Pfeffer, 1998)) and logic programming (e.g., (Sato & Kameya, 1997)).

One thing that most of the current FOPL approaches have in common is that they all make the unique names assumption: they all assume that the constants we use to refer to the objects in a knowledge base uniquely identify each such object. This is not always reasonable, since objects in the real world are not usually labeled with unique identifiers, and even when they are, those identifiers may not be perceived perfectly. In general, two observed objects may or may not correspond to the same actual object. This uncertainty over what the observed objects correspond to — over their actual identities — is what we term *identity uncertainty*, and it is a pervasive problem in real-world data analysis domains. Two types of applications that have to handle identity uncertainty as a matter of course are data association, where the aim is to track individual objects from sensor to sensor, and database merging, where the aim is to identify and combine duplicate records. Several approaches have been proposed for handling both types of problems: however, typically they are application-specific, or even domain-specific, and the representations used tend to have no first-order elements. Since FOPLs already provide a general representation for probabilistic relational applications, we hope that extending them to handle identity uncertainty will yield a general representation for identity-uncertain applications. The extension we propose involves removing the unique names assumption, and designing a language capable of describing a consistent and unique probability distribution over the newly uncertain aspects of possible worlds: the term-to-object mappings, and the number of objects in the world. However, by increasing the number of uncertain aspects of possible worlds, we increase the number of the worlds themselves. We thus make inference in FOPLs with identity uncertainty even harder than inference in ordinary FOPLs, which are already attempting to combine logical and probabilistic inference, two generally intractable problems. Because the problem is so difficult, we have decided to turn to approximate inference methods. One particularly appropriate technique is Markov Chain Monte Carlo (MCMC), commonly used to approximate sums of probabilities by sampling a small number of high-probability terms. We have developed a family of MCMC-based inference algorithms for use in identity-uncertain settings.

1.1 Dissertation Contributions

The thesis behind this dissertation is that MCMC methods can provide a flexible and practical approximate approach to reasoning about identity uncertainty in first-order probabilistic representations. This statement is interesting because the problem of identity uncertainty is, in itself, an interesting, pervasive problem; because FOPLs are a very promising representation; and because it mentions a practical approach for an extremely hard inference problem. It is also undeniably novel, because the problem of generalized identity uncertainty has not been studied in any depth, so that there is no established representation for identity-uncertain domains, let alone an inference algorithm. We hope that the dissertation will demonstrate that it is also well-supported.

The contributions of this dissertation include:

- A method for augmenting first-order probabilistic languages so that they define consistent and unique distributions over identity-uncertain possible worlds, by defining distributions over term-to-object mappings. We have investigated several possible ways of defining such distributions, corresponding to several possible types of identity-uncertain domains.

- An extension of relational probabilistic models which enables them to handle generalized identity uncertainty.
- A family of approximate, MCMC-based algorithms for inference in identity-uncertain settings. For each algorithm, we provide a set of conditions which, when met, guarantee that the algorithm approximates the appropriate quantity.
- Bounds on the convergence time of two Markov chains, a Gibbs sampler for Bayesian networks, and a Metropolis-Hastings sampler for weighted bipartite matchings. We use a flow network approach to prove that, under some conditions, both chains have a polynomial convergence time. These bounds are of interest to us because chains like these are used by some versions of our algorithms.
- The application of our approach to two interesting domains, traffic surveillance, which is an example of data association, and citation matching, which resembles general database merging. In each case, we define an identity-uncertain model for the domain, and then construct and test an inference algorithm.

1.2 Dissertation Overview

In the next chapter, Chapter 2, we outline some of the background concepts and notations used throughout this dissertation. We present first-order languages and typed relational languages, and general probabilistic concepts and Bayesian networks; we then go on to describe the possible world approach in general, discussing several approaches based on it, with particular emphasis on probabilistic relational models. The real contributions of this dissertation begin in Chapter 3, where we describe our approach to identity uncertainty in the context of a simple first-order probabilistic logic, using examples set in the ball and urn domain. We describe a representation for possible worlds, and then discuss several ways of defining consistent and unique distributions over this representation. Chapter 4 deals with Markov chain Monte Carlo, introducing the algorithm and some relevant concepts; it also presents our bounds on the running time of two types of chains. In Chapter 5, we propose a family of approximate, MCMC-based algorithms for inference in identity-uncertain domains. As we introduce each method, we describe the conditions under which it can be used, and prove that it will, when those conditions are met, approximate the right distribution. Next, in Chapter 6, we explain how our approach relates to probabilistic relational models, and apply our algorithms to a simple toy domain. The following two chapters concern our main application domains. Chapter 7 describes a freeway traffic surveillance system involving probabilistic inference about the identities and properties of hundreds of vehicles. Chapter 8 describes the *Citeseer* domain, where we attempt to deduce the identity of the papers responsible for various citations by clustering co-referring citations together. Finally, in Chapter 9, we draw come conclusions and suggest directions for future work.

Some of the work presented here has appeared previously in conference papers. The material in Chapter 7 appeared in (Pasula, Russell, Ostland, & Ritov, 1999), while the material in Chapter 8 appeared in (Pasula, Marthi, Milch, Russell, & Shpitser, 2002). Chapter 6 contains a substantial re-working of the ideas first presented in (Pasula & Russell, 2001). Chapter 3 and Chapter 5 draw inspiration from all three papers.

Chapter 2

Background Information

In this chapter, we review the bulk of the background information used throughout this dissertation, and, in the process, define some of the vocabulary and notation used in later chapters. In Section 2.1, we describe basic logical concepts and present two relational logical representations: simple first-order languages and typed relational languages. In Section 2.2, we review some basic probability theory and introduce Bayesian networks. Elements from the two sections are combined in Section 2.3, where we outline the possible worlds approach to combining probabilistic and relational representations, and go on to discuss relevant work.

2.1 Logic

In this section, we briefly describe the syntax and semantics of simple first-order languages; we also touch upon the unique names assumption, a common assumption which makes identity uncertainty impossible. A more detailed discussion of such topics can be found in any standard text on logic, such as that of Enderton (1972). We also introduce typed relational languages, a semantic-network-like representation which groups objects into classes with pre-defined attributes, and is therefore well suited to reasoning about different types of objects, their properties, and their identities. We relate the concept of possible worlds to both types of languages.

2.1.1 First-Order Languages

We begin by describing the **syntax** of first-order logic. A first-order language $\mathcal{L}$ usually consists of the following elements:

- **Logical symbols**: connectives (such as $\vee$, $\wedge$, $\neg$, $\Rightarrow$, and $\Leftrightarrow$), quantifiers, (such as $\forall$ and $\exists$; we also use the numerical quantifiers $\exists_1$, $\exists_2$, ...,[1]) variables (e.g., x_1, y), and the equality symbol $=$. The equality symbol is optional, but in this dissertation we always include it.
- A set **I** of **constants**.
- For each positive integer n, a set $\mathbf{f}_n$ of **functions** of arity n.
- For each positive integer n, a set $\mathbf{R}_n$ of **predicates** of arity n.

Terms, logical expressions representing objects, are formed by combining variables, constant symbols, and function symbols. Predicates are applied to terms to yield **atomic formulas**, and **well-formed formulas** are built up from atomic formulas using the connectives and quantifiers. Variables outside the scope of a quantifier are **free**. A formula is **closed** if it contains no free variables; closed formulas are also called **sentences**. A collection of sentences is called **consistent** if it contains no contradictions; a **knowledge base** is a (consistent) collection of sentences. Logical expressions that contain no variables at all are **ground**. Ground terms refer to specific objects, either explicitly (using constant

[1] Where $\exists_i$ states that there exist exactly i objects.

symbols) or implicitly (by applying function symbols to constant symbols.) In this dissertation, we sometimes concern ourselves with the set of ground terms of a language, $\mathbf{T}$. As long as there is at least one constant in the language, $\mathbf{T}$ is also known as the **Herbrand Universe** of that language.

The **semantics** of a first-order language is defined by specifying an **interpretation** ω for the language. In addition to the meanings of the logical symbols, which we assume to be conventional, such an interpretation consists of the following:

- a set of objects, Δ^ω.
- for each constant $I \in \mathbf{I}$, an object $I^\omega \in \Delta^\omega$.
- for each $f \in \mathbf{f_n}$, a function f^ω mapping from $(\Delta^\omega)^n$ to Δ^ω.
- for each predicate $R \in \mathbf{R_n}$, a relation $R^\omega \subset (\Delta^\omega)^n$, i.e., a set of n-tuples of objects.

Thus, an ω specifies three things: what objects are present in the world, which of the instantiations of the predicates are true, and which object each of the ground terms in $\mathbf{T}$ refers to. Uncertainty over the latter is the focus of Chapter 3 (and, indeed, of much of this dissertation.) This uncertainty can be constrained by making the **unique names assumption** (Reiter, 1980), which states that if terms cannot be proven equal from the knowledge base, they must be unequal. Under this assumption, only the interpretations with the fewest possible **co-referring** terms (that is, terms of the language that refer to the same object of an interpretation) are valid interpretations for the knowledge base. Since the set of objects can be chosen arbitrarily, many of the interpretations will be **isomorphic**—that is, they will differ only in the names of the objects.

Given a particular interpretation ω, a sentence S is either true or false. If it is true, we can write $\models_\omega S$. A sentence S is said to be **entailed** by a knowledge base $\mathbf{K}$ if it is true under every interpretation that makes the sentences in $\mathbf{K}$ true. This is written as $\mathbf{K} \models S$. Inference algorithms attempt to implement the entailment relation between sentences; some algorithms make the **closed world assumption**, which states that if a statement cannot be inferred from the knowledge base, the statement must be false.

Possible worlds can be defined in several ways.[2] In this dissertation, we define them in terms of interpretations: given a knowledge base $\mathbf{K}$, the possible worlds of $\mathbf{K}$ are the interpretations under which all the sentences of $\mathbf{K}$ are true. However, most of the time, we group some isomorphic possible worlds together into sets, and abuse this definition slightly by referring to each such *set of worlds* as a possible world.

2.1.2 Typed Relational Languages

Typed relational languages have been described in depth by Pfeffer (2000) and Getoor (2001), and elaborated upon in later publications such as (Getoor et al., 2001a). Using the terminology presented by Getoor (2001), a **relational language** consists of:

- $\mathbf{C}$, a set of **classes**, and possibly $\sqsubseteq$, a partial ordering over $\mathbf{C}$, which defines the class hierarchy. (Multiple inheritance is not permitted.)

- $\mathbf{A}$, a set of **descriptive attributes**. An $A \in \mathbf{A}$ is a specialized function with a domain type $Dom[A] \in \mathbf{C}$ and a range type $Range[A] = Val[A]$, where $Val[A]$ is a predefined set of values. Thus, descriptive attributes are discrete, with finite ranges. (Pfeffer (2000) refers to descriptive attributes as **simple attributes**.)

- $\mathbf{R}$, a set of **reference slots**. An $R \in \mathbf{R}$ has a domain type $Dom[R] \in \mathbf{C}$ and a range type $Range[R] \in \mathbf{C}$.[3] Reference slots are not necessarily one-on-one: **multivalued** slots can return a *set* of objects of type $Range[R]$. For simplicity, the

[2] One common definition states that they are "maximally consistent sets of propositions".
[3] Getoor (2001) augments the language with the inverses of the functions in $\mathbf{R}$.

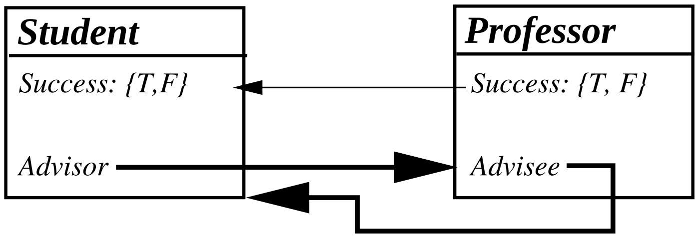


Figure 2.1. A relational schema for our university domain example. The large rectangles represent classes, and the dark arrows indicate the ranges of their reference slots.

individual members of this set cannot be accessed directly, but only through quantifiers, which define quantities on the members' attributes. We define **slot chains** as sequences (possibly empty) of reference slots $B_1.B_2.\ \ldots\ .B_n$, where $Dom[B_i]$ always matches $Range[B_{i-1}]$: slot chains can be applied to the objects of any class matching $Dom[B_1]$. (Pfeffer (2000) refers to reference slots as **complex attributes**.)

- **I**, a set of constants, where each constant $I \in \mathbf{I}$ has an associated class $C[I]$.

The set of language elements is assumed to be finite. The classes, attributes, and reference slots together form a **relational schema**, which defines each class $C \in \mathbf{C}$ in terms of the attributes and reference slots that take that class as their domain type. The reference slots describe the ways in which objects may relate to each other, while the descriptive attributes describe the non-relational properties of the objects.

Example 2.1: *The university domain.* Let us define a very simple language for representing the world of universities. This language contains only the classes $\{$*Student*, *Professor*$\}$, both of which are subclasses only of themselves. The *Student* class has the reference slot *Advisor*, ranging over *Professor*s, and the *Professor* class has the corresponding (multivalued) reference slot *Advisee*, ranging over *Student*s. Since *Advisee* is multivalued, we cannot access its values directly: to access it at all, we will have to create quan-

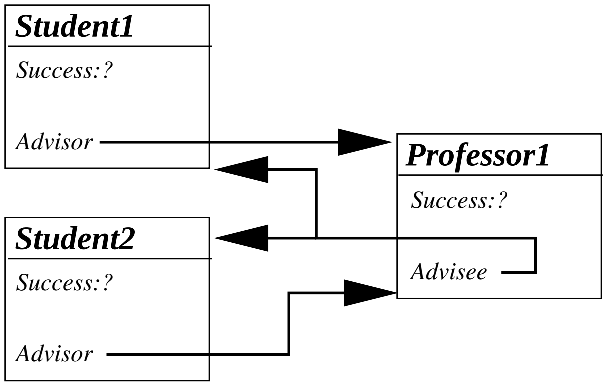


Figure 2.2. A possible relational skeleton for our university domain example, where *Prof*1 is the advisor of both the *Student*s. The large rectangles now represent objects, and the dark arrows indicate the relations between them.

tifiers, such as $\#[\mathit{Advisee.Success} = T]$, which counts up a *Professor*'s successful advisees. As for descriptive attributes, the two classes share a single one: *Success*, where $Val[\mathit{Success}] = \{T, F\}$. This simple relational schema is illustrated in Figure 2.1. There language also contains three typed constants, *Student*1, *Student*2, and *Prof*1, not shown in the schema.

A **relational knowledge base** $\mathcal{K}$ (Pfeffer, 2000) is a relational language augmented with additional constraints: statements setting the values of some of the attributes and reference slots. A **relational skeleton** (Getoor, 2001) combines the set of constants with a full specification of the values of all the reference slots, so that the only remaining unknowns are the values of the descriptive attributes. One possible relational skeleton for our example language is presented in Figure 2.2

Just as with first-order logic, the semantics of a relational language is defined in terms of interpretations. An interpretation ω (called an **instance** $\mathcal{I}$ by Getoor (2001)) specifies the following:

- A set of objects, Δ.
- For each class $C \in \mathbf{C}$, a subset $\Delta_C \subseteq \Delta$. (Subsets are constrained according to the class hierarchy $\sqsubseteq$.)
- For each descriptive attribute $\delta.A$ (where $\delta \in \Delta$), a value in $Val[A]$.
- For each reference slot $\delta.R$, an element of $\Delta_{Range[R]}$
- Similarly, for each constant I, an element of $\Delta_{C[I]}$.

As before, those interpretations that ensure the truth of all known constraints (such as those expressed in a relational skeleton or knowledge base) are known as **possible worlds**. To avoid isomorphism, Pfeffer (2000) constrains the set of possible worlds further by restricting Δ to a subset of the Herbrand universe $\mathbf{T}$, one where all the terms are known to be distinct. In this restricted Δ, every subset $\mathbf{S} \subseteq \mathbf{T}$ such that all the terms in $\mathbf{S}$ are known to be equal is represented by a single term. The unique names assumption guarantees that there is a unique restricted Δ, since it ensures that the terms that are not forced to be equal by some constraint cannot be equal in an interpretation, and therefore that inter-term equalities are fully specified by the constraints of the skeleton. In our example domain, as presented in Figure 2.2, *Student*1 can never be equal to *Student*2, and

$$\Delta = \{\mathit{Prof}1, \mathit{Student}1, \mathit{Student}2\} .$$

Note that the object *Prof*1 here represents not just the term *Prof*1, but also the term *Student*1.*Advisor* and so on. The situation is a little more complicated if there is no relational skeleton, but there are a few other constraints. Consider the constraints $\mathit{Student}1.\mathit{Advisor} = \mathit{Prof}1$ and $\mathit{Prof}1.\mathit{Advisee} = \{\mathit{Student}1.\mathit{Advisor}\}$. In that case, the unfilled slots (here, *Student*2.*Advisor*) are not equal to any constant, and are therefore represented by functional terms, so that

$$\Delta = \{\mathit{Prof}1, \mathit{Student}1, \mathit{Student}2, \mathit{Student}2.\mathit{Advisor}\}\ .$$

Either way, however, the set of objects and the mappings of both the constants and the reference slots are fully specified by the language and the constraints; the only aspects of ω that remain unconstrained are the mappings the descriptive attributes, such as

$$\{\mathit{Prof}1.\mathit{Success}, \mathit{Student}1.\mathit{Success}, \mathit{Student}2.\mathit{Success}, \mathit{Student}2.\mathit{Advisor}.\mathit{Success}\}\ .$$

Each possible world assigns values to these four attributes. When we introduce probability, we will view such attributes as random variables.[4]

It is worth noting that this approach makes an additional implicit assumption that the world contains no extraneous objects unreferenced by constants or functional terms.

PRM extensions

The semantics given in (Pfeffer, 2000; Getoor, 2001), and outlined above, assumes that the set of objects is known, as are the fillers of all the reference slots, either because all the slot fillers are pre-set by constraints, or because they are filled with unnamed objects representing the functional terms. Several extensions have been proposed. These include:

- **Reference uncertainty**, which occurs when the values of the reference slots are not known. In the simplest case, we assume that we know the number, if not the actual values, of the objects referred to. Then, a slot of arity n ranges over all n-subsets of the class $Range[B]$. For example, *Professor.Advisee*, if known to refer to just one student, might range over $\{\{\mathit{Student}1\}, \{\mathit{Student}2\}\}$.

[4] It is worth emphasizing that *random* variables are quite different from the *logical* variables that are an element of a first-order language.

- **Number Uncertainty**, which occurs when the number of values an attribute B can take can varies from possible world to possible world. Here, the class containing B gets an additional attribute $\#[B]$, signifying the number of values of B. $\#[B]$ ranges over some interval $\{1, \ldots, \max(\#[B])\}$. We might thus introduce the attribute $\#[$*Advisee*$]$, which ranges over $\{0, 1, 2\}$.

  In this situation, possible worlds are defined slightly differently. First of all, they must meet the additional constraint that the number of fillers of a number-uncertain attribute B should be equal to the value of $\#[B]$. Secondly, they are expanded so that they now contain all the attributes of all the **possible entities**, that is, all the objects that might exist when the number-uncertain reference slots are at their maximum capacity.

- **Existence Uncertainty**, which, as presented by Getoor et al. (2001a), occurs when the set of objects is not fully known: instead, we are given a set of potential objects, but we are not certain as to which of them actually exist. A class exhibiting existence uncertainty is called **undetermined**, and each instance I of the class gets an additional **existence attribute** E, where $Val[E] = \{T, F\}$. As before, possible worlds contain all the attributes of all the possible entities.

We will say more about our interpretations of these extensions in later chapters, especially in Chapter 6.

2.2 Probability

In Section 2.1.2, we mentioned that possible worlds can be described as value assignments to a set of random variables. In this section, we formally explain the concept of random variables, outline some basic probability notation, and go on to introduce Bayesian networks (Pearl, 1988), a compact and popular probabilistic representation.

In general, a **probability space** consists of a set of values $\mathbf{S}$ (called a **state space** or **sample space**), a σ-algebra $\mathbf{X}$ on $\mathbf{S}$, and a measure P on $\mathbf{X}$ such that $\sum_{S \in S} P(S) = 1.0$. P is known as a **probability distribution**. For each element X of $\mathbf{X}$, also known as an **event**, $P(X)$ represents its probability. A **random variable** can be defined as a measurable function from a **probability space** to an $\langle \mathbf{S}, \mathbf{X} \rangle$ pair. In this dissertation, we focus mostly on discrete random variables. In a discrete setting, the state space $\mathbf{S}$ is finite or countable, and P is simply a function on $\mathbf{S}$. Discrete random variables range over the elements of a sample space, sometimes known as **outcomes**.

A probability distribution is **consistent** with the above definitions if it obeys the **axioms of probability**: if it is non-negative, if it is additive over mutually exclusive events, and if it sums to one.

Let $\mathbf{V} = \{V_1 \dots V_n\}$ be a set of discrete random variables, where each variable V_i ranges over a set $Val[V_i]$. The outcomes of the variables in $\mathbf{V}$ are represented using lowercase letters, such as $\mathbf{v}$ or $\mathbf{x}$, and can be indexed using either indices (so that $\mathbf{v}_i$ denotes the value of V_i in $\mathbf{v}$), or sets (so that $\mathbf{v_X}$ denotes the values of the elements of set $\mathbf{X} \subset \mathbf{V}$ in $\mathbf{v}$.) The function $P(\mathbf{v})$ is shorthand for $P(v_1 \dots v_n)$ or $P(\mathbf{v_V})$, and denotes the probability that $\mathbf{V} = \mathbf{v}$.

The expression $\mathbf{P}(\mathbf{V}) = \mathbf{P}(V_1, V_2, \dots, V_n)$ denotes a distribution over all the possible values, known as the joint probability distribution over all the variables, or simply the **joint**. The probability distribution over a single variable is known as a **marginal**, and can be obtained from the joint by summing out all other variables. E.g., for $n = 2$,

$$P(v_1) = \sum_{v_2 \in Val[V_2]} P(v_1, v_2) \; .$$

Probabilities can also be expressed using **conditional** distributions, where the distribution over some set of variables, $\mathbf{X} \in \mathbf{V}$, is dependent on the values of some other set of variables, $\mathbf{Y} \in \mathbf{V}$. Such a distribution is written as $P(\mathbf{X}|\mathbf{Y})$ and defined as

$$P(\mathbf{X}|\mathbf{Y}) = \frac{P(\mathbf{X},\mathbf{Y})}{P(\mathbf{Y})} \tag{2.1}$$

if $P(\mathbf{Y}) \neq 0$; its undefined otherwise. Conditional probabilities are also referred to as **posterior** probabilities or **posteriors**, in contrast to unconditional probabilities which are known as **prior** probabilities or **priors**.

We can use Equation 2.1 to derive other interesting rules, such as the **product rule**,

$$P(\mathbf{X},\mathbf{Y}) = P(\mathbf{X}|\mathbf{Y})P(\mathbf{Y})\,,$$

and **Bayes' rule**,

$$P(\mathbf{X}|\mathbf{Y}) = \frac{P(\mathbf{Y}|\mathbf{X})P(\mathbf{X})}{P(\mathbf{Y})}\,.$$

As we will see, probabilistic inference often makes use of Bayes' rule. The product rule, meanwhile, permits us to express joint probabilities by cascading conditionals as in

$$P(V_1 \ldots V_n) = P(V_1|V_2 \ldots V_n)P(V_2|V_3 \ldots V_n) \ldots P(V_n)\,.$$

This may not look like much of an improvement, but conditional independence can help us simplify the expression further. In general, a set of variables $\mathbf{X}$ is **conditionally independent** of another set of variables $\mathbf{Y}$ if the distribution over $\mathbf{X}$ is the same no matter what value $\mathbf{Y}$ takes. When

$$P(\mathbf{X}|\mathbf{Y},\mathbf{Z}) = P(\mathbf{X}|\mathbf{Z})$$

we say that $\mathbf{X}$ is conditionally independent of $\mathbf{Y}$, given $\mathbf{Z}$. When conditional independence is present, the product of the conditionals can be much more compact than the full joint. For example, let us assume that every variable V_i is conditionally independent of $V_{i+2} \ldots V_n$, given V_{i+1}. We can then rewrite the joint as

$$\mathbf{P}(V_1 \ldots V_n) = \mathbf{P}(V_1|V_2)\mathbf{P}(V_2|V_3) \ldots \mathbf{P}(V_{n-1}|V_n)\mathbf{P}(V_n)\,.$$

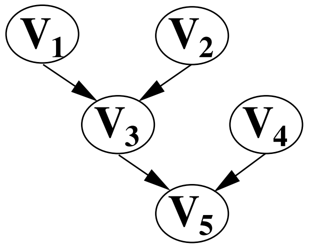
The new version tends to be more space-efficient. For example, if we represent all of our distributions using tables, and if If $|Val[V_i]| = d$ for all the V_is, the tables called for on the right-hand side will have a total of $(n-1)d(d-1)+d-1$ independent entries, while the table for the full joint would require $d^n - 1$ entries.

It may sometimes be the case that a variable V_1 is independent of V_2 only when some other variable V_3 has a certain value. This is known as **context-specific independence**.

Probabilistic inference usually involves calculating the posterior distribution over some subset of $\mathbf{V}$ given some known assignment of values to another subset of $\mathbf{V}$. The known variables are called **evidence** or **observed** variables. We will refer to the set of observed variables as $\mathbf{O} \subseteq \mathbf{V}$. The variables we want to reason about are called **hidden** variables, and we will refer to them as $\mathbf{H} \subseteq \mathbf{V} - \mathbf{O}$. When $P(\mathbf{H}|\mathbf{O})$ is not a readily available conditional distribution, either Equation 2.1 or Bayes' rule can be used to rewrite it. However, since $P(\mathbf{O})$ will be constant for any particular set of observations, we often replace it with a **normalization constant** α, as in $P(\mathbf{H}|\mathbf{O}) = \alpha P(\mathbf{H}, \mathbf{O})$. Figuring out the value of α (by summing over all the values of $\mathbf{H}$) is called **normalization**.

2.2.1 Bayesian Networks

A Bayesian network (Pearl, 1988) is a directed acyclic graph, $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where the vertices (nodes) correspond to probabilistic variables, and the edges correspond to direct conditional dependencies. Let us define Pa_i as the set of nodes pointing to V_i (i.e., its parents), and Ch_i as the set of nodes it points to (i.e., its children). Associated with each node V_i is a **conditional probability function**(CPF), $\mathbf{P}(V_i|Pa_i)$. The distribution over all the variables is then the product of all these conditional probabilities; Figure 2.3 shows what this product would look like for the simple Bayesian network displayed. It is worth emphasizing that the underlying graph of a Bayesian network must be acyclic, as otherwise the probability model of the network cannot be seen as a cascade of conditionals, and is ill-defined.



$$P(V_1,V_2,V_3,V_4,V_5) = P(V_5|V_3,V_4)\,P(V_3|V_1,V_2)\,P(V_4)\,P(V_2)\,P(V_1)$$

Figure 2.3. An example Bayesian network, together with the definition of its joint.

The **Markov blanket** of a node in a Bayesian network consists of the node's parents, children, and its children's other parents. Thus, the Markov blanket of V_3 in Figure 2.3 contains all the other nodes, while that of V_4 contains only V_3 and V_5. We will sometimes refer to the Markov blanket of node V_i as MB_i. It can be shown that, given its Markov blanket, a node is conditionally independent of the rest of the network. Exact inference in Bayesian Networks is commonly done using algorithms such as **variable elimination**, where the variables are summed out one by one. The time and space cost of these algorithms is $O(nd^{(m+1)})$, where m is the **tree width** (see Appendix A) of the **moral graph** of the network. (The moral graph of a Bayesian network is the undirected version of the network, where additional undirected links are introduced to connect the parents of every node together.)

Ordinary Bayesian networks are limited in how they can represent, or make use of, context-specific independence. However, they can be generalized to yield **Bayesian Multinets** (Geiger & Heckerman, 1996), where the structure of the network is dependent on the values of some subset of the networks nodes: essentially, every set of such values is associated with a different network. Multinet inference is carried out by summing over these networks.

2.3 Possible Worlds

As we have explained in Chapter 1, recent work in AI has made clear the advantages to be derived from combining probability theory with (at least some of) the expressive power of first-order logic (Wellman et al., 1992). One approach to annotating first-order representations with probability (or, alternatively, introducing relational concepts to probabilistic representations) involves defining a distribution $P(\omega)$ over the set of possible worlds. Given a $P(\omega)$, the probability of any given sentence S is then defined to be the sum of the probabilities of all the worlds in which the sentence is true, as in

$$P(S) = \sum_{\omega : \models_\omega S} P(\omega) \, . \tag{2.2}$$

However, before we can make use of Equation 2.2, we need to make sure that $P(\omega)$ is unique and consistent. Because of the size of the space, it is not usually practical to define $P(\omega)$ explicitly, by enumerating all the possible worlds. Instead, we must design a representation that combines logic and probability in a factored way, by defining a set of smaller distributions over different aspects of the possible worlds which can then be combined to yield $P(\omega)$.

One rather obvious approach is to simply annotate each of the sentences of a logical knowledge base with a number in the interval $[0, 1]$, indicating the probability that the sentence is true. However, there is a complication: the truth values of sentences are not always independent of each other, and so their probabilities may constrain each other. Nilsson (1986) discusses such constraints, and gives examples such as

$$P(\exists\, y p(y)) + P(\forall\, x p(x) \rightarrow p(x)) - 1.0 \leq P(\exists\, z p(z)) \leq 1.0 \, .$$

Obviously, there is no guarantee that an arbitrary assignment of probabilities will be consistent. Nilsson (1986) also discusses the problem of deriving $P(\omega)$ from the probabilities of the sentences, and describes a general method. This method involves encoding the truth of each sentence in each possible world in a matrix $\mathbf{M}$ where each column represents a world, each row represents a sentence, and the truth or falsehood of a sentence in a world is represented using a 1 or a 0. Then, if π is the vector of sentence truth values, and $\mathbf{P}$ is $P(\omega)$ in vector form, Equation 2.2 permits lets us state that

$$\pi = \mathbf{MP}$$

and the problem can be solved by solving for $\mathbf{P}$ (when possible). However, this approach is clearly feasible only for very small problems.

More practical approaches tend to restrict the language, disallowing arbitrarily complex sentences, and sometimes constraining the set of sentences that may be annotated with probabilities. As a result, they can guarantee that the probability distribution over possible worlds will be consistent as well as possible to calculate. We will now go through several such approaches. Most of them base their restricted language around Prolog-like **definite clauses**: sentences of the form $a_0 \leftarrow a_1 \wedge \ldots a_n$ where the a_is are atomic formulas, and all free variables are assumed to be universally quantified. In a definite clause, a_0 is called the head; the remaining atomic formulas form the body. Definite clauses without a body are called unit clauses.

Poole (1992) and Sato and Kameya (1997) both use knowledge bases consisting of a set of deterministic definite clauses, and an additional element defining a joint distribution over a set of unit clauses. In each case, the deterministic clauses are restricted in that no predicate appearing in the unit clauses can appear in the head of a deterministic clause. The distributions are defined slightly differently. Poole (1992) uses probabilistic sentences of the form *disjoint*($[a_1 : p_1, \ldots, a_n : p_n]$) where the a_is are atoms sharing the same variables, and the p_is are positive reals which sum to 1.0. The atoms represent mutually exclusive possibilities, so that each *disjoint* sentence is equivalent to a set of $P(a_i) = p_i$ statements together with the deterministic clause $a_i \wedge \ldots a_n \rightarrow F$. An example disjoint

Success(x, T)	0.6
Success(x, F)	0.4

Table 2.1. A probability distribution over possible assignments to the final variable of the *Success* predicate.

sentence might be

$$disjoint(successful(x) : 0.6, unsuccessful(x) : 0.4) .$$

Sato and Kameya (1997) define distributions over all the possible assignments to the final variable v_n of an atomic formula $q(v_1 \dots v_n)$; this is rather like making a *disjoint* statement where the atoms differ only in the assignment to v_n. Such distributions may be expressed in the form of a lookup table, as in Table 2.3. They go on to prove that the deterministic sentences extend the distribution over all the other predicates of the language.

Probabilistic Logic Programs (Ngo & Haddawy, 1995) also define disjoint distributions over the final variables of atomic formulas: however, they do not restrict this only to unit clauses. In their language, any definite clause may be annotated with a distribution. If the clause has a body, this is a conditional distribution over the final variable of the head given that the body is true. We might thus use Table 2.3 to define a distribution over y in

$$\mathit{Success}(x, y) \leftarrow \mathit{Advisor}(x, \mathit{Prof}1) .$$

When any predicate that heads more than one clause, Ngo and Haddawy (1995) provide a **combining rule**, a method for merging the defined distributions.

Bayesian Logic Programs (Kersting & De Raedt, 2000) enhance this type of representation by replacing the predicates with range-restricted functions, so that instead of defining a distribution over the final variable v_n, they define a distribution over some set of possible values $Val[q]$ of a function $q(v_1 \dots v_{n-1})$. (Thus, *Success* as used in Table 2.3 would become a regular predicate, since $Val[\mathit{Success}] = \{T, F\}$.) This removes the need for exclusivity constraints, and also simplifies inference. The definite rules are constructed out of such range-restricted functions. In addition, Bayesian Logic Programs impose the constraint that, for any definite clause in the knowledge base, a distribution must be defined for all the possible assignments of values to the functions in the body. Thus, if we let $Val[advisor] = \{\mathit{Prof}1, \mathit{Prof}2\}$, the statement

$$\mathit{Success}(x) \leftarrow \mathit{Advisor}(x)$$

would have to be accompanied with distributions over $\{T, F\}$ for both values of *Advisor*. Kersting and De Raedt (2000) also emphasize the connection between this representation and Bayesian networks: when ground, the range-restricted functions are equivalent to random variables with range $Val[q]$, the definite clauses provide the directed links, and the CPFs are defined by the annotating distributions. Thus, the clauses of a Bayesian Logic Program can be viewed as quantified descriptions of Bayesian network fragments.

Finally, Probabilistic Relational Models (Pfeffer, 2000; Getoor, 2001) restrict the logical syntax of the knowledge base quite severely – as we have shown in Section 2.1.2, where we presented the Typed Relational Languages on which they are based. Probabilistic annotations are used to describe the distributions over the attributes and reference slots. Formally, each uncertain attribute B of a class C gets a **local probability model**, which consists of:

- A set of parents $\mathbf{A} = A_1, \ldots, A_n$, where each S_i is a (possibly empty) slot chain on C and each A_i is an attribute of the appropriate class (that is, the range of the final slot in the chain.)

- A CPF $\mathbf{P}_B(B|\mathbf{A})$, from $Val[\mathbf{A}]$ to $Val[B]$

As with Bayesian Logic Programs, each of these local probability models can be said to define a Bayesian network fragment, quantified over all the instances of the class C.

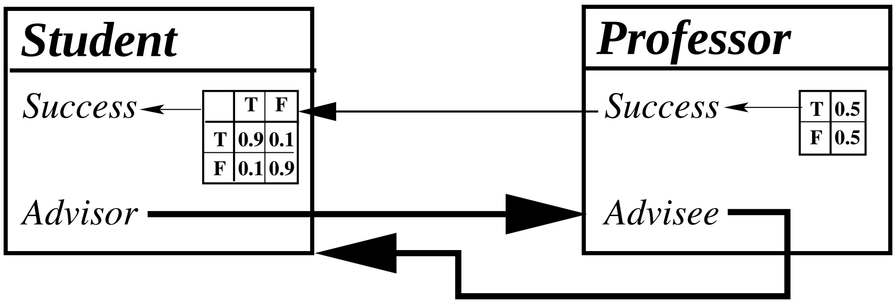


Figure 2.4. A PRM corresponding to the our university domain schema of Figure 2.1, with probability models in the form of conditional probability tables.

Since we will be using PRMs extensively in later chapters, let us consider an example. The attribute *Student.Success* might have *Student.Advisor.Success* as its only parent, and a CPF where the advisor's success, or lack thereof, is transferred to the student with probability 0.9. Meanwhile, *Professor.Success* might have no parents, and a uniform distribution. These probability models can be added to the schema of Figure 2.1 in the form of tables, as in Figure 2.4.

Given a set of instances to ground all the variables, the local probability models can be combined into a network in the obvious way, by taking the uncertain attributes as the vertices and introducing directed edges from parent to child. (For our example, this results in the network of Figure 2.5.) Whenever the constructed network is acyclic, it is obviously a Bayesian network: in fact, it is the **equivalent Bayesian network** of the relational knowledge base. This network defines a joint distribution over all the uncertain attributes; since possible worlds are defined in terms of attributes, it will also define a distribution over the possible worlds.

When the network contains cycles, it is an ill-defined probabilistic model. Given an underlying PRM, a set of instances, and a set of inter-instance relations, we can make sure that our model is well-defined by constructing the equivalent BN and checking it for cycles. We can also make more general guarantees. Getoor (2001) does so by constructing a **class dependency graph**, where the nodes are all the attributes of the PRM and the directed links represent all the possible parent-child dependencies among those attributes, and proving that as long as this graph is acyclic, so is the equivalent BN for every possible set of instances and inter-instance relations.

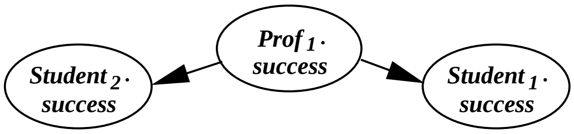


Figure 2.5. The Bayesian network equivalent to the PRM of Figure 2.4.

2.3.1 Possible Worlds and Inference

Given that probabilities of sentences are defined as in Equation 2.2, as summations over the probabilities of the possible worlds, conditional probabilistic inference works in the conventional way: For a set of evidence sentences $\mathbf{O}$, the probability of a set of query sentences $\mathbf{H}$ is defined as

$$P(\mathbf{H}|\mathbf{O}) = \frac{P(\mathbf{H},\mathbf{O})}{P(\mathbf{O})} = \frac{\sum_{\omega:\models_\omega(\mathbf{H}\cap\mathbf{O})} P(\omega)}{\sum_{\omega:\models_\omega\mathbf{O}} P(\omega)} \tag{2.3}$$

In general, $\mathbf{O}$ and $\mathbf{H}$ could conceivably include any sentence of the relevant language. However, to make inference easier, restrictions are usually placed on both. Where the representations reviewed in this section allow for conditional inference, they generally define both the evidence and the query as conjuncts of atomic formulas.

Still, it is not always obvious, when setting up a probabilistic inference problem of this type, just which sentences should be part of the knowledge base as deterministic constraints (so that they affect the distribution by constraining the state space), and which sentences should be expressed as observations to be conditioned on. In this dissertation, we have generally tried to pick the approach that makes defining a general probabilistic model easiest.

Chapter 3

Identity Uncertainty

We define **identity uncertainty** as uncertainty about the identity of the objects referred to by terms. We represent this form of uncertainty using the concept of possible worlds, which was discussed in Section 2.3. As in other work based on this concept, we present a restricted relational language augmented with some probabilistic distributions, and explain how this language defines a consistent and unique distribution over possible worlds. However, while other approaches invoke the unique names assumption to remove all uncertainty over the term-to-object mapping, we treat this mapping as one of the uncertain aspects of our possible worlds. We have tried to ensure that our approach is of practical use by considering different ways in which this form of uncertainty can occur in real-world settings, such as record merging, feature correspondence, and tracking.

The general ideas behind our approach are discussed in Section 3.1. In Section 3.2, we describe a family of simple first-order probabilistic languages which allow for identity uncertainty, but which define a probability distribution over term-to-object mappings by making certain assumptions, rather than by allowing us to express complex distributions within the language. Then, in Section 3.3, we present a representation for identity-uncertain possible worlds: we make use of this representation in Section 3.4, where we explain how

Figure 3.1. The ball and urn domain, where coloured balls are randomly drawn from inside a covered urn, has long been used to illustrate various aspects of probability theory. We will use several adaptations of it throughout the chapter. In general, we will be given some imperfect method of observing the balls, and will then attempt to determine the number and characteristics of the balls within the urn.

our languages define a probability distribution over possible worlds. In Section 3.5, we describe how our simple languages may be extended to express more complicated distributions, where the term-to-object mappings may depend on the attributes of the different objects. Finally, in Section 3.6, we apply the ideas in this chapter to dynamic domains, where the contents of the knowledge base are indexed by time, and a temporal process controls which of the objects may be referenced at any moment.

3.1 The General Approach

The general assumption behind our approach is that there exists, in the world, some set of objects, and that there also exists some probabilistic process that controls which of these objects come to our attention, and so end up represented within our knowledge base.

In the simplest case, these objects are "observed" one by one, as if they were being sampled using some distribution over the full set of objects. This description clearly applies to balls sampled uniformly at random from an urn, as in Figure 3.2, but we suggest

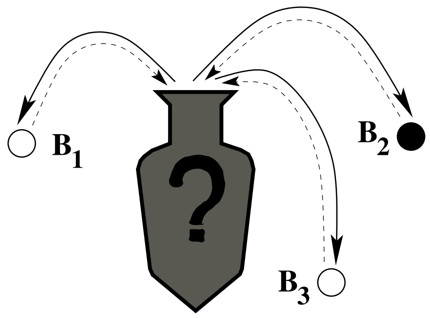


Figure 3.2. The simplest version of the ball and urn domain, where single balls are drawn from the urn with replacement. The three balls represent three different samples, with B_i representing the ith sample.

that there are many other domains where "observations" might be said to be generated with replacement from some set of objects. Paper citations, for example, can be viewed as observations of papers, and credit records can be viewed as observations of credit card customers. In each of these settings, we may be required to reason about the properties of the underlying set of objects (the balls in the urn, the papers, or the customers): this would involve reasoning about what the actual set of underlying objects is. Sometimes, the "observations" may be grouped into sets guaranteed to contain no duplicates (as citations might be grouped together in a bibliography, and credit records in a database), and the objective will be to merge these sets. In these situations, it makes sense to think of a set of objects being sampled all at once, or without replacement: for example, a bibliography containing four citations might correspond to four papers drawn simultaneously from the same large collection of papers.

A rather different type of scenario we might want to handle is one where all the objects present in the world are observed simultaneously by a sensor, but this sensor can miss observations, either because it is unreliable, or because some objects are obscured with respect to the sensor. We will call a set of observations made by such a sensor an image. We might collect several such images — either using different sensors, or at different moments in time when either the sensor or the objects are mobile — with the objective of matching up the objects or features observed at each time with each other. This description applies to many vision domains (Dellaert, 2001). In addition, reasoning about these matches might involve reasoning about the mechanism responsible for the differences between the images: about the relative positions of different objects (e.g., when reconstructing the three-dimensional structure of an object from several two-dimensional images displaying the object's various features); the motion of the objects (e.g., when tracking multiple targets); or the motion of the sensor (e.g., when performing motion recovery.) A very simple ball-and-urn version of this scenario is illustrated in Figure 3.3. In such domains, it no longer makes sense to think of the observed objects as being sampled: it is more appropriate to model the probability that each particular object is observed by the sensor, and consider a distribution over all the possible images.

Finally, in addition to generative distributions motivated by the problem domains mentioned above, we also allow for user-defined distributions to handle a restricted subset of the situations where the user has some knowledge as to which of the terms in the knowledge base correspond to the same object. In particular, we permit the user to specify a distribution over how the functional terms of the language might map to its constant terms. This will prove useful when we come to apply our approach to PRMs with reference uncertainty.

3.2 A Simple Language

In general, we hope that our approach can be used in conjunction with many kinds of logical languages, including both conventional first-order languages and typed relational languages, as defined in Chapter 2. In this chapter, we focus on conventional first-order

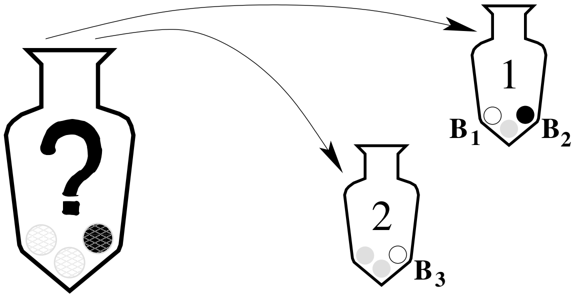


Figure 3.3. The ball and urn version of the incomplete observation domain, where two different images of the inside of the urn (represented by the smaller, numbered urns) contain incomplete observations of the balls. The B_is represent the balls actually observed; the indices i are used to distinguish them. Note that, since B_1 and B_2 occur in the same image, we will have to constrain them to be different.

languages, possibly with the addition of specialized, **range-restricted** functions: that is, functions which take their values not from a set of objects, but from some predefined set of values $Val[]$ which do not map to objects. (Such functions can also be viewed as generalized predicates ranging over more than just $\{T, F\}$.) Range-restricted functions are used in other probabilistic relational representations (Kersting & De Raedt, 2000; Pfeffer, 2000). We will sometimes group the predicates and range-restricted functions of a language together, and refer to them as the **attributes** of that language.

Throughout this section, we will make use of a simple language describing our example domain.

Example 3.1: *The Ball and Urn Domain.* In order to represent this domain, we define a family of simple first-order languages. Each language $\mathcal{L}$ consists of:

- Logical symbols $\vee$, $\wedge$, $\neg$, $\Rightarrow$, and $\Leftrightarrow$, quantifiers (including numerical quantifiers), variables (such as b) and the equality symbol $=$.

- The set of predicates $\{Black, White\}$

- The set of constants $\{B_1 \ldots B_k\}$, for some positive integer k.

Note that, since the set of constants is part of the language, we will always have to know how many observations we make before defining $\mathcal{L}$. The situations presented in Figure 3.2 and Figure 3.3 would both be described using a language where $k = 3$: we will use that particular language in later examples.

It is worth noting that we can describe this domain in a more intuitive way if we use range-restricted functions: the predicates $\{$*Black*, *White*$\}$ can be replaced with the range-restricted function *Colour*, ranging over $\{$*Black*, *White*$\}$.

3.2.1 Probability

We introduce probability into our language using an approach very similar to that of Bayesian Logic Programs (Kersting & De Raedt, 2000). Bayesian Logic Programs (BLPs) are collections of definite clauses, annotated with probability tables defining distributions over the possible values of the heads given all possible combinations of values for the bodies. Thus, BLPs define a distribution over possible worlds as a product of distributions associated with atomic formulas. In our language, we assign probabilities not to atomic formulas, but to facts, where a fact can be either:

- An atomic formula, or the negation of an atomic formula. (Note that these can contain free variables.)

- A **counting statement** describing how many objects exist in the world. We will define the counting statement $\# = i$ as follows:

  $$\# = i \;\equiv\; \exists x_1, \ldots x_i \;\; x_1 \neq x_2 \neq \ldots \neq x_i \bigwedge \neg\exists x_1, \ldots x_{i+1} \;\; x_1 \neq x_2 \neq \ldots \neq x_{i+1} \,.$$

  Thus, $\# = i$ states that there are exactly i objects in the world; counting statements are rather like numerical quantifiers with nothing to quantify over.

We describe distributions over facts in an intuitive way, using expressions of the form $P(X|\mathbf{Y}) = p$, where X is a fact, $\mathbf{Y}$ is a (possibly empty) conjunction of facts, and p is a probability (that is, a number in $[0, 1]$). Naturally enough, the expression $P(X|\mathbf{Y}) = p$ is taken to mean that the the probability of X being true given that $\mathbf{Y}$ is true is p. Thus, $P(X \leftarrow \mathbf{Y}) = p$ would be an alternative notation. Free variables are assumed to be universally quantified outside the algebraic expression. Deterministic sentences (where $P(X|\mathbf{Y}) = 1.0$) do not need to be annotated, as unannotated sentences are assumed to be deterministic.

It should be obvious that certain sets of facts represent mutually exclusive and exhaustive possibilities: we will call such possibilities **alternatives.** Examples of sets of alternatives include an atomic formula and its negation, a set of atomic formulas assigning all the possible values to a range-restricted function, and the set of $\# = i$ statements for $i \in [1, \infty]$. Let us consider a set of alternative facts, $\mathbf{X} = \{X_1 \ldots X_n\}$. In the interest of making sure that distributions over such sets are fully-defined and consistent, we find it convenient to introduce some syntactic sugar permitting us to define the distribution over a set of alternatives all at once. So, we write

$$P(\{X_1, \ldots X_n\}|\mathbf{Y}) = \{p_1, \ldots p_n\}$$

in place of the individual $P(X_i|\mathbf{Y}) = p_i$ statements, and make sure that all the p_is sum to 1.0. Facts with a probability of 0.0 can be left off the list, permitting us to define a distribution over the infinite set of $\# = i$ statements by dropping all but a finite number.

Let us now consider $\mathbf{Y}$, the set of facts restricting the situations where the $P(X_i|\mathbf{Y})$ statements apply. To ensure that the distribution over $\mathbf{X}$ is always fully specified, we want to write down similar probability-annotated statements to cover all other eventualities. We define a **conditional probability model** (CPM) for $\mathbf{X}$ as a set of statements defining distributions over $\mathbf{X}$ for every possible combination of alternatives to the facts in $\mathbf{Y}$. Like Kersting and De Raedt (2000), we do impose the constraint that all our probability distributions be defined using fully-specified conditional probability models. Following this constraint, if we define $P(\mathit{White}(b)|\mathit{Black}(b))$, we also have to define $P(\mathit{White}(b)|\neg\mathit{Black}(b))$.

Now, since we are adapting the possible worlds approach, any representation we define will be used for three purposes: to construct a knowledge base, to introduce evidence, and to form queries. We will now explain how we will restrict our language as we apply it to these three tasks.

3.2.2 Constructing a knowledge base

Since the knowledge base defines our distribution over possible worlds, we find it very useful to restrict our syntax: not just because we want to ensure that the distribution is consistent, but also because we would like the distribution to be simple to define. With this in mind, we place several restrictions on the probability-annotated statements in our knowledge base.

First of all, we impose an ordering on the functions and predicates of the language. Whenever we define a CPM for a fact involving a function or a predicate, the conditioning facts can only make use of those functions or predicates that precede this fact in the ordering. Thus, if our ordering is such that *Black* precedes *White*, then facts formed using the *White* predicate can depend on facts formed using the *Black* predicate, but not vice-versa.

The knowledge base must then contain the following:

- A distribution over the counting statements, such as

$$P(\{\# = 1,\ \# = 2,\ \# = 3,\ \# = 4\}) = \{0.4, 0.3, 0.2, 0.1\}$$

- For every predicate, a set of CPMs unambiguously defining the probability of the truth for every possible ground atomic formula constructed using that predicate. For single-argument predicates, this implies either one CPM where the argument is a universally quantified variable, or a set of statements dealing with every possible constant. (So, for the predicate *White*, we might define a CPM for *White*(b), or a set of CPMs handling each of *White*(B_1), *White*(B_2), and *White*(B_3); or maybe a hybrid consisting of a CPM for *White*(B_1) and a CPM for *White*(b) given that $b \neq B_1$.) For multi-argument predicates, the same reasoning applies to every argument.

The scenario portrayed in Figure 3.2 might be represented using the statements $P(\textit{White}(b)|\textit{Black}(b)) = 0.0$ and $P(\textit{White}(b)|\neg\textit{Black}(b)) = 1.0$.

- Analogously, for every function, a CPM defining a probability distribution over a set of values for every possible grounding of the function. For range-restricted functions, this set of values will be the range of the function; for conventional functions, it will be the set of constants, or at least some subset of this set.

  The scenario of Figure 3.2 could be represented using the statement $P(\{\textit{Colour}(b) = \textit{Black}, \textit{Colour}(b) = \textit{White}\}) = \{0.6, 0.4\}$.

The knowledge base might also contain some *alldiff* statements over the constants, where *alldiff*$(\mathbf{T})$ is shorthand for the set of all possible inequality statements that can be made about the terms in $\mathbf{T}$, so that

$$\textit{alldiff}(\mathbf{T}) \equiv \forall T_i, T_j \in \mathbf{T}\ ,\ \ i \neq j \Leftrightarrow T_i \neq T_j\ .$$

We allow for such statements so that we can impose a version of the unique names assumption over subsets of the constants in $\mathbf{T}$. However, we allow each constant to participate in only one *alldiff* statement.

The scenario portrayed in Figure 3.3 would contain the statement *alldiff*(B_1, B_2).

The knowledge base described thus far contains no statements dealing with the identity mappings of the constants. This is because, under this simplest approach, we assume the existence of some default definitions which do not need to be included explicitly. The exact nature of the default statements depends on which of the approaches of Section 3.1 we wish to adopt.

- If we decide to view the constants as samples drawn from the objects of the world, then, for every constant $I \in \mathbf{I}$, we will assume the existence of statements like

  $$P(I = x | \# = i) = 1/i$$

  for every possible i. By stating that the probability of I mapping to any given object x is the reciprocal of the number of objects in the world, these statements express the assumption that the constants are sampled using a uniform distribution over the objects in the world. We make this assumption because any other distribution would be rather hard to justify in view of the fact that the objects are a priori indistinguishable.

- If we decide to view the constants as incomplete observations of the objects in the world, we will assume the existence of the two statements

  $$P(\bigvee_{I \in \mathbf{I}} x = I) = o$$

  and

  $$P(\bigwedge_{I \in \mathbf{I}} x \neq I) = 1 - o$$

  for some $o \in [0, 1]$. The first statement states that the probability that the object x corresponds to some constant term I, and thus the probability that it has been observed is o; the second states that the probability that it corresponds to none of constant terms, and thus the probability that it has not been observed, is $1 - o$. The actual value of o does not matter in this setting.

The reason for this will be explained in Section 3.4, where we will describe how both types of statements can be translated into a probability distribution over possible worlds.

3.2.3 Introducing the evidence

We will introduce evidence as a conjunction of ground atomic formulas.

The evidence of Figure 3.2 or Figure 3.3, could be introduced using the atomic formulas *White*(B_1), $\neg$*Black*(B_1), *Black*(B_2),$\neg$*White*(B_2), *White*(B_3) and $\neg$*Black*(B_3). In range-restricted case, and the evidence would be introduced as *Colour*$(B_1) =$ *White*, *Colour*$(B_2) =$ *Black*, and *Colour*$(B_3) =$ *White*.

3.2.4 Forming queries

We will permit a query to be any sentence of our underlying logical language; our inference algorithm will be expected to return the probability that the sentence is true.

Possible queries for our example include $B_1 = B_2$ ("B_1 and B_2 refer to the same object") and $\exists b_1, b_2 \mathit{White}(b_1) \wedge \mathit{White}(b_2) \wedge b_1 \neq b_2$ ("There are at least two white objects in the world").

3.2.5 Overview

The language we have described so far is fairly restricted, and very similar to the Bayesian Logic Programs of Kersting and De Raedt (2000). The major difference (that they define distributions only over attributes, while we define them also over functions and counting statements) is a direct result of our dismissal of the unique names assumption. Now that we can no longer rely on that assumption to constrain certain aspects of possible worlds for us, we have to define something to take its place. The distributions over functions and counting statements are similar in format to those over the attributes, and this results in a fairly rich and general syntax. However, the language does not provide a rich and general mechanism for defining distributions over the identity mappings of the constants. The reason for this is that making such mappings dependent on other aspects of the possible world is somewhat problematic, as we shall see in Section 3.5, where we do extend that aspect of the representation.

3.3 Possible Worlds

Before we can show how a knowledge base of the type defined above encodes a distribution over possible worlds, we would like to describe a factored representation for possible worlds. Such a representation will permit us to factor the distribution in an analogous manner; we will then be able to relate the factors to the probabilistic statements in our knowledge base.

In PRMs (Pfeffer, 2000), possible worlds are represented as value assignments to a set of predicates and range-restricted functions. This is enough because, as we have explained in Section 2.1.2, these assignments are the only aspect of interpretations that can vary under Pfeffer's (2000) assumptions. The other aspects – the set of objects, and the mappings of both constants and functions – are fixed in every interpretation, and so can be ignored. When we remove the unique names assumption, and hence introduce identity uncertainty, term mappings become uncertain, and the size of the set of objects becomes variable. (It is not uncertain, since it can be deduced from the mappings.) Then, when we remove the assumption that there are no extraneous objects, we introduce uncertainty also over the number of objects.

In this section, we present two possible ways of representing the newly uncertain aspects of possible worlds. Both capture all the relevant ways in which the worlds may vary, while ignoring most isomorphisms. The first representation is very general: it is really just a slight variant of the standard first-order logic semantics. The changes we have made make it easier to relate this representation to our second representation, which is more compact, but can be used only in problem settings where the language has a finite number of terms. We consider it somewhat more intuitive in such settings, and make extensive use of it in later chapters. On the other hand, we find it much easier to use the first representation when discussing probability distributions: fortunately, as we will show, the relationship between the two representations is quite clear.

3.3.1 Simple Possible Worlds

As we have stated, this first representation is just a slight variation on how interpretations are usually described. We assume the number of objects to be finite in every ω, and introduce the variable $n = |\Delta^\omega|$ to keep track of the number of objects in the world. This variable can take on values from the interval $[1, \infty]$. We can then refer to the n objects using the indices 1 through n.

Now, for each predicate R of arity a, we define V_R as an a-dimensional array where every dimension is of size n. This array keeps track of the truth/falsehood of all the possible instantiations of the predicate. More formally,

$$V_R(i_1 \dots i_a) \quad = \quad \begin{cases} T & \text{if } (i_1 \dots i_a) \in R^\omega \\ F & \text{if } (i_1 \dots i_a) \notin R^\omega \end{cases}$$

We will refer to the set of all the entries of V_R as $\mathbf{V}_R$, and to the union of all the $\mathbf{V}_R$s of all the predicates simply as $\mathbf{V}$. This slight variant of the standard first-order logic semantics will allow us to treat all the $V_R[i_1 \dots i_a]$s as random variables ranging over $\{T, F\}$. It can be extended to functions in a very natural way: for any such function f, the V_f matrix will be filled not just with Ts and Fs, but with a wider range of possible values. For a range-restricted function, these values can include the elements of $Val[f]$; for a conventional function, they will be integers in the interval $[1, n]$.

For example, if a language contains only the unary predicates *Black* and *White*, and if $n = 2$, then $\mathbf{V} = \mathbf{V}_{\mathit{White}} \cup \mathbf{V}_{\mathit{Black}}$, and one possible assignment of values to $\mathbf{V}$ might be

$V_{White} = [T, F], V_{Black} = [F, T]$, corresponding to the state where

$$V_{White}(1) = T,\ V_{White}(2) = F,\ V_{Black}(1) = F,\ V_{Black}(2) = T\ .$$

If a language contains only the range-restricted function *Colour*, then $\mathbf{V} = \mathbf{V}_{Colour}$, and an assignment might be $V_{Colour} = [\mathit{White}, \mathit{Black}]$, corresponding to the state where

$$V_{Colour}(1) = \mathit{White},\ V_{Colour}(1) = \mathit{Black}\ .$$

Now, we will consider what happens when we the language contains functions. It is hard to think of a reasonable unconstrained function on balls, but let us assume that each ball has is decorated with a picture of another ball, and that this other ball may be referred to using the function *Picture*. We now have to expand $\mathbf{V}$ to include $V_{Picture}$. A possible assignment might be simply $V_{Picture} = [1, 1]$.

Now, let us consider the mapping for the constant terms of the language **I**. To represent this mapping, we introduce the **identity mapping function** δ^{ω}. This function takes a set of terms $\mathbf{T}$ and returns an n-dimensional vector where the ith component contains the set of terms corresponding to object i. More formally,

$$\delta^{\omega}(\mathbf{T}) \quad = \quad [\delta_1 \ldots \delta_n] \quad s.t.\ \delta_i = \{T : I \in \mathbf{T}, [T]^{\omega} = i\}\ .$$

Thus, $\delta^{\omega}(\{T_1, T_2\})$ can take on one of four values: $[\{T_1, T_2\}, \{\}]$, $[\{T_1\}, \{T_2\}]$, $[\{T_2\}, \{T_1\}]$, and $[\{\}, \{T_1, T_2\}]$. Notice that some of these values contain empty sets, representing those objects that are not referred to by either term. We will sometimes call these objects **unobserved** objects; similarly, the objects which are represented in the knowledge base will be referred to as **observed**.

We define the **identity mapping vector** δ as the result of applying δ^{ω} to the full set of all the constant terms of the language **I**. We then combine the three elements we have just described to get the following general definition:

Definition 3.1: A **simple possible world** is a tuple $\langle n, \delta, \mathbf{V} \rangle$, where

- n is the number of objects in the interpretation.
- $\delta = [\delta_1 \ldots \delta_n]$ is a vector of clusters of terms, where the ith cluster contains the terms that refer to the ith object.
- $\mathbf{V}$ is a set of arrays keeping track of the value assignments for all the possible applications of all the predicates and functions.

3.3.2 Compact Possible Worlds

In settings where the number of terms of the language is finite, we can use an alternative representation. This representation is more efficient, in that we no longer need to keep track of identity mapping vectors which may contain many empty sets. It also seems to be more intuitive. The key idea behind it is that the term-to-object mapping is represented as a set of $m \leq n$ equivalence classes, where each class contains all the terms (both functional and constant) that refer to the same object, and where the objects have no explicit names or indices beyond the equivalence classes themselves. As an illustration, consider Figure 3.4, where the knowledge base contains two constants, T_1 and T_2. As the number of objects in the possible world increases, so will the number of possible assignments to δ. Even when $n = 2$, there are already four possibilities, $[\{T_1, T_2\}\{\}]$, $[\{\}\{T_1, T_2\}]$, $[\{T_1\}\{T_2\}]$, and $[\{T_2\}\{T_1\}]$. However, there are actually only two ways in which the terms can be clustered together.

To formalize compact possible worlds, we introduce the **identity clustering function** ι^ω. This function takes a set of terms and partitions it into a set equivalence classes, where each such class contains all the terms that refer to the same (observed) object, so that

$$\iota^\omega(\mathbf{T}) = \{\mathbf{E}_1 \ldots \mathbf{E}_m\} \quad \text{where} \quad \bigcup_i \mathbf{E}_i = \mathbf{T},\ \forall i \neq j\ \mathbf{E}_i \cap \mathbf{E}_j = \varnothing$$

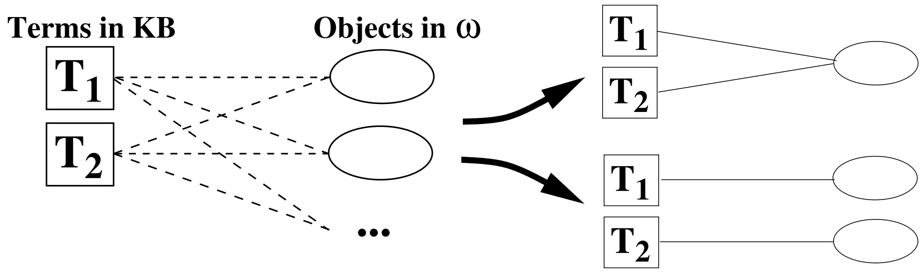


Figure 3.4. The term-to-object mappings possible when there are two terms. While there are many possible mappings, there are only two ways in which the terms can be clustered together into equivalence classes.

$$\forall T, T' \in \mathbf{T} \;\; [T]^\omega = [T']^\omega \iff \exists i \text{ s.t. } T, T' \in \mathbf{E}_i$$

We take ι to refer to the result of applying ι^ω to the set of all the terms of the language. (This set is required to be finite.) We call ι the **identity clustering**, and it ranges over all the possible **set partitions** (see Appendix B) of the set of terms. Where the constant terms are concerned, the relationship between ι^ω and δ^ω is as follows

$$\iota^\omega(\mathbf{T}) = \{\mathbf{E} : \exists i \, \delta^\omega(\mathbf{T})[i] = \mathbf{E}\}$$

Identity mappings can be transformed into identity clusterings in the following way. First, consider a language with no functions, so that all the terms of the language are already in δ. We impose an ordering on the objects, and pay attention only to those possible worlds that conform to this ordering. Each such world then represents a set of simple possible worlds: those which differ from it only in the way the objects are ordered. The ordering we impose makes use of the terms of the language: assuming an arbitrary way of ordering the terms, we order the objects according to the "earliest" term that maps to that object. Unobserved objects are placed at the end of the list—in fact, since we know (from n) how many unobserved objects there are, we can drop them altogether.

The ordering we impose is just a way of translating the equivalence classes into object indices. The unobserved objects, which are hinted at by n, but not represented amongst the equivalence classes, are a bit of a complication. We will refer to them using the indices $m+1$ through n. As an example, when $n = 2$ and there are only two constants, T_1 and T_2, two of the possible values of $\delta_{\mathbf{I}}$ are $[\{T_1\}\{T_2\}]$, and $[\{T_2\}\{T_1\}]$. When we impose the ordering, they will both become $[\{T_1\}\{T_2\}]$. For the two remaining values, $[\{T_1, T_2\}\{\}]$ and $[\{\}\{T_1, T_2\}]$, the imposed ordering would place the object corresponding to $\{T_1, T_2\}$ before the one corresponding to $\{\}$ (which could then be dropped from the list without any loss of information.)

The set of term clusters resulting from these transformations is the ι **corresponding** to the δ used.

Theorem 3.3.1: Given an identity clustering ι, let Δ_ι be the set of δs such that $\delta \in \Delta_\iota$ iff ι corresponds to δ. Then, given that n is the number of objects in the world, and m is the number of *observed* objects,

$$|\Delta_\iota| = \frac{n!}{(n-m)!}$$

In other words, for each identity clustering ι, there exist exactly $\frac{n!}{(n-m)!}$ corresponding δs.

Proof: An ι contains m clusters, representing m observed objects. Each of the corresponding δs will contain the same clusters, placed somewhere among the n objects. There are n ways of picking the index of the first object, $n-1$ ways of picking the index of the second, and so on all the way until the mth object, which can take any of the remaining $n-m+1$ indices. Putting it all together, there are

$$n * (n-1) \dots (n-m+1) = \frac{n!}{(n-m)!}$$

possible ways in which the m clusters might have been placed. ■

If we are using a language that contains functions, we alter the δ before re-ordering by adding functional terms to the appropriate locations according to the those elements of $\mathbf{V}$ that correspond to functional mappings, obtaining δ_f. This is a fully deterministic and reversible recursive process. As an example, consider the function mapping array $V_{\mathit{Picture}} = [2, 2]$, and the identity mapping

$$\delta = [\{T_1, T_2\}\{\}] \, .$$

We can use V_{Picture} to turn the identity mapping into

$$[\{T_1, T_2\}\{\mathit{Picture}(T_1), \mathit{Picture}(T_2)\}\}$$

and then into

$$[\{T_1, T_2\}\{\mathit{Picture}(T_1), \mathit{Picture}(T_2), \mathit{Picture}(\mathit{Picture}(T_1)), \mathit{Picture}(\mathit{Picture}(T_2))\}\}$$

and so on: this should make it clear that, in the general case, where there is an infinite number of functional terms, this will be an endless process. However, as we will demonstrate in Chapter 6, there do exist typed languages that have a finite number of terms without forbidding the use of functions.

It is worth pointing out that one consequence of using ι to keep track of all terms is that it implicitly keeps track of function mappings, so that, in compact possible worlds, $\mathbf{V}$ does not need to keep track of conventional functions explicitly. Thus, we can transform a simple possible world into a compact possible world by removing those elements of $\mathbf{V}$ that keep track of conventional function mappings, and using them to deterministically transform δ into a δ_f. This δ_f can then be turned into the corresponding ι. We say that a compact possible world **corresponds** to those simple possible worlds that can be mapped to it using this transformation. We then make the following formal definition:

Definition 3.2: A **compact possible world** is a tuple $\langle n, \iota, \mathbf{V} \rangle$, where

- n is the number of objects in the interpretation.

- $\iota = \{\iota_1 \ldots \iota_m\}$ is a set of clusters of terms, ordered using the names of the terms, where each cluster contains the terms which refer to some object that has been observed in the knowledge base.

- $\mathbf{V}$ is a set of arrays keeping track of the value assignments for all the possible applications of all the attributes.

Moreover, the tuple $\langle n, \iota, \mathbf{V} \rangle$ must correspond to a non-empty set of simple possible worlds.

The main effect of the final statements is to disallow ιs which suggest nonsensical function mappings, such as $\{\{B_1, B_2, \mathit{Picture}(B_1)\}\{\mathit{Picture}(B_2)\}\}$ where $\mathit{Picture}(B_1)$ and $\mathit{Picture}(B_2)$ are not equal even though B_1 and B_2 are.

Theorem 3.3.2: A compact possible world $\langle n, \iota, \mathbf{V} \rangle$ represents $\frac{n!}{(n-|\iota|)!}$ simple possible worlds.

Proof: Once we notice that $|\iota| = m$, the number of observed objects, this follows from Theorem 3.3.1. ■

Example 3.2: Consider a situation where there are three terms, B_1, B_2, and B_3, corresponding to three balls drawn from an urn. If we assume that $n = 3$ (i.e., the urn contains three balls) there will be $3^3 = 27$ possible values of δ. Figure 3.5 illustrates how these values of δ, represented here as observation sequences, line up with the five possible values of ι.

It is worth noting that, in a compact possible world, the values of conventional functions are not represented explicitly: the clusters in ι mix together functional terms and constants. However, the function mappings can be recovered from ι quite easily.

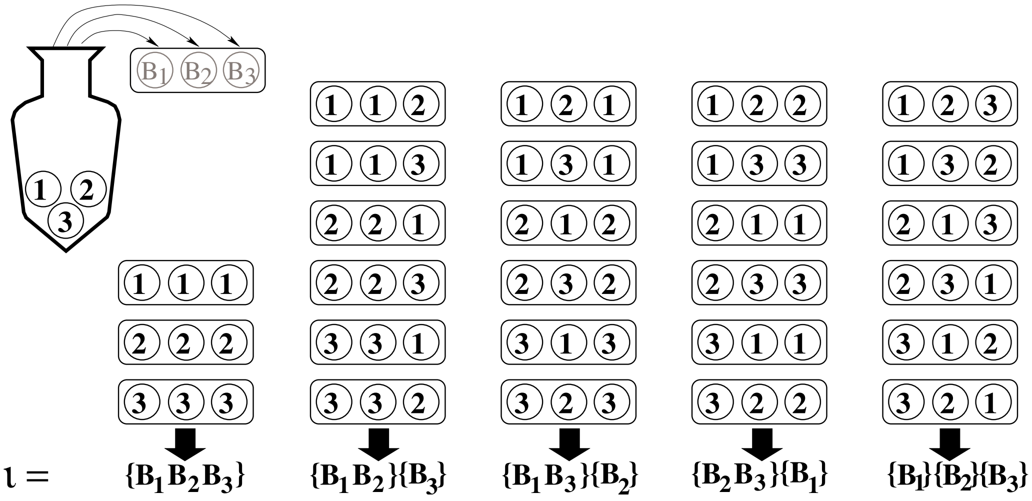


Figure 3.5. The twenty-seven observation sequences possible when three balls are sampled with replacement three times, divided into groups corresponding to each possible value of ι.

3.4 The Probability Distribution

When discussing a probability distribution over possible worlds, we find it convenient to use our simple possible worlds representation. As we will show in Section 3.4.2, this distribution can, if the language permits, then be extended to cover compact possible worlds using Theorem 3.3.2.

3.4.1 Simple Possible Worlds

What we need is a unique probability distribution over the simple possible worlds, or a $P(n, \delta, \mathbf{V})$ which assigns a probability to every possible combination of values of n, δ, and $\mathbf{V}$, and which, when summed over all these combinations, sums to 1.0. We will show how our knowledge base describes such a distribution by using the product rule to factor $P(n, \delta, \mathbf{V})$, and then explaining how each of the resulting distributions may be derived from the contents of the knowledge base.

First, let us impose an ordering on those aspects of possible worlds that correspond to the functions and predicates of the language: that is, the value mapping arrays in $\mathbf{V}$. We will call the ordered list of these arrays $\mathbf{E}$, and the ith component will be referred to as E_i. We constrain this ordering using the $P(X|\mathbf{Y})$ statements in the knowledge base by stating that the function or predicate used to form the fact X cannot occur earlier in the ordering than the functions and predicates used to form the conditions $\mathbf{Y}$. For example, if the knowledge base contains a statement of the form $(P(\mathit{White}(b)|\mathit{Black}(b))$, then, for $E_i = \mathbf{V}_{\mathit{White}}$ and $E_j = \mathbf{V}_{\mathit{Black}}$, it must be the case that $j \leq i$. We know that there exists at least one ordering that meets these constraints, because we imposed such an ordering before constructing a knowledge base in Section 3.2.2.

Now, since $\mathbf{E}$ contains all the components of $\mathbf{V}$, we can rewrite $P(n, \delta, \mathbf{V})$ as $P(n, \delta, \mathbf{E})$, and factor it in the following manner:

$$
\begin{aligned}
P(n, \delta, \mathbf{E}) &= P(n)P(\delta|n)P(\mathbf{E}|\delta, n) \\
&= P(n)P(\delta|n)\prod_{i=1}^{|\mathbf{E}|} P(E_i|E_{i-1}\dots E_1, \delta, n) \qquad (3.1)
\end{aligned}
$$

By trivial application of the chain rule, $P(n, \delta, \mathbf{V})$ will be unique and consistent as long as so are all of its factors. Let us consider each of these factors in turn, relating them to the distributions defined using the simple language of Section 3.2:

- $P(n)$ is clearly equivalent to the distribution over the counting statements, and so it is defined directly in every knowledge base.

- Each $P(E_i|E_{i-1}\dots E_1, \delta, n)$ is a distribution $P(V_A|\dots, \delta, n)$, where A is either an attribute or a function. The knowledge base must contain conditional statements fully defining both types of distributions; moreover, the ordering we have imposed will ensure that all the dependencies imposed by the knowledge base are reflected in the ordering. In other words, we will not need to refer to $E_{i+1}\dots E_{|\mathbf{E}|}$ when figuring out which conditions apply to E_{i+1}.

In the case of attributes, the definitions in the knowledge base define a distribution over values directly. Where functions are concerned, however, the knowledge base defines a distribution over terms when what we want is a distribution over objects. We relate the two by defining the probability that a ground functional term maps to object i as the sum of the probabilities of all the terms that refer to object i – that is, the terms in $\delta[i]$. Consider the grounded functional term $f(X_1 \ldots X_a)$, and assume that the identity mappings of the terms $X_1 \ldots X_a$ are $i_1 \ldots i_a$. We then state that

$$P(\delta_f(i_1 \ldots i_a) = i|\delta) \;=\; \sum_{T \in \delta[i]} P(f(X_1 \ldots X_a) = T)\,. \tag{3.2}$$

Example 3.3: Consider a ground functional term F, and the constants B_1, B_2, and B_3. Suppose that the user defines the distribution

$$P(\{F = B_1, F = B_2, F = B_3\}) = \{0.5, 0.3, 0.2\}$$

Then, if B_1 and B_2 both refer to object 1, and B_3 refers to object 2, F will have a $0.5 + 0.3 = 0.8$ probability of referring to object 1, a 0.2 probability of referring to object 2, and a 0.0 probability of referring to any other object.

- $P(\delta|n)$ is not defined directly: instead, it must be derived from those statements in the knowledge base that deal with the identity mappings of constants: that is, the implicit statements we have described in Section 3.2, and the *alldiff* statements. Since this process is not trivial, and since its exact nature depends on whether we think the constants are sampled or observed, we will expand on it in the next two sections.

Sampling

In this setting, terms are assumed to be sampled with replacement from a set of objects, as in Figure 3.2. Let us consider the term T. When applied to a single term, the identity mapping function δ^ω can return only n possible values: one for each object the term could map to. (For example, if $n = 2$, $\delta^\omega(T)$ can return only $[\{T\}\{\}]$ and $[\{\}\{T\}]$.) If we assume that terms are sampled uniformly from objects, each of these values should have a probability of $1/n$, so that $P(\delta^\omega(T)|n) = 1/n$. That is exactly what our implicit $P(T = y|\# = i) = 1/i$ statements encode.

When the knowledge base contains no *alldiff* statements, the identity mappings of the terms are independent, and the situation is very simple. Given that $P(\delta^\omega(I)|n) = 1/n$ for all constants $I \in \mathbf{I}$,

$$P(\delta|n) = P(\delta^\omega(\mathbf{I})|n) = \prod_{I \in \mathbf{I}} P(\delta^\omega(I)|n) = \left(\frac{1}{n}\right)^{|\mathbf{I}|}. \tag{3.3}$$

It is the independence assumption that permits us to factor $P(\delta^\omega(\mathbf{I})|n)$ so neatly. When the knowledge base does contain *alldiff* statements, the independence assumption no longer holds: if we know that *alldiff*($\mathbf{T}$), where $\mathbf{T} = \{T_i, T_2, \ldots, T_k\}$, then the knowledge that term T_1 maps to object i restricts our mapping choices for term T_2. While T_1's mapping is chosen from the full set of n objects, that of T_2 is chosen from a smaller set of $n-1$ objects from which object i has been removed. (We sometimes think of the terms $\mathbf{T}$ as being sampled *without replacement*.) Since we can no longer make an independence assumption, we do not factor the distribution over the identity mappings of $\mathbf{T}$, but instead define it as

$$\begin{aligned} P(\delta^\omega(\mathbf{T})|n) &= P(\delta^\omega(T_1)|n)P(\delta^\omega(T_2)|n-1)\ldots P(\delta^\omega(T_k)|n-k+1) \\ &= \frac{1}{n}\frac{1}{n-1}\cdots\frac{1}{n-k+1} \\ &= \frac{(n-k)!}{n!}. \end{aligned} \tag{3.4}$$

In general, the full set of constant terms of the language $\mathbf{I}$ can be split into a set of disjoint sets $\mathbf{S} = \{\mathbf{S}_1 \ldots \mathbf{S}_{|\mathbf{S}|}\}$, where each $\mathbf{S}_l$ is either a set of terms constrained by an *alldiff* statement, or a single term that is not involved in any such statements. (This partition is possible because we do not allow any term to participate in more than one disjoint statement.) We then define the distribution over the identity mappings of all the constant terms as

$$P(\delta^\omega(\mathbf{I})|n) \;=\; \prod_{l=1}^{|\mathbf{S}|} P(\delta^\omega(\mathbf{S}_l)|n)\;, \tag{3.5}$$

where the $P(\delta(\mathbf{S}_l)|n)$ distributions are themselves defined as in Equation 3.4.

Incomplete Observations

Let us now turn to the second scenario, where we try to observe all the objects in the world at once, but where there exists some probability of missing each object. (Figure 3.3 shows a ball and urn example of such a scenario.) We assume that terms observed simultaneously, as part of one image, are not equivalent, and we enforce this assumption by applying *alldiff* statements to the terms in each image. We also assume that terms not observed simultaneously do not affect each other in any way, so that their identity mappings are independent. Thus, the distribution over the identity mappings of all the objects can, as in Equation 3.5, be factored into components handling sets of terms constrained using *alldiff* statements (representing sets of simultaneous observations).

Consider one such set of simultaneously-observed terms, $\mathbf{S}_i$. Now, let us assume that the probability of successfully observing an object, and adding a constant that represents it to the language, is $o \in [0, 1]$ (as we have stated in the knowledge base). Let us also assume that the process of observing an object does not affect the observation probabilities of the objects observed simultaneously. Under these assumptions, the probability of observing $k \in [1, n]$ particular objects (and so missing the remaining $n - k$) is clearly

$$o^k(1-o)^{(n-k)}\;.$$

This expression defines a probability distribution over all the 2^n possible ways of observing and missing n objects. However, in our situation, the value of k is known (it is $|\mathbf{S}_i|$), so

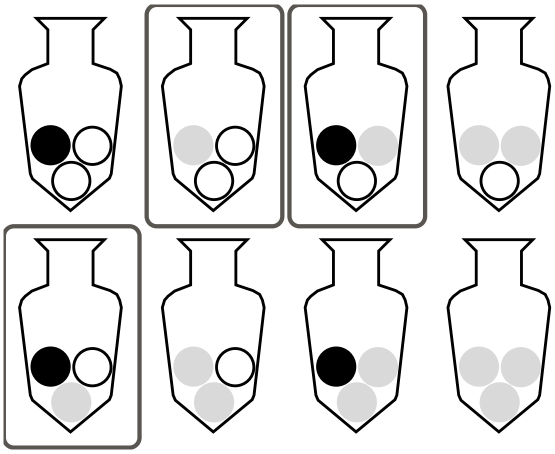


Figure 3.6. The eight images possible when three balls are observed simultaneously. Missed balls are grayed out. The outlines mark the three images possible given that we know two balls were observed successfully.

some of these possibilities are irrelevant. As an example, consider Figure 3.6, which displays the eight possible ways of observing and missing three balls, and demonstrates that only three that remain relevant when we know that exactly two balls were observed successfully. To obtain a probability distribution only over those images that are possible given that k is known, we have to normalize: that is, we have to divide the probability of each relevant image by the sum of these probabilities. However, since o is the same for all objects, all the images containing k objects will have the same probability, and so all we have to do is calculate the number of relevant images and take its reciprocal. Counting the images is simple: the images can differ in which k objects are picked from n, and also in the way in which these k objects are assigned variable names. Therefore, there are

$$\frac{n!}{(n-k)!}$$

possible images, since hat is the number of ways in which k ordered objects can be chosen from a set of n. Taking the reciprocal of this, we state that

$$P(\delta^{\omega}(\mathbf{S}_i)|n) = \frac{(n-k)!}{n!}$$

where k is the number of non-empty sets in $\delta^{\omega}(\mathbf{S}_i)$.

Of course, the observation model we have used here is very restrictive. We discuss how the assumption that the probability of observing an object is fixed in Section 3.5.

3.4.2 Compact Possible Worlds

Given a distribution over the simple possible worlds, the probability of a compact possible is the sum of the probabilities of all the corresponding simple possible worlds, or

$$P(n, \iota, \mathbf{V}) = \sum_{\delta \in \Delta_{\iota}} P(n, \delta, \mathbf{V})$$

The summation is not actually necessary, because of the following theorem:

Theorem 3.4.1: Given a compact possible world $\langle n, \iota, \mathbf{V} \rangle$, and any identity mapping $\delta' \in \Delta_{\iota}$ corresponding to the identity clustering ι,

$$P(n, \iota, \mathbf{V}) = \frac{n!}{(n-|\iota|)!} P(n, \delta', \mathbf{V})$$

Proof: We know from Theorem 3.3.2 that each compact possible world corresponds to $\frac{n!}{(n-m)!}$ simple possible worlds, where $m = |\iota|$. Since all these worlds are isomorphic, and since the probability of a simple possible world does not depend on object names, they all have the same probability. It is therefore enough to calculate the probability of $\langle n, \delta', \mathbf{V} \rangle$ and multiply it by $\frac{n!}{(n-|\iota|)!}$. ■

3.5 Extending the Language

In this section, we consider a few ways of enriching the simple language we have been using throughout this chapter by allowing for more varied distributions over the identity mappings.

There are a few simple extensions that can be made without much effort. It is quite easy, for example, swap the ways in which identity mappings are represented for constant and functional terms. We can represent distributions over function mappings as sampling distributions: all we have to do is introduce statements like $P(T = y|\# = i) = 1/i$ where T is a functional and not a constant term, and thus define the probability of each function mapping as $1/n$. We could also define the distribution over the identity mappings of some terms as a distribution over other terms, and map this to a distribution over objects as we do for functions, using Equation 3.2. The discussion of these possibilities has been delayed until this section only in an effort to keep the earlier explanations simple.

The one real extension we want to make is allowing the identity mappings of terms to depend on the attributes of those terms. One example of this would be letting the probability that a ball is sampled (or observed) depend on its *Colour*, or even on the *Colour* of its *Picture*. Such dependencies can clearly be expressed in terms of the attributes and functions of the language, by rewriting the old implicit statements along the lines of

$$P(B_2 = x|\mathit{Colour}(\mathit{Picture}(x)) = \mathit{White}, \# = i) = p$$

or

$$P(\bigvee_{I \in \mathbf{I}} x = I|\mathit{Colour}(x) = \mathit{Black}) = o\,,$$

depending on whether objects are assumed to be sampled or observed. However, some complications will arise in each scenario: we discuss them below.

One issue that arises in both scenarios is that, when we make term mappings dependent on other aspects of possible worlds, we have to rework Equation 3.1 slightly, and we have to ensure that an acyclic factorization is still possible. Therefore, we impose an additional constraint on the knowledge base: the distributions over those attributes and functions used to represent the dependencies of the constant term mappings cannot refer to objects by name. Thus, if we make object mappings dependent on $\mathit{Colour}(\mathit{Picture}(x))$, we can no longer state that $P(\mathit{White}(B_1)) = 0.6$ and $P(\mathit{White}(B_2)) = 0.3$: we have to use variables, and define a single distribution over the attributes of unnamed objects, as in $P(\mathit{White}(b)) = 0.6$. In addition, the distribution over *Picture* can no longer refer to terms: we have to express it as a sampling distribution, like $P(\mathit{Picture}(x) = y, \# = i) = 1/i$. It should be obvious that this constraint, just like the old ordering constraints, aims to avoid cyclic dependencies among the probability distributions. Then, when we let δ become one of the elements of the ordered list $\mathbf{E}$ in Equation 3.1, we can choose its position in the ordering accordingly. In this section, we will use $\mathbf{X}$ to refer to the subset of $\mathbf{V}$ corresponding to the attributes and functions that precede δ in the ordering – in other words, the subset of $\mathbf{V}$ on which δ may depend.

Sampling

Making the sampling probability of an object depend on its attributes is not simply a question of placing statements like $P(B_2 = x|\mathit{Colour}(x) = \mathit{White}, \# = i) = p_{\mathit{White}}$ and $P(B_2 = x|\mathit{Colour}(x) = \mathit{White}, \# = i) = p_{\mathit{Black}}$ into the knowledge base. The reason for this is that we cannot, in general, guarantee that such statements will define a probability distribution; we cannot guarantee that the probabilities will sum to one. Our statements might work very well if $p_{\mathit{White}} = 0.2$, $p_{\mathit{Black}} = 0.3$, and there are two black and two white balls in the world, or just five white ones: however, any other combination of balls will result in an inconsistent distribution. The problem occurs because we are trying to define a distribution over *all the objects in the world*, and, since objects are a semantic concept, the syntax of our logic does not allow us to refer to them (and their properties) explicitly.

One possible solution might be to partition the objects in the world into disjoint sets, and define a distribution over these sets. Sampling would then be a two-step process: one of the sets would be chosen according to the distribution, and an object in that set would then be chosen uniformly at random. The resulting distribution is quite easy to express in our language once we allow facts to include numerical quantifiers that count up the objects with some particular property, like $\exists_i y \mathit{Colour}(y) = \mathit{White}$. For example, we could partition the set of balls into *White* and *Black* ones, and define p_{White} and p_{Black} so that they sum to 1.0. Our troublesome distribution could then be expressed using the statements

$$P(B_2 = x | \mathit{Colour}(x) = \mathit{White}, \exists_i y\ \mathit{Colour}(y) = \mathit{White}) = \frac{p_{\mathit{White}}}{i}$$

and

$$P(B_2 = x | \mathit{Colour}(x) = \mathit{Black}, \exists_i y\ \mathit{Colour}(y) = \mathit{Black}) = \frac{p_{\mathit{Black}}}{i} .$$

This is the approach adopted by Getoor et al. (2001a) when they define distributions over uncertain reference slots.

Another solution would be to treat probabilities like p_{White} and p_{Black} as individual sampling probabilities, and explicitly normalize them with the help of these numerical quantifiers, resulting in definitions like

$$P\left(\begin{array}{lll} B_2 = x & | & \mathit{Colour}(x) = \mathit{White}\ , \\ & & \exists_i y\ \mathit{Colour}(y) = \mathit{White}\ , \\ & & \exists_j z\ \mathit{Colour}(z) = \mathit{Black} \end{array}\right) = \frac{p_{\mathit{White}}}{i \times p_{\mathit{White}} + j \times p_{\mathit{Black}}} . \qquad (3.6)$$

The denominator on the right-hand side is normalizing the probabilities p_{White} and p_{Black} to account for the number and types of balls present in the world.

In Chapter 5 and Chapter 6, we will be using this second approach. However, we introduce an alternative notation which assumes that the normalization is done automatically.

Instead of defining sampling probabilities, we will define **sampling potentials**, written as $S(T = x|\mathbf{Y}) = s$, which are essentially unnormalized probabilities. Analogously to our probability notation, such statements mean that the "potential" of the identity mapping $T = x$ is s given that the facts in $\mathbf{Y}$ are true. Note that the statement $T = i$ represents the event that $\delta^\omega(T)[i] = \{T\}$, or that δ^ω maps T to i. We then define the distribution over the identity mapping of T as

$$P(\delta^\omega(T)[i] = \{T\}|\mathbf{X}, n) \quad = \quad \frac{S(T = i|\mathbf{X}, n)}{\sum_{j=1}^{n} S(T = j|\mathbf{X}, n)} \tag{3.7}$$

It should be clear how the denominator corresponds to that of Equation 3.6.

Any inference algorithm we use will need to enforce Equation 3.7. A naive algorithm might need to carry out the summation. Note, however, that the normalization constants of any two possible worlds $\langle n, \delta, \mathbf{V}\rangle$ and $\langle n, \delta', \mathbf{X}\rangle$ differing only in δ will be the same. Thus, they will cancel each other out in a calculation of their probability ratio.

Incomplete Observations

Now, let us consider what happens when we make the observation probabilities dependent on object attributes. This might involve statements like

$$P(\bigvee_{I \in \mathbf{I}} x = I | \mathit{Colour}(x) = \mathit{White}) = o$$

which states that the probability that object x corresponds to one of the constants of the language **I**, and has therefore been observed, is o if the *Colour* of the object is equal to *White*. We have assumed that the probability that an object is observed is independent of the other objects. Because of this independence assumption, and in contrast to the sampling scenario, (implicit) statements like this one do in fact define the probability of an object being sampled in a consistent way.

Nevertheless, permitting observation probabilities depend on object attributes does lead to complications. As has been explained in Section 3.4.1, observation probabilities need to be normalized to account for the fact that we know that exactly k of the n objects were observed. This involves summing over all the relevant identity mappings. When all objects are equally likely to be observed, we do not have to reason about the probabilities of all these relevant mappings, but can simply use combinatorics to calculate the number of such mappings; however, once we permit observation probabilities to vary, we do have to count up the actual probabilities. In the worst case, we will have to generate all $\frac{n!}{(n-k)!}$ possible images and sum up their probabilities. Let us define $\mathbf{D}$ as the set of identity mappings corresponding to these relevant images. The probability of the identity mapping $\delta^\omega(\mathbf{T})$ of a set of terms $\mathbf{T}$ in a single image will then be

$$P(\delta^\omega(\mathbf{T})|\mathbf{X}, n) \quad = \quad \frac{\prod_{i=1}^{n} P_{OBS}(\delta^\omega(\mathbf{T})[i]|\mathbf{X})}{\sum_{\delta' \in \mathbf{D}} \prod_{i=1}^{n} P_{OBS}(\delta'^\omega(\mathbf{T})[i]|\mathbf{X})} \, .$$

Here, $\delta^\omega(\mathbf{T})[i]$ corresponds to the ith element of the $\delta^\omega(\mathbf{T})$ vector, and thus to an empty set iff the ith object has been unobserved; and $P_{OBS}()$ expresses the observation probability:

$$P_{OBS}(\delta^\omega(\mathbf{T})[i]|\mathbf{X}_i) \quad = \quad \begin{cases} v & P(\bigvee_{T \in \mathbf{T}} x = I|\mathbf{X}) \quad \text{if} \quad \delta^\omega(\mathbf{T})[i] \neq \{\} \\ & P(\bigwedge_{T \in \mathbf{T}} x \neq I|\mathbf{X}) \quad \text{if} \quad \delta^\omega(\mathbf{T})[i] = \{\} \end{cases}$$

It is, again, worth noting the normalization constants of any two possible worlds differing only in δ will cancel each other out in a calculation of their probability ratio.

3.6 Dynamic Domains

There exist many problem domains which involve tracking moving objects across images collected at different points in time. These objects might correspond to vehicles, or bombs, or people walking across a particular courtyard. The sensors used may be imperfect, perhaps like the sensors we have considered in Section 3.4.1: however, that is not what we want to focus on here. Instead, we will focus on another interesting aspect of such domains, and that is the fact that moving objects may move in and out of the field

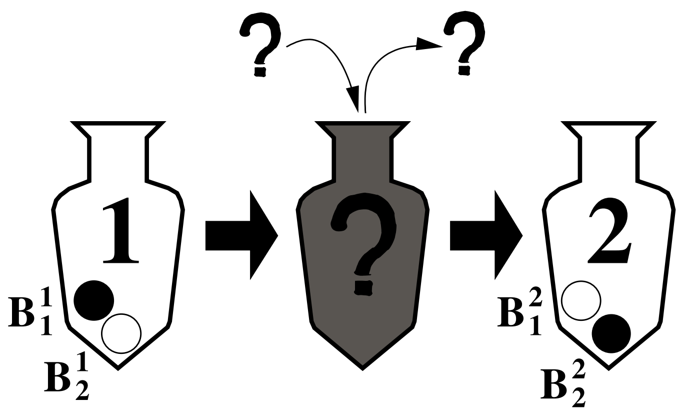


Figure 3.7. A dynamic version of the ball and urn domain, where complete observations of the contents of the urn are taken at different time-steps, but the set of balls inside can change over time. The observations made at time-steps 1 and 2 are represented by the numbered urns. The B_i^ts are terms representing the balls observed, where t corresponds to the time-step, and i are indices used to distinguish different observations.

of view, and therefore different objects may go unobserved at different times, even if the sensors are perfect.

As usual, we will illustrate the problem using a ball-and-urn example. Consider the situation shown in Figure 3.7, where observations of the contents of the same urn are taken at two subsequent time-steps, and there is a possibility that these contents may have changed in the meantime. We will assume that the observations were made perfectly, so that each object was observed exactly once at each time. The knowledge base will, therefore, contain at least the statements $\textit{Colour}(B_1^1) = \textit{Black}$, $\textit{Colour}(B_2^1) = \textit{White}$, $\textit{Colour}(B_2^1) = \textit{White}$, $\textit{Colour}(B_2^2) = \textit{Black}$, $\textit{alldiff}(\{B_1^1, B_2^1\})$, and $\textit{alldiff}(\{B_1^2, B_2^2\})$.

Formally, we define a dynamic domain as one where all the constants in the knowledge base are indexed by time. We will refer to the set of constants associated with a time t as $\mathbf{I}^t$. Thus, if we are considering time-steps $1 \ldots u$, the full set of constants $\mathbf{I}$ is a union of the disjoint sets $\mathbf{I}^1 \dots \mathbf{I}^u$. We use *alldiff* statements to impose unique names assumptions on the terms $\mathbf{I}^t$ associated with each time t, implying that no object can be observed more than once at any given time. The identity mapping function can be applied to each set $\mathbf{I}^t$ independently, to get $\delta^\omega(\mathbf{I}^t)$; however, since this notation is rather unwieldy, we will use δ^t to signify $\delta^\omega(\mathbf{I}^t)$. The set of terms from $\mathbf{I}^t$ referring to object i can is then $\delta^t[i]$.

If we make the Markov assumption, we can factor the distribution over δ as follows

$$P(\delta|\mathbf{X}, n) \quad = \quad P(\delta^1|\mathbf{X}, n) \prod_{t=2}^{u} P(\delta^t|\delta^{t-1}, \mathbf{X}, n) \, .$$

The prior distribution $P(\delta^1|\mathbf{X}, n)$ is just a distribution over δ, and can be defined using any of the methods already listed in this chapter. As for $P(\delta^t|\delta^{t-1}, \mathbf{X}, n)$, we will assume that the probabilities of the different objects leaving or entering the field of vision are independent of each other, so that we can factor it into distributions corresponding to the different objects, like $P(\delta^t[i]|\delta^{t-1}[i], \mathbf{X}, n)$. Since we have assumed that each object can be observed only once at each point in time, every $\delta^t[i]$ can either be a singleton set $\{I\}$ (which means that, at time t, the object i was observed as term I) or the empty set $\{\}$ (which means that the object was not observed at time t). There are thus two different types of distributions to be defined,

$$P(\delta^t[i]|\delta^{t-1}[i] = \{\}, \mathbf{X}, n)$$

and

$$P(\delta^t[i]|\delta^{t-1}[i] = \{I^{t-1}\}, \mathbf{X}, n) \, .$$

Obviously, the first pertains to objects that went unobserved at the previous time-step, and the second to those that were observed as term I^{t-1}. In both cases, $\delta^t[i]$ will range over $\{\}$ (the possibility that the object leaves – or fails to enter – the field of view), as well as over all of $\mathbf{I}^t$. Sampling distributions seem like a fair way to define such distributions. A *White* ball from time T^{t-1} might have an equal probability of co-referring with any one of the *White* balls from time T^t; if we enriched the language to include some description of location in the image, we might use that to weight the distribution.

Let us now briefly discuss how the $P(\delta^t|\delta^{t-1})$ distributions might translate into probabilities over possible worlds. As in Section 3.4.1, there is the problem that these distributions, by defining the probability that each object is referenced by a term independently for each object and each point in time, together define a distribution over all the different possible combinations of observed and unobserved objects; meanwhile, we know exactly how many objects were observed, because we know how many terms the knowledge base contains. So, when deriving the probabilities of individual possible worlds, we have to normalize by summing over all the possible ways in which those particular numbers of terms might have been observed at each point in time. Then, if the individual distributions are defined in terms of sampling potentials, we also have to normalize to account for that as well, as we did in in Section 3.5. Fortunately, if we are only interested in calculating probability ratios, the normalization constants will cancel as we have described before.

3.7 Inference

Let us now briefly discuss inference. General inference in the presence of identity uncertainty will involve considering all the possible term-to-object mappings. Thus, it may become necessary to sum over the very large state space of all the possible values of the identity mapping δ, or at least the identity clustering ι. As the number of instances in the knowledge base grows, this calculation quickly becomes intractable. We have therefore developed an approximate algorithm based on Markov Chain Monte Carlo (MCMC). MCMC methods often depend on the calculation of probability ratios, which makes them particularly appropriate when handling the extended languages of Section 3.5.

Chapter 4

Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a well-known method for sampling the elements of a large state space according to a probability distribution. It is commonly used to approximate an expectation w.r.t some distribution $\pi(x)$, especially when the state space of x is too large to sum over, or when sampling from $\pi(x)$ directly is very difficult. The weighted sum (or integral) over all the values of x is approximated using a sample average, where samples from $\pi(x)$ are generated using an appropriately constructed Markov chain.

In this chapter, we provide a formal definition of a Markov chain, and outline some useful MCMC concepts, as presented by Gilks, Richardson, and Spiegelhalter (1996) and Neal (1993). We describe two common MCMC methods, Metropolis-Hastings and Gibbs sampling, and then go on to explain how the running time of the algorithm may be bounded using a flow network based approach. This approach is then used to provide original loose bounds for two inference problems relevant to this dissertation. The first problem is that of Bayesian network Gibbs sampling, and it is relevant because RPM inference can be translated into Bayesian network inference; we find that the running time of our MCMC algorithm is dependent on the ratio of maximum and minimum CPF values, as well as on a property of the underlying graph. The second problem is that of weighted bipartite matching, and it is relevant because some identity-uncertain domains can be viewed as probabilistic bipartite matching problems; we find that the running time of our MCMC algorithm is proportional to the ratio of maximum and minimum "weights" or local match probabilities.

4.1 The Markov Chain

Consider a sequence of random variables, $\{x^0, x^1, \ldots\}$. Such a sequence is called a Markov chain if it possesses the **Markov property**: the property that, given a present state, the future state is independent of all the past states. More formally,

$$P(x^{t+1}|x^t, x^{t-1}, \ldots, x^0) = P(x^{t+1}|x^t)$$

The conditional probability distributions $P(x^{t+1}|x^t)$ (which we will, from now on, write as $P_{t+1}(.|.)$) describe the **transition functions** of the Markov chain. When these functions do not depend on the value of t, the chain is **homogeneous**, with a single transition function. A homogeneous chain can be characterized using this transition function, $T(x'|x)$, and the distribution over x at time 0, $P_0(x)$. The distribution over x at later points in time is simple to calculate. In this dissertation, we focus exclusively on discrete Markov chains (with countable state spaces), where

$$P_{i+1}(x') = \sum_x P_i(x)T(x'|x) . \tag{4.1}$$

When $P_{i+1}(x) = P_i(x) = \pi(x)$, $\pi(x)$ is a **stationary distribution** of the Markov chain. After constructing a chain (that is, defining $T(x'|x)$ and $P_0(x)$), we can prove that it has the desired stationary distribution by proving that Equation 4.1 holds for all x and i. However, it is often easier to demonstrate that the more restrictive **detailed balance condition**

$$\pi(x)T(x'|x) = \pi(x')T(x|x')$$

holds for all x. Chains which satisfy this condition are known as **reversible**.

By definition, once a chain reaches the stationary distribution (i.e., once $P_i(x) = \pi(x)$), it will stay there for all subsequent values of i, and can be used to generate samples from $\pi(x)$. We want to make sure that any chain we construct will converge to the correct distribution, no matter what starting state (or $P_0(x)$) we choose. In order to prove that it is so, we must show that the chain satisfies three properties:

- It is **irreducible**: that is, every pair of possible states in x is connected by a finite path of non-zero probability.
- It is **positive recurrent**: that is, the expected time to revisit a state is finite. This is true of any irreducible chain with a stationary distribution, and therefore of any irreducible chain which meets the detailed balance condition for some $\pi(x)$.
- It is **aperiodic**: that is, it has no periodic states. A state is periodic if the probability of revisiting it in n steps is non-zero only if $n = kd$, where $k, d > 1$ are integers (d is known as the **period** of the state). A chain is naturally aperiodic if all its states have non-zero self-loop probabilities ($\forall x\; T(x|x) > 0$)

Theorem 4.1.1: Positive recurrent, aperiodic chains are **ergodic** with a single stationary distribution (here, $\pi(x)$). If $E_\pi[f(x)]$ is the expectation of function $f(x)$ with respect to $\pi(x)$, then, for an ergodic chain,

$$P\left[\frac{1}{N}\sum_{t=0}^{N} f(x_t) \rightarrow E_\pi[f(x)]\right] \;=\; 1\,.$$

That is to say, expectations can be approximated by drawing samples from an ergodic chain with the appropriate distribution (Gilks et al., 1996).

To summarize: when proving that a chain is ergodic, we will demonstrate three things:

- That every pair of states is connected by a path of non-zero probability, and so the chain is irreducible.

- That the detailed balance condition holds, and so the chain is positive recurrent.

- That every state has a non-zero self-loop, and so the chain is aperiodic.

There are several well-known templates for constructing reversible Markov chains. We will now outline two of them, describing the transition functions $T(x'|x)$ algorithmically.

4.1.1 Metropolis-Hastings

In the **Metropolis–Hastings** method (M-H), each chain transition is constructed in two steps:

- A candidate next state x' is generated from the current state x using a (more or less arbitrary) **proposal distribution** $q(x'|x)$.

- The transition to x' is not automatic, but occurs with an **acceptance probability** defined as

$$\alpha(x'|x) = min\left(1, \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)}\right)$$

It should be clear that the detailed balance condition holds w.r.t. $\pi(x)$ for any proposal distribution: when $x' = x$, it holds trivially, and when $x' \neq x$, it is easy to show that $\alpha(x'|x)q(x'|x)\pi(x) = \alpha(x|x')q(x|x')\pi(x')$. The main constraint on the proposal is that it should suggest enough transitions to make sure that all pairs of states are connected by some possible path.

It is not necessary that all the variables that make up the state x be updated simultaneously. **Single-component** M-H alters the variables one by one; as long as the variables to be altered are selected using some random distribution, detailed balance will still hold. Any factors of $\pi(x)$ that do not refer to variables changed by the transition cancel in the ratio $\pi(x')/\pi(x)$, simplifying the calculation. This is all still true if we generalize single-component M-H and let the algorithm update sets of variables at a time.

Lemma 4.1.2: The detailed balance condition holds for any chain $\mathcal{M}$, which, at each step, has a non-zero probability of updating each of the sets $s_1 \ldots s_m$, where $\forall i\, s_i \subset x$ and $\bigcup_i s_i = x$.

Proof: Let the probability of updating set s_i be p_i, and let the distribution used to propose new s_i values be $q_i(x'|x)$. The transition probability $T(x'|x)$ for an $x \neq x'$ where the set of changed variables is $c = x - (x \cap x')$ is then

$$T(x'|x) = \sum_{i:c\subset s_i} p_i q_i(x'|x)\alpha(x'|x)$$

and it is easy to see that

$$\pi(x) \sum_{i:c\subset s_i} p_i q_i(x'|x)\alpha(x'|x) = \pi(x') \sum_{i:c\subset s_i} p_i q_i(x|x')\alpha(x|x)$$

by comparing the components corresponding to each i in turn. ■

4.1.2 Gibbs Sampling

The Gibbs sampler is a special case of single–component Metropolis–Hastings, in which the proposal distribution for each variable x_i is simply the conditional distribution of x_i given the current values of all the other variables ($x_{.-i}$), $P(x_i|x_{.-i})$. Since $\pi(x) = P(x_i|x_{.-i})P(x_{.-i})$ and $\pi(x') = P(x_i'|x_{.-i})P(x_{.-i})$, the acceptance ratio, and hence the acceptance probability, is always 1.0. Thus, the new value for x_i is simply drawn from

$$q(x_i'|x) = P(x_i|x_{.-i}) \ .$$

It seems natural to apply Gibbs sampling to Bayesian networks (Pearl, 1988), where each variable x_i is conditionally dependent only on the variables in its Markov blanket (its parents $Pa[x_i]$, its children y_j, and its children's other parents), so that

$$P(x_i|x_{-i}) = \alpha P(x_i|Pa[x_i])) \prod_j P(y_j|Pa[y_j]) \ . \tag{4.2}$$

This method is very simple and also *local*: transitions are generated referring only to parts of the model directly connected to the variable in question. So, for non-dense Bayesian networks, the cost per transition is independent of model size. However, Gibbs sampling has the drawback that all possible values of x_i must be considered at every step, which can be time-consuming when x_i has a large domain.

4.2 Running Time

The running time of a Markov chain is dependent on a couple of factors. It is obviously dependent on the time needed to perform each transition; this is usually easy to analyze. It is also dependent on how long it takes for the chain to **mix**, or reach the stationary distribution. This **mixing time** is of practical as well as theoretical value, as most MCMC applications **burn-in** the chain by running it until it mixes before taking any samples. To formalize the concept, we first define an error metric, the **variational distance**:

$$\delta_t = \max_x \frac{1}{2} \sum_x |\pi(x) - P_t(x)|$$

The mixing time is then the time it takes for δ_t to shrink so that it meets some error bound ϵ,

$$\tau(\epsilon) = \min\{t : \delta_{t'} \leq \epsilon, \forall t' \geq t\}$$

(It can be proved that δ_t is non-increasing with t, so the inequality is not really necessary.)

In Section 4.2.2, we explain how the mixing time can be analyzed, and then present the bounds we have found for two problems related to ours: general Bayesian network inference and weighted bipartite matchings. In both cases, it can be shown that, under some assumptions, the mixing time grows polynomially with problem size (defined as the number of nodes in the case of Bayesian network inference, and as the number of objects to be matched in the case of bipartite matching.) This is encouraging, although the assumptions called for in the Bayesian network case unfortunately include some assumptions about the structure of the network.

4.2.1 Experimental Evaluation of Mixing Time

Unfortunately, even when theoretical bounds are available, they are not always useful when choosing a burn-in time, since they can be quite loose, at least in their constant factors. Sometimes, theoretical bounds are so loose as to be useless; they might, for example, suggest that the mixing time is proportional to the size of the state space of x. Since MCMC is usually used to avoid summing over a too-large state space, this is not helpful.

In such situations, it is best to investigate the mixing time experimentally. When handling a small problem, where enumerating all values of x is feasible, we might simply calculate $\pi(x)$ and then test how long chains take to reach a certain variational distance. In slightly larger problems, we might, instead, calculate an expectation $E_\pi[f(x)]$ exactly, and compare it to the approximation found by the chain. If the problem we hope to handle is a large one, but it is possible to scale its size, we might apply one of these methods to various smaller versions of the problem, and then estimate the desired mixing time by extrapolating our results. In situations where we cannot calculate the answer exactly – this includes most of the useful ones – we can use the method presented in Gelman (1996). This method involves several chains, which are started at over-distributed points in the state space. As the chains run, the method tracks both the between-chain variance B, and the within-chain variances W, of some quantity ψ. These two variances combine to give an estimate of the actual variance of ψ, which is then compared to W. When the two are comparable, the chains are assumed to have converged.

4.2.2 The Flow Network Approach

There are several methods for bounding the mixing time, $\tau(\epsilon)$. Here, we describe the **flow network** approach, and then demonstrate how it can be used to provide bounds for Gibbs Bayesian network inference and weighted bipartite matchings.

The general idea behind this approach, which was developed by Sinclair (1992), is to express the state space of the Markov Chain as a flow network, so that the **chain states** become **network vertices** and the **state transitions** become **network edges**. (We will be using the letters s,t, and x to represent vertices, and either the letter e or vertex pairs such as (x, x') to represent edges.) Some further properties of the network are defined as follows:

- The **demand** $D(x, x')$ between two vertices x and x' is $\pi(x)\pi(x')$.
- The **capacity** $Q(x, x')$ across an edge (x, x') is $\pi(x)T(x'|x)$. (And also $\pi(x')T(x|x')$, when the detailed balance condition holds.)
- A **flow** f is a function which, for every pair of vertices (s, t), picks a single path from s to t (known as the **canonical path**) along which all the demand $D(s, t)$ will be sent.
- The **edge flow** $F_f(x, x')$ across the edge (x, x') is the sum total of the demand sent across that edge. To calculate it, we need to sum over all the (s, t) pairs that, for a particular flow f, send demand across (x', x). If $p_f(x, x')$ is the set of such pairs, then $F_f(x, x') = \sum_{(s,t)\in p_f(x,x')} D(s, t)$.
- The **length** $l(f)$ of a flow f is the length of the longest path in that flow.
- The **cost** $\rho(f)$ of a flow f is $\max_{(x,x')} \frac{F_f(x,x')}{Q(x,x')}$, so that

$$\rho(f) = \max_{(x,x')} \frac{\sum_{(s,t)\in p_f(x,x')} \pi(s)\pi(t)}{\pi(x)T(x'|x)} . \tag{4.3}$$

Theorem 4.2.1: For any flow f, we can upper-bound the mixing time as follows (Sinclair, 1992).

$$\tau(\epsilon) \leq \alpha\rho(f)l(f)(\ln \pi(x)_{min}^{-1} + \ln \epsilon^{-1}) \tag{4.4}$$

The hardest part is usually bounding $\rho(f)$, as defined by Equation 4.3. One helpful technique involves the use of an **encoding** $\nu_{(x,x')}(s,t)$, an injective mapping which returns a state that somehow encodes all of the state information present in s and t, but not in x or x'. In situations where an encoding can be defined, we can try to derive a constraint along the lines of

$$\pi(s)\pi(t) \quad \leq \quad \gamma\pi(x)\pi(\nu_{(x,x')}(s,t)) \tag{4.5}$$

where γ is a constant independent of the values of x, x', s, and t.

Consider a flow where all the canonical paths change the different components of s to their t values in some predetermined order. Then, for any pair x and x' of adjacent states along the canonical path from s to t, x and x' will agree with t on those aspects of the state that have already been changed, and with s on those aspects that have not yet been touched. Thus, given x and x', we will know the "early" components of t, as well as the "late" components of s. This will place a clear constraint on $p_f(x,x')$, the set of (s,t) pairs sending demand across (x,x'): each pair in $p_f(x,x')$ will be equivalent to a different set of values for those components of s that come before x, and for those components of t that come after x'. The encoding $\nu_{(x,x')}(s,t)$ will have to encode that very set of values. Then, when we substitute Equation 4.5 into Equation 4.3, we will get

$$\begin{aligned} \rho(f) \quad &\leq \quad \max_{(x,x')} \frac{\gamma\pi(x)\sum_{(s,t)\in p_f(x,x')}\pi(\nu_{(x,x')}(s,t))}{\pi(x)T(x'|x)} \\ &\leq \quad \max_{(x,x')} \frac{\gamma}{T(x'|x)} \,. \end{aligned} \tag{4.6}$$

The summation disappears because it turns into a summation over a disjoint set of states: $\nu_{(x,x')}(s,t)$ always returns a state, and each of these states will be different since it will be encoding those aspects of a pair $(s,t) \in p_f(x,x')$ that make it different from all other such pairs. The sum of the probabilities of a disjoint set of states is ≤ 1.0 by definition.

The two proofs that follow are both examples of how the flow network approach, with encoding, can be used to bound the mixing time of a Markov chain. The first is an original proof; the second is an extension of the work of Sinclair (1992). Both will prove useful later in the dissertation.

4.2.3 Bound on Gibbs Sampling for Bayesian Networks

We use the Bayesian network definition of Section 2.2.1. The nodes of the network are the variables $\mathbf{V} = V_1 \ldots V_n$, where each V_i takes values from $Val[V_i]$, and we represent V_i's parents as Pa_i, and its children as Ch_i. In addition, we assume that the variables are in reverse topological order, so that every child precedes its parents in the sequence.

In general, Bayes network inference involves conditioning on some set (possibly empty) of observed variables $\mathbf{O} = O_1 \ldots O_o$, with the assignment $\mathbf{o}$. The task is to estimate the posterior distribution over the other (hidden) variables, $\mathbf{H} = \mathbf{V} - \mathbf{O}$, or $\mathbf{P}(\mathbf{H}|\mathbf{o})$. We will accomplish this by using Gibbs sampling to draw samples from $\mathbf{H}$ according to a distribution that converges to

$$\pi(\mathbf{H}) = \mathbf{P}(\mathbf{H}|\mathbf{o}) = \frac{1}{\mathbf{P}(\mathbf{o})}\mathbf{P}(\mathbf{H}, \mathbf{o})$$

The chain will have $\prod_{H \in \mathbf{H}} |Val[H]|$ possible states. Since the values of all the variables, hidden or observed, will be known in each state, the states can then be expressed as complete assignments, e.g., $\mathbf{V} = v$, or $V_1 = v_1, \ldots, V_n = v_n$. And since we use Gibbs sampling, each state is connected to all the other states that differ from it in the value of a single variable—and, of course, to itself.

Before going on to describe a flow for this network, let us define a few useful quantities. First, let $|Ch[V_i]| \leq k$. Then, since we are assuming all variables to be discrete, we can upper-bound the number of values a variable can take, $|Val[V_i]|$ as $\leq d$. We can also define the minimum and maximum conditional probabilities of variables as follows

$$\hat{p} = \max_{i,v} P(v_i | v_{Pa_i})$$

$$\check{p} = \min_{i,v} P(v_i | v_{Pa_i})$$

We then define the **local variance bound** (Dagum & Luby, 1993) of the Bayes network as

$$\lambda = \hat{p}/\check{p}$$

λ can be used to lower-bound the state probability $\pi(\mathbf{H})$,

$$\pi_{min}(\mathbf{H}) = \min_{\mathbf{H}} \frac{P(\mathbf{H}, o)}{P(o)} \geq \frac{\check{p}^n}{\sum_{\mathbf{h}} P(\mathbf{h}, o)} \geq \frac{\check{p}^n}{\prod_{H \in \mathbf{H}} |Val[H]| \hat{p}^n} \geq \lambda^{-n} d^{-|\mathbf{H}|} \geq (d\lambda)^{-n} \quad (4.7)$$

as well as the transition probability $T(v'|v)$ (assuming a transition at variable V_l).

$$\begin{aligned} T_{min}(\mathbf{v}'|\mathbf{v}) &= \min_{\mathbf{v},l} \frac{P(v_l'|v_{Pa_l}) \prod_{V_i \in Ch[V_l]} P(V_i|v_{Pa_i})}{\sum_{v_l \in Dom[V_l]} P(v_l|v_{Pa_l}) \prod_{V_i \in Ch[V_l]} P(v_i|v_{Pa_i})} \\ &\geq \frac{\check{p}^{|Ch[V_l]|+1}}{|Dom[V_l]| \hat{p}^{|Ch[V_l]|+1}} \\ &\geq \lambda^{-k-1} \end{aligned} \quad (4.8)$$

We now define the flow f. Given an origin state s and a destination state t, we will move from s to t by changing the values of all the differing variables one by one, starting at V_1. E.g., whenever s_1 and t_1 differ, the first step will take us from $s_1 s_2 \ldots s_n$ to $t_1 s_2 \ldots s_n$, as shown in Figure 4.1.

Note that the length $l(f)$ of this flow is clearly $\leq n$. Let us now bound the cost $\rho(f)$.

Theorem 4.2.2: Consider the chain transition $x \to x'$, which is responsible for changing the lth variable, so that $x = t_1 \ldots t_{l-1} s_l \ldots s_n$. Consider also the encoding

$$\nu_l(s,t) = s_1 \ldots s_{l-1} t_l \ldots t_n \ .$$

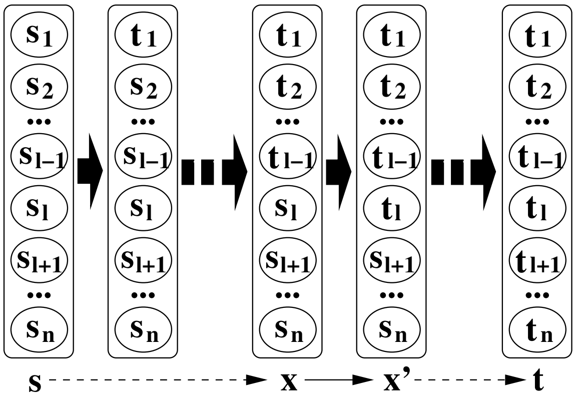


Figure 4.1. The flow defined for Bayesian network Gibbs sampling.

It is then the case that

$$\pi(s)\pi(t) \quad \leq \quad \gamma\pi(\nu_l(s,t))\pi(x)$$

where

$$\gamma = \max_{(s,t,l)} \prod_{i=1}^{l-1} \Big(\frac{P(s_i|s_{i+1\ldots n})}{P(s_i|s_{i+1\ldots l-1}t_{l\ldots n})} \frac{P(t_i|t_{i+1\ldots n})}{P(t_i|t_{i+1\ldots l-1}s_{l\ldots n})} \Big) \tag{4.9}$$

Proof:

$$\begin{aligned}
& \pi(s)\pi(t) \\
= \quad & \frac{1}{P(\mathbf{o})^2} P(s)P(t) \\
= \quad & \frac{1}{P(\mathbf{o})^2} \prod_{i=1}^{l-1} \Big(P(s_i|s_{i+1\ldots n})P(t_i|t_{i+1\ldots n}) \Big) \prod_{i=l}^{n} \Big(P(s_i|s_{i+1\ldots n})P(t_i|t_{i+1\ldots n}) \Big) \\
\leq \quad & \frac{1}{P(\mathbf{o})^2} \gamma \prod_{i=1}^{l-1} \Big(P(t_i|t_{i+1\ldots l-1}s_{l\ldots n})P(s_i|s_{i+1\ldots l-1}t_{l\ldots n}) \Big) \prod_{i=t}^{n} \Big(P(s_i|s_{i+1\ldots n})P(t_i|t_{i+1\ldots n}) \Big) \\
= \quad & \frac{1}{P(\mathbf{o})^2} \gamma P(x)P(\nu_l(s,t)) \\
= \quad & \gamma\pi(x)\pi(\nu_l(s,t))
\end{aligned}$$

■

Now all that remains is to derive a simple bound for γ. Since Equation 4.9 treats t and s in exactly the same way, and we're maximizing over both, we can rewrite that equation as

$$\gamma = \max_{(s,t,l)} \prod_{i=1}^{l-1} \Big(\frac{P(s_i|s_{i+1\ldots n})}{P(s_i|s_{i+1\ldots l-1}t_{l\ldots n})}\Big)^2$$

Note that, for any variable V_i which is independent of the variables $V_{l\ldots n}$ given the variables $V_{i+1\ldots l-1}$, the numerator and denominator will cancel, and for the variables where they do not, the ratio will be $\leq \lambda$. Therefore

$$\gamma \leq \lambda^{2r} \tag{4.10}$$

where r denotes the number of variables which occur before V_l, but have at least one parent that occurs after it. This quantity will be minimized when the nodes are ordered according to the Minimum Destination Vertex Set Linear Arrangement (see Appendix A). Using this bound on γ and Equation 4.6, we can now constrain the flow cost as

$$\rho(f) \leq \frac{\gamma}{T_{min}(x'|x)} = \lambda^{2r+k+1} \, .$$

Finally, we can rewrite Equation 4.4 using our bounds on $l(f)$ and $\rho(f)$, and the bound on $\pi_{min}(\mathbf{H})$ of Equation 4.7), to get

$$\begin{aligned} \tau(\epsilon) &\leq \alpha\rho(f)l(f)(\ln \pi_{min}(x)^{-1} + \ln \epsilon^{-1}) \\ &\leq \alpha\lambda^{2r+k+1}n(n\ln(\lambda d) + \ln \epsilon^{-1}) \end{aligned} \tag{4.11}$$

where, as a reminder, α is some constant, n is the number of variables, λ is the local variance bound of the network, k is the maximum number of children a node may have, d is the maximum number of values a node may take, and r is the score of the Minimum Destination Vertex Set Linear Arrangement: the maximum number of nodes pointed to by the edges that all cross the same gap between the ordered nodes. This number is clearly $\geq k$, because all the edges pointing out of a node will cross the gap out of that node, and they certainly all point to different nodes. Thus, λ^{2r+k+1} may be rewritten as λ^{3r+1}.

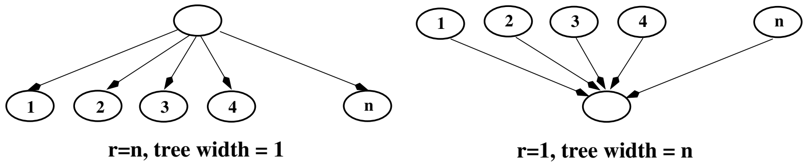


Figure 4.2. Two extreme types of Bayesian network, one with n children, and one with n parents.

There do exist many networks where r can be bounded by a constant independent of n. Hidden Markov models, as illustrated in Appendix A, have an r of two. (Equivalent to k, the number of children.) In general, the r of a dynamic Bayesian network will be, at most, equal to the number of nodes in each time-slice. It should be noted that r has little to do with *tree width*, the graph-theoretic quantity that constrains the running time of exact Bayesian network inference: tree width is highly dependent on the maximum number of parents a node may have, and often independent of the number of children. Figure 4.2 contrasts the two quantities for two extreme types of Bayesian network. Unfortunately, although r and tree width are unrelated, the bound of Equation 4.11 can still depend on the number of parents. The local variance bound λ is the ratio of maximum and minimum CPF values, and there are many conditional probability distributions where the value of a conditional probability function is dependent on the number of parents: noisy-or, for example. If this number of parents is proportional to n, λ will be exponential in n and the approximate MCMC algorithm will be intractable. Of course, this is hardly surprising, since Dagum and Luby (1993) have shown that approximate Bayesian network inference is **NP**-hard for all Bayesian networks where $\lambda > 2^{cn}$ for some constant c. For this to occur, a network must either have some zero CPF values, or some CPF values that are exponential in n themselves (meaning that some nodes in the network must have $O(n)$ parents). In general, using MCMC makes less and less sense as the conditional probability distributions become more and more deterministic.

4.2.4 Bound on M-H for Weighted Monomer-Dimer Systems

A **monomer-dimer system** is an undirected graph $G = (\mathbf{V}, \mathbf{E})$, where the vertices can be covered by non-overlapping arrangements of **monomers** (molecules occupying a single vertex) and **dimers** (molecules occupying two neighbouring vertices.) An arrangement can be expressed as $(\mathbf{M}, \mathbf{D})$, where $\mathbf{M} \subseteq \mathbf{V}$ is the set of monomers, and $\mathbf{D} \subseteq \mathbf{E}$ is the set of dimers. Each arrangement is then given a weight $\lambda^{|\mathbf{D}|}$, where λ is a positive real parameter representing the energy contribution of a dimer. The weights of all the possible arrangements can be normalized to yield a probability distribution over the space of arrangements. A Monte Carlo Markov chain over this state space is described, and analyzed using flow techniques, by Jerrum and Sinclair (1997).

The monomer-dimer problem is a more general version of **bipartite matching**. In a bipartite graph, the vertices are divided into two sets, $\mathbf{V_1}$ and $\mathbf{V_2}$, and each edge has to span both sets. A matching for this graph is a set of edges $\mathbf{D} \subseteq \mathbf{E}$ such that no two edges share an end-point: essentially, a set of dimers. A **weighted** bipartite graph is one where each edge has an associated weight $w_E(E)$, and the weight of any matching on such a graph is the product of the weights of $\mathbf{M}$. For our purposes, we want to expand the definition so that *unmatched* vertices $\mathbf{M} \subseteq \mathbf{V}$ also have weights, written $w_V(M)$. The weight of a matching then becomes

$$w(\mathbf{M}, \mathbf{D}) \quad = \quad \prod_{M \in \mathbf{M}} w_V(M) \prod_{D \in \mathbf{D}} w_E(D)\,. \tag{4.12}$$

Again, we can normalize this to obtain a probability distribution, $\pi(\mathbf{M}, \mathbf{D})$. Since there are at most $|V|!$ possible matchings, this distribution can be bounded below by

$$\pi_{min}(\mathbf{M}, \mathbf{D}) \geq (|V|!)^{-1} \check{w}_V^{|V|} \check{w}_E^{|E|} \tag{4.13}$$

where $\check{w}_E = \min_{(E \in \mathbf{E})} w_E(E)$ and $\check{w}_V = \min_{(V \in \mathbf{V})} w_V(V)$. (We define $\hat{w}_E$ and $\hat{w}_V$ analogously, to represent the corresponding maxima.)

We have extended the proof of Jerrum and Sinclair (1997) to handle weighted monomer-dimer systems. We begin by adapting their Markov chain, defining the transition function $T(x' = (\mathbf{M}', \mathbf{D}')|x = (\mathbf{M}, \mathbf{D}))$ as follows:

```
Select an edge (u,v) ∈ E u.a.r.
if (u,v) ∈ D then
  D' = D − {(u,v)}, M' = M ∪ {u,v}
else if u,v ∈ M then
  D' = D ∪ {(u,v)}, M' = M − {u,v}
else if u ∈ M, (w,v) ∈ D then
  D' = D ∪ {(u,v)} − {(w,v)}, M' = M ∪ {w} − {u}
else if v ∈ M, (u,w) ∈ D then
  D' = D ∪ {(u,v)} − {(u,w)}, M' = M ∪ {w} − {v}
else
  D' = D, M' = M
end if
Accept the transition with probability min(1, w(M',D')/w(M,D))
```

At each time step, a single edge in $\mathbf{E}$ is selected uniformly at random. If this edge is part of the current matching, it is removed; if both of its endpoints are free, it is added. If only one of its end-points is free, the other one is freed by removing the edge it currently belongs to, and the chosen edge is added instead. If neither endpoint is free, nothing happens. In each case, both $\mathbf{D}$ and $\mathbf{M}$ are updated appropriately.

Depending on the edge selected, each transition can therefore involve either an edge addition, an edge deletion, or an edge "replacement", where one edge is replaced by another, adjacent, edge. The transition function is

$$T(x|x') = |\mathbf{E}|^{-1} \min\left(1, \frac{\pi(x')}{\pi(x)}\right) .$$

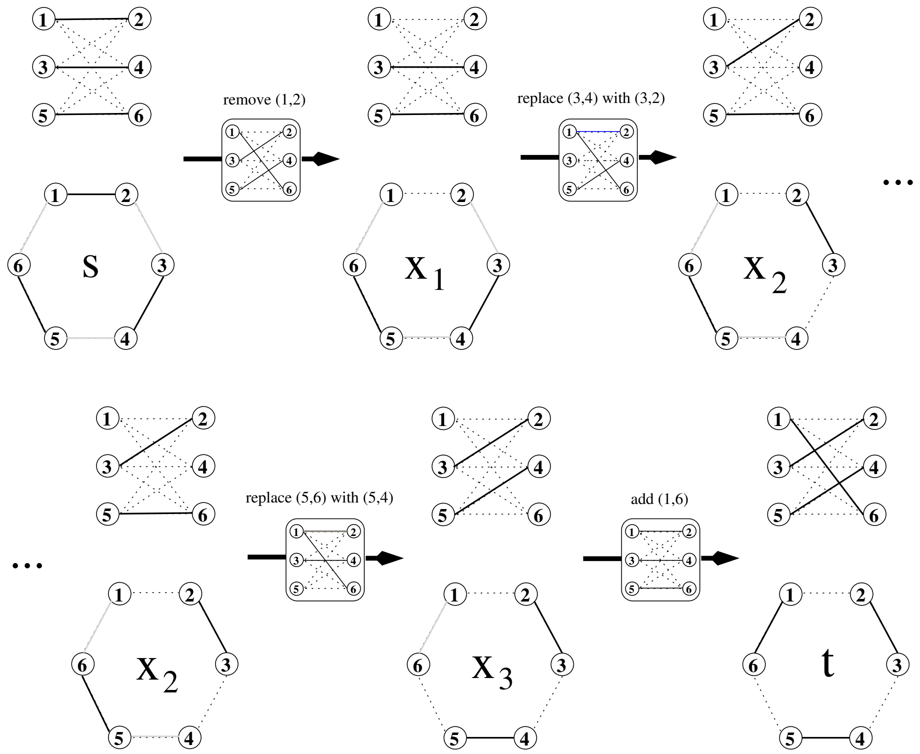


Figure 4.3. The canonical path from s to t. States are represented both as matchings, and as compounds; transitions display their encodings. Black edges always represent the current state. In the compounds, grey edges show the edges of t, and in the encodings, the grey edge is the removed "starter edge".

We now describe a flow f. We begin by observing that, for any pair of states $s = (\mathbf{M}_s, \mathbf{D}_s)$ and $t = (\mathbf{M}_t, \mathbf{D}_t)$, the connected components of the symmetric difference $\mathbf{D}_s \triangle \mathbf{D}_t$ are paths or cycles, each of which has edges that belong alternately to $\mathbf{D}_s$ and to $\mathbf{D}_t$. We assume that we have some way of ordering these components, and of picking a start vertex for each one (this must be an end vertex for a path, but is arbitrary for a cycle.) The canonical path from s to t then takes each component in turn, and "unwinds" it. For a path starting with an edge in t, this boils down to replacing the second edge (which must be an edge from s) with the first, then replacing the fourth edge with the third, and so on until the end, where a final edge might need to be simply added. If the path begins with an edge in s, then this edge will have to be deleted before the whole process can start. Cycles are processed by, first, removing the s edge adjacent to the start vertex, then, unwinding as before, and, finally, adding in a final edge. Figure 4.3 illustrates how this works for a small bipartite matching problem, where the only component is a cycle of six edges.

It should be clear that the maximum length of a canonical path in the flow is $l(f) \leq |\mathbf{V}|$. To bound the cost of the flow, we will adapt the encoding of Jerrum and Sinclair (1997). For a starting state s, a destination state t, and the transition $x_l \rightarrow x_{l+1}$, where $x_l = (\mathbf{M}, \mathbf{D})$, $x_{l+1} = (\mathbf{M}', \mathbf{D}')$, this encoding contains all the edges s and t have in common, as well as all the edges of s that are no longer in x_l, and all the edges of t that are not yet in x_l. This can be expressed as $D_s \bigtriangleup D_t \bigtriangleup (D \cup D')$. If the transition is one of the replacement transitions of a cycle, the encoding is changed slightly, in that the first edge deleted from the cycle ($e_1 = (u_1, v_1)$) is removed from the encoding as well: this is done because e_1 is certain to share a vertex with another edge in the encoding, and we want the encoding to return a valid state. The transitions in Figure 4.3 are accompanied by illustrations of the relevant encodings.

Since we are just as interested in vertices as in edges, let us now consider the set of vertices that make it into each encoding. For the edge addition and deletion transitions, this set will consist of $M_s \bigtriangleup M_t \bigtriangleup (M \cap M')$. When the transition is a replacement transition, the encoding will also contain the "pivot" vertex, the vertex shared by the edge which was removed and the edge that was added. Finally, any encoding where e_1 has been deleted will obviously contain its free vertex.

As in the time-complexity proof of Jerrum and Sinclair (1997), let us now review four different types of transitions, and, in each case, consider how the γ of

$$\pi(s)\pi(t) \leq \gamma\pi(x_l)\pi(\nu_l(s,t))$$

may be bounded, and then combined with $T(x_{l+1}|x_l)$ to bound $\rho(f)$ as in Equation 4.6.

- **Edge deletions**. Here, $\nu_l(s,t) = (D_s \triangle D_t \triangle D, M_s \triangle M_t \triangle M)$. Therefore, when combined into multi-sets, the edges and vertices of $x_l \cup \nu_l(s,t)$ will match exactly the combined edges and vertices of $s \cup t$. Since, according to Equation 4.12, the probability of each state is a normalized product of the weights of all the edges and vertices, this means that

$$\pi(s)\pi(t) = \pi(x_l)\pi(\nu_l(s,t))$$

and so $\gamma = 1.0$, and $\rho(f) \leq T(x_{l+1}|x_l)^{-1} \leq |\mathbf{E}| \max_{x_l,x_{l+1}} \pi(x_l)/\pi(x_{l+1})$ Meanwhile, x_{l+1} contains one edge less, and two vertices more, than x_l, meaning that $\pi(x_l)/\pi(x_{l+1}) \leq \hat{w}_E/\check{w}_V^2$, and, therefore

$$\rho(f) \leq |\mathbf{E}|\frac{\hat{w}_E}{\check{w}_V^2}$$

- **Edge additions**. Here, $\nu_l(s,t) = (D_s \triangle D_t \triangle D', M_s \triangle M_t \triangle M')$. Therefore, we can argue as above to show that

$$\pi(s)\pi(t) = \pi(x_{l+1})\pi(\nu_l(s,t))$$

and so $\gamma = \pi(x_{l+1})/\pi(x_l)$, and $\rho(f) \leq |\mathbf{E}| \max_{x_l,x_{l+1}} \pi(x_{l+1})/\pi(x_l)$ Meanwhile, the relative contents of x_{l+1} and x_l are reversed, meaning that, here, $\pi(x_{l+1})/\pi(x_l) \leq \hat{w}_E/\check{w}_V^2$, and, again

$$\rho(f) \leq |\mathbf{E}|\frac{\hat{w}_E}{\check{w}_V^2}$$

- **Edge replacements along a path**. Consider the deleted edge $e = (u,v)$, and the added edge $e' = (u',v)$. Then, $\nu_l(s,t) = (D_s \triangle D_t \triangle (D+e'), M_s \triangle M_t \triangle (M - x') + v)$, and the multiset of the edges of $s \cup t$ contains one more edge, e', and two fewer vertices, u' and v, than $x_l \cup \nu_l(s,t)$. This means that

$$\pi(s)\pi(t) \leq \frac{\hat{w}_E}{\check{w}_V^2}\pi(x_l)\pi(\nu_l(s,t))$$

and so $\gamma = \hat{w}_E / \check{w}_V^2$. Meanwhile, $\pi(x_l)/\pi(x_{l+1}) \leq (\hat{w}_E \hat{w}_V)/(\check{w}_E \check{w}_V)$, and so

$$\rho(f) \leq |\mathbf{E}| \frac{\hat{w}_E^2 \hat{w}_V}{\check{w}_E \check{w}_V^3}$$

- **Edge replacements within a cycle**. This is set up like the previous case, but here $\nu_l(s,t) = (D_s \triangle D_t \triangle (D + e') - e_1, M_s \triangle M_t \triangle (M - u') + v + v_1 + u_1)$, and so

$$\pi(s)\pi(t) \leq \frac{\hat{w}_E^2}{\check{w}_V^4} \pi(x_l)\pi(\nu_l(s,t))$$

  Meaning that $\gamma = \hat{w}_E^2 / \check{w}_V^4$. Meanwhile, the ratio $\pi(x_l)/\pi(x_{l+1})$ is the same as before, leading to

$$\rho(f) \leq |\mathbf{E}| \frac{\hat{w}_E^3 \hat{w}_V}{\check{w}_E \check{w}_V^5}$$

The last type of transition clearly dominates here. We can now substitute our bounds on $\rho(f)$, $l(f)$, and π_{min} into Equation 4.4, to get

$$\begin{aligned} \tau(\epsilon) &\leq \alpha \rho(f) l(f) (\ln \pi_{min}(x)^{-1} + \ln \epsilon^{-1}) \\ &\leq \alpha |V||E| \frac{\hat{w}_E^3 \hat{w}_V}{\check{w}_E \check{w}_V^5} (|V| \ln |V| - |V| \ln \check{w}_V - |E| \ln \check{w}_E + \ln \epsilon^{-1}) \end{aligned} \tag{4.14}$$

The most significant element here is clearly the ratio of maximum and minimum weights. If either the edge or the vertex weights are exponential in $|V|$, then so is the mixing time. In general, the wider the disparity between the maximum and minimum weights, the slower the chain will mix; and if either type of weight has a minimum value of zero, we cannot guarantee that the chain will mix at all. Unfortunately, the problem of very low, or zero, weights of either type is not unlikely to occur in practical applications where the weights measure the probability that two vertices might be matched. Of course, if it is only the edge weights that can reach zero, this problem can be partially overcome by removing the zero-weight edges from $\mathbf{E}$. It might even make sense to set the very low weights of some of the very unlikely matches to zero, and then remove them.

Chapter 5

MCMC for Identity Uncertainty

In a representation based on possible worlds, the probability of a query is proportional to the sum of the probabilities of all the worlds in which both the query and the evidence are true. When there are many such worlds, performing exact probabilistic inference by summing over all of them may be impractical. This problem becomes particularly urgent when we attempt to reason about identity uncertainty since, in order to do so, we augment the representation of possible worlds with variables encoding term-to-object mappings or clusterings. The state space of these variables can grow rather large: at worst, the possible clusterings will correspond exactly to the possible partitions (see Appendix B) of the set of terms. The growth of the number of partitions as the number of terms increases is described by the Bell numbers, whose asymptotic behaviour is more than exponential. Thus, in the general case, summing over all the clusterings quickly becomes intractable as we add objects to the knowledge base. It is to overcome this problem that we have designed a family of approximate, MCMC-based algorithms. In this chapter, we outline several such algorithms, appropriate in various identity-uncertain settings. We also describe some conditions under which the algorithms can be proved to approximate the correct distribution. Throughout the chapter, we illustrate our explanations with examples based on the ball and urn domain of Example 3.1.

5.1 The General Approach

The state space of our Markov chain is the space of all the compact possible worlds. We use this representation rather than simple possible worlds in order to minimize the state space: in fact, this is one of the main reasons why we the representation was first introduced. As a reminder, a **compact possible world**, as defined in Section 3.3.2, is a triple $\omega = \langle n, \iota, \mathbf{V} \rangle$, where n is the number of objects in the world, ι describes the identity mappings of terms using a set of equivalence classes, and $\mathbf{V}$ is a set of arrays associated with various attributes, each one assigning values to all the possible applications of the attributes.

The stationary distribution of the chain is the probability distribution over all the compact possible worlds given the evidence: that is, the probability of all the possible worlds that disagree with the evidence is set to 0.0, and the probability of the remaining worlds is normalized accordingly. We will refer to this distribution as $\pi(\omega) = \alpha P(n, \iota, \mathbf{V})$.

Example 5.1: As an example, consider the ball and urn domain of Figure 3.2, where three balls (B_1, B_2, and B_3) have been sampled with replacement, and the evidence consists of the facts *Colour*(B_1) = *White*, *Colour*(B_2) = *Black*, and *Colour*(B_3) = *White*. Two possible worlds for this domain, as illustrated in Figure 5.1, might include

$$\langle n = 5, \ , \iota = \{\{B_1, B_3\}\{B_2\}\}, \ \mathbf{V} = \{V_{\mathit{Colour}} = [\mathit{White}, \mathit{Black}, \mathit{White}, \mathit{White}, \mathit{Black}]\}\rangle$$

and

$$\langle n = 3, \ , \iota = \{\{B_1\}\{B_2\}\{B_3\}\}, \ \mathbf{V} = \{V_{\mathit{Colour}} = [\mathit{White}, \mathit{Black}, \mathit{White}]\}\rangle \, .$$

Note that these two words show the only two values of ι possible given the observations.

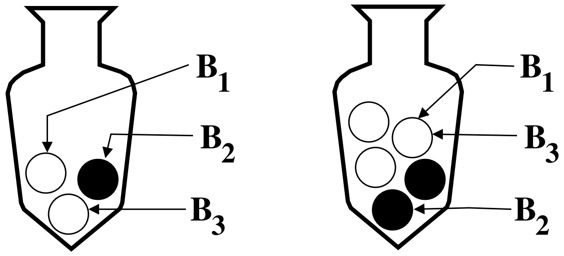


Figure 5.1. Two of the possible worlds for the domain introduced in Figure 3.2. They are described in Example 5.1.

Our general algorithm is a mixture of Metropolis-Hastings transition kernels, each one dealing with only a few aspects of the state at a time, so that many of the probabilities cancel in the acceptance ratios. More formally, we have a set of proposal distributions, and, at each step, we randomly select one of these distributions, use it to propose a new state, and then accept or reject this state using the usual acceptance ratio. In this chapter, we focus on distributions capable of suggesting transitions in n and ι. For the time being, we assume that we have a mechanism for proposing changes to all unobserved variables in $\mathbf{V}$ for the current ω, and that, given some assumptions, this mechanism connects every pair of possible $\mathbf{V}$ assignments with some path of non-zero probability. In the ball and urn domain, we do not need such a mechanism: the attributes of all the observed balls are known, and the attributes of the unobserved balls are, as we will show, irrelevant. In later chapters, where we consider more complex domains, we will define mechanisms for proposing changes to $\mathbf{V}$ as needed.

5.2 Transitions in n

A method for suggesting transitions in n can be defined using a simple random walk on the line fragment $[|\iota| \ldots n_{MAX}]$. (The fragment begins at $|\iota|$ because the number of objects in the world clearly cannot be less than the number of objects referred to by terms: for example, if we know that B_1 and B_2 co-refer but B_3 is separate and unique, we know that there are at least two objects in the world.) Note, however, that a transition in n may affect $\mathbf{V}$ as well, since, when n is decreased to $n-k$, we will have to remove the variables referring to the last k objects, and when n is increased to $n+k$, we will have to introduce new variables to represent the new objects' attributes. When we introduce new variables, we will need to propose values for any that are unobserved. Thus, we can write the general proposal distribution as a product of two distributions, one dealing with n, and one dealing with the set of relevant unobserved attributes $\mathbf{U}$:

$$q_n(n'|n)q_{\mathbf{U}}(\mathbf{V}'|n', \iota, \mathbf{V}) \, .$$

The corresponding acceptance probability is

$$\min\left(1.0, \frac{q_n(n|n')q_{\mathbf{U}}(\mathbf{V}|n, \iota, \mathbf{V}')P(n', \iota, \mathbf{V}')}{q_n(n'|n)q_{\mathbf{U}}(\mathbf{V}'|n', \iota, \mathbf{V})P(n, \iota, \mathbf{V})}\right) \, .$$

One possible simple $q_{\mathbf{U}}(\mathbf{V}'|n', \iota, \mathbf{V})$ is one that draws a value for each $U \in \mathbf{U}$ u.a.r. from $Val[U]$. Unfortunately, this simple proposal may suggest very unlikely value combinations, resulting in low acceptance ratios. Depending on how the distribution over $\mathbf{V}$ is defined, we may be able to design more intelligent proposals.

5.2.1 Summing out Unobserved Objects

It is worth noting that, in domains where the identity mappings of the constants do not depend on any object attributes, we can sum out and ignore the attributes of the unobserved objects. To explain this, let us first recall that, by the first part of Equation 3.1,

$$P(n, \delta, \mathbf{V}) = P(n)P(\delta|n)P(\mathbf{V}|\delta, n)\,.$$

Also recall that, by Theorem 3.4.1,

$$P(n, \iota, \mathbf{V}) = \frac{n!}{(n-|\iota|)!}P(n, \delta', \mathbf{V})$$

where δ' is any one of the identity mappings corresponding to the identity clustering ι.

Note that we can easily split every value mapping array in $\mathbf{V}$ into two parts: one describing the mappings of the observed objects, and one describing the mappings of the unobserved objects. Let us place the former arrays in the set $\mathbf{V_O}$, and the latter arrays in the set $\mathbf{V_U}$. Since there is no way of referring to the attributes of unobserved objects when defining distributions over attribute values, the attributes of the observed objects must be independent of those of the unobserved objects, and we can state that

$$P(\mathbf{V}|\delta, n) = P(\mathbf{V_U}|\delta, n)P(\mathbf{V_O}|\delta, n)\,.$$

We can combine all three equations to get

$$P(n, \iota, \mathbf{V}) \;=\; \frac{n!}{(n-|\iota|)!}P(n)P(\delta'|n)P(\mathbf{V_U}|\delta', n)P(\mathbf{V_O}|\delta', n)\,.$$

Now, we can sum over all the possible values of $\mathbf{V_U}$ as follows:

$$\begin{aligned}\sum_{\mathbf{V_U}} P(n, \iota, \mathbf{V}) \;&=\; \frac{n!}{(n-|\iota|)!}P(n)P(\delta'|n)P(\mathbf{V_O}|\delta', n)\sum_{\mathbf{V_U}} P(\mathbf{V_U}|\delta', n)\\ &=\; \frac{n!}{(n-|\iota|)!}P(n)P(\delta'|n)P(\mathbf{V_O}|\delta', n)\end{aligned}$$

Moreover, in domains where attribute values do not depend on the number of objects in the world, so that $P(\mathbf{V_O}|\delta, n) = P(\mathbf{V_O}|\delta)$, we can sum out and ignore even the *number* of unobserved objects. Recall that, when the identity mappings of the constants do not depend on any object attributes, $P(\delta|n)$ is just a function of n and the number of observed objects $m = |\iota|$: let us call this function $f(n, |\iota|)$. Now, when we sum over all possible values of n, we get

$$\sum_n P(n, \iota, \mathbf{V}) \;=\; P(\mathbf{V_O}|\delta') \sum_n \frac{n!}{(n - |\iota|)!} P(n) f(n, |\iota|)\,.$$

The whole $\sum_n \frac{n!}{(n-|\iota|)!} f(n, |\iota|) P(n)$ component is a function of $|\iota|$, one that we can cache or pre-calculate.

In the domain of Example 5.1, balls are sampled u.a.r. with replacement, and so the identity mappings of the constants do not depend on their attributes, and $f(n, |\iota|) = n^{-|\iota|}$ by Equation 3.3. When $P(n)$ is non-zero only for a limited set of n values, pre-calculating the summation is quite simple. Since the attribute values are all known, this means that we can use an algorithm that moves only through the space of ιs, query permitting.

5.3 Transitions in ι

In this section, we will describe several methods for proposing transitions in ι. Each one is useful under slightly different conditions. In every case, we will explain what those conditions are, and prove that an algorithm based on the relevant method approximates the correct probability distribution. However, before going on to list the individual methods, we will discuss some issues that apply to all of them.

In general, a change to ι may affect $\mathbf{V}$ just as a change to n does: when proposing a new ι', we may need to propose changes to some of the $\mathbf{V}$s. The reason for this is that a new ι' can propose (among other things) that two currently separate terms be clustered together into a single object. If these two terms currently have different values for some attribute, and if we leave these values alone, the probability of the destination state will be zero. When this attribute is an observed one, then the two terms should not co-refer, and everything is as it should be. However, when the attribute is unobserved, this approach can lead to trouble. Identity transitions will have a chance of occurring only when the instances "line up"—that is, a sequence of normal transitions causes all of their attribute values to match exactly. When instances have many attributes, line-up may be very rare, resulting in a low acceptance ratio and a slow algorithm. Moreover, there are cases in which waiting for line-up makes the Markov chain non-ergodic—the algorithm can be trapped in a subset of the possible ιs with no escape.

As an example, suppose that the *Colour* attribute is sometimes unobserved: so that, while we know that ball B_1 is *White* and ball B_2 is *Black*, we know nothing about ball B_3. Obviously, B_3 should be free to co-refer with any of the other two balls. However, if the current ι is $\{\{B_1, B_3\}, \{B_2\}\}$, then B_3's *Colour* is set to *White*, and so a move to $\{\{B_1\}, \{B_2, B_3\}\}$ will have a probability of zero.

We will avoid this problem by, again, using a second proposal distribution to suggest new values for all the relevant unobserved attributes $\mathbf{U}$. The general proposal distribution can then be written as

$$q_\iota(\iota'|n, \iota, \mathbf{V})q_{\mathbf{U}}(\mathbf{V}'|n, \iota', \mathbf{V})$$

and the corresponding acceptance probability is

$$\min\left(1.0, \frac{q_\iota(\iota|\iota', \mathbf{V}', n)q_{\mathbf{U}}(\mathbf{V}|\iota, \mathbf{V}', n)P(n, \iota', \mathbf{V}')}{q_\iota(\iota'|\iota, \mathbf{V}, n)q_{\mathbf{U}}(\mathbf{V}'|\iota', \mathbf{V}, n)P(n, \iota, \mathbf{V})}\right). \tag{5.1}$$

The proposal distribution $q_{\mathbf{U}}(\mathbf{V}'|n, \iota', \mathbf{V})$ can be defined as in Section 5.2. Now, before outlining several ways of defining $q_\iota(\iota'|n, \iota, \mathbf{V})$, we would like to make some general formal claims about MCMC as applied to this setting. We begin by defining a **consistent state** ω as a state which is consistent with both the knowledge base and the evidence, and has a positive probability according to the probability models. We will now go on to prove a couple of helpful lemmas.

Lemma 5.3.1: Consider an old state $\omega = \langle n, \iota, \mathbf{V}\rangle$ and a new state $\omega' = \langle n, \iota', \mathbf{V}'\rangle$, proposed using some $q_\iota(\iota'|n, \iota, \mathbf{V})$ and some $q_{\mathbf{U}}(\mathbf{U}'|n, \iota', \mathbf{V})$. The new state ω' will have a non-zero acceptance probability, as long as the following conditions hold:

1. The states ω and ω' are both consistent.

2. For each attribute $V_i \in \mathbf{V}$ with some parents Pa_i (which may include ι), and for every possible combination of Pa_i values, the distribution $P(V_i|Pa_i)$ contains no zeroes.

3. The proposal distribution $q_\iota()$ used to suggest new ιs is reversible in the sense that, if $q_\iota(\iota'|n, \iota, \mathbf{V}) > 0$, then so is $q_{\iota'}(\iota|n, \iota', \mathbf{V}')$, for all possible values of $\mathbf{V}$ and $\mathbf{V}'$. [1]

4. The proposal distribution $q_{\mathbf{U}}()$ used to suggest new $V_i \in \mathbf{U}$ values has some chance of suggesting every $v \in Val[V_i]$.

Proof: The acceptance probability will be non-zero if the ratio in Equation 5.1 is greater than 0.0. All the q terms are guaranteed to be non-zero: the $q_\iota()$ as a consequence of condition 3, and the $q_{\mathbf{U}}()$ as a consequence of condition 4. Let us now consider the $P()$ terms. Since the only aspects of the world state that change are ι and $\mathbf{U}$, much of the $P()$ ratio will cancel. Let $\mathbf{W}$ contain all the children of the variables of $\mathbf{U}$ that are not in $\mathbf{U}$ themselves, so that

$$\mathbf{W} = \{W : W \in \mathbf{V} - \mathbf{U}, \exists V_i \in \mathbf{U} \; s.t. \; W \in Ch_i\} \; .$$

Also, let $\mathbf{X}$ represent all those aspects of $\mathbf{V}$ on which identity mappings depend. Some of them might be in $\mathbf{V}$ or $\mathbf{W}$ as well, but this does not matter: we know that the distribution over possible worlds can be factored according to the chain rule. The $P()$ ratio can then be written as

$$\frac{P(n, \iota', \mathbf{V}')}{P(n, \iota, \mathbf{V})} = \frac{(n - |\iota|)!}{(n - |\iota'|)!} \frac{P(\delta' \in \Delta_{\iota'}|\mathbf{X}', n)}{P(\delta \in \Delta_\iota|\mathbf{X}, n)} \prod_{V_i \in \mathbf{U}} \frac{P(V_i'|Pa_i')}{P(V_i|Pa_i)} \prod_{V_j \in \mathbf{W}} \frac{P(V_j'|Pa_j')}{P(V_j|Pa_j)} \; .$$

[1] Although, in this chapter, the $\mathbf{V}$ values will not matter: the proposals we use pick ι' based only on ι.

The first term of this product is clearly non-zero. The second is non-zero given condition 1. Finally, the products over $\mathbf{U}$ and $\mathbf{W}$ will be zero only if one of the conditional probabilities in one of the numerators is zero. This is impossible for the variables in $\mathbf{U}$, since, according to condition 1, ω' must have a non-zero probability; and it is impossible for the children of variables in $\mathbf{U}$, which obviously must have at least one parent, because of condition 2. If none of the terms can equal zero, then neither can their product, and the new state must have a non-zero acceptance probability. ■

We will make use of Lemma 5.3.1 later. Now, suppose that we are given mechanisms m_n and $m_{\mathbf{V}}$ for proposing new values of n and $\mathbf{V}$, respectively. Let us define $\mathcal{M}$ as a Markov chain which combines these mechanisms with a mechanism m_ι for proposing new ιs by, at each step, choosing from among all these mechanisms uniformly at random. The selected mechanism is used to propose a new world state, which is accepted or rejected according to the usual M-H ratio.

Lemma 5.3.2: Every pair of consistent states in $\mathcal{M}$ is connected by a path of non-zero probability, as long as the following conditions hold:

1. The mechanism m_n can be used to construct a positive-probability path connecting every possible pair of values of n.
2. The mechanism $m_{\mathbf{V}}$ can be used to construct a positive-probability path connecting every possible pair of values of $\mathbf{V}$.
3. The mechanism m_ι can be used to construct a positive-probability path connecting every two possible values of ι.

In this chapter, we assume that the m_n and $m_{\mathbf{V}}$ we are given have the required property, and, as we describe various m_ιs, we prove that the property holds for them as well.

Proof: Consider a starting state $s = \langle n_s, \iota_s, \mathbf{V}_s \rangle$ and a final state $t = \langle n_t, \iota_t, \mathbf{V}_t \rangle$. Assume that both states are consistent. The states s and t can be connected by carrying out the following sequence of steps:

- Make sure that n is at least $|\mathbf{T}|$. This is accomplished using the proposal mechanism for n. If condition 1 holds, this part of the path will have a non-zero probability.
- Move from ι_s to ι_t using the positive-probability path guaranteed to exist by condition 3.
- Change n to n_t; again, if condition 1 holds , this part of the path will have a non-zero probability.
- Make sure all predicates are at the values of $\mathbf{V}_t$, changing them as necessary. This can be accomplished using the proposal mechanism for $\mathbf{V}$. If condition 2 holds, this final part of the path will definitely have a non-zero probability.

As long as each part of this sequence has a positive probability, so does the whole sequence, and, under our assumptions, this will be true for every part of the sequence. Therefore every pair of states in $\mathcal{M}$ is connected by a path of non-zero probability. ■

We can use Lemma 5.3.2 to demonstrate that the chain $\mathcal{M}$ can be used to perform approximate inference.

Theorem 5.3.3: Consider the chain $\mathcal{M}$, which is a randomized mixture of Metropolis-Hastings transition kernels which cover the state space fully, and which satisfies Lemma 5.3.2. Consider also a query sentence S. We can then approximate the probability that S is true by running $\mathcal{M}$ and keeping track of how often it is true in the possible worlds visited by the chain.

Proof: We can prove that the chain is ergodic with the right stationary distribution $\pi(\omega)$ by proving the three conditions set out in Section 4.1. This can be done as follows:

- According to Lemma 5.3.2, every pair of states is connected by a path of non-zero probability. Therefore, $\mathcal{M}$ is irreducible.
- According to Lemma 4.1.2, the detailed balance condition holds for chains which are randomized mixtures of Metropolis-Hastings transition kernels which cover the state space fully. Therefore, the detailed balance condition holds for $\mathcal{M}$.
- Since there is, at each step of the chain, some probability that the proposed state will be identical to the current one, every state has a non-zero self-loop probability, and so $\mathcal{M}$ is aperiodic.

Since all three conditions hold, the chain $\mathcal{M}$ is ergodic with stationary distribution $\pi(\omega)$. By Theorem 4.2, samples drawn from such a chain will approximate expectations over this distribution. This includes the expected truth-value of any query sentence. ■

We have just proved that the chain $\mathcal{M}$ can be used to perform approximate inference about the truth of the sentence S. Note, however, that we have said nothing about how long this might take. In some cases, we may be able to obtain formal results; in general, we can informally suggest that the chain will tend to converge more slowly when the acceptance probabilities are too low. This can occur when the probability distributions involved are sharply peaked, or when most of the proposed states conflict with the knowledge base or the evidence. Note, also, that the chain is not guaranteed to work at all if any of the conditions are not met.

We will now describe our suggested mechanisms for proposing new ιs. We present three such mechanisms: cluster-mixing, which is appropriate when few restrictions are imposed on ι; a mechanism based on bipartite matching, which is appropriate when we are asked to merge several sets of distinct terms; and a specialized mechanism for use on those aspects of ι where the probability distributions are defined in the form of reference distributions.

5.3.1 The cluster-mixing algorithm

The **cluster-mixing algorithm**, shown here in Figure 5.2, takes the current identity clustering and proposes a new one by, first, selecting, with replacement, two clusters of terms corresponding to two objects. The distribution $q(i_i)$ used to sample these clusters is not important, as long as there is some probability of picking each cluster. It is, of course, possible to pick the same cluster twice. Once the clusters have been selected, their terms are combined, and then divided u.a.r. into two newly-created clusters. Note that there is some probability that one of these new clusters will be empty, so that the two selected clusters will, in effect, be merged; note also that, if the same cluster is picked twice, it will probably be split.

`Sample two clusters` $i_1, i_2 \in \iota$ `from` $q(i_i)$

`Create two empty clusters` i'_1 `and` i'_2

`Place each term` $i \in i_1 \cup i_2$ `u.a.r. into` i'_1 `or` i'_2

`Propose` $\iota' = \iota - \{i_1, i_2\} \cup \{i'_1, i'_2\}$

Figure 5.2. The cluster-mixing algorithm for proposing new values of ι.

The proposed ι' may be invalid given the knowledge base (if it contradicts the *alldiff* statements), or if it is in conflict with the evidence (if the new clusters match together terms that are known to have different attribute values, or fail to match together terms that are known to be equal.) In that case, the move to ι' is impossible and the current transition becomes a self-loop. Otherwise, the proposal probability of a new ι' will be

$$q_\iota(\iota'|\iota) = \frac{q(\iota_1)q(\iota_2)}{2^{|\iota_1|+|\iota_2|}} .$$

Note that, if there is some chance of proposing ι' given ι, there will also be some chance of proposing ι given ι': each step of the cluster-mixing algorithm is reversible. This algorithm thus meets condition 4 of Lemma 5.3.1. Let us assume that the conditions 2 and 3 (which constrain the probability distribution and the proposal distribution associated with $\mathbf{V}$) hold as well. We can then use Lemma 5.3.1 to prove that, as long as the new state $\omega' = \langle n, \iota', \mathbf{V}' \rangle$ is a consistent state, it has a non-zero probability of being accepted. That fact will help us prove the following theorem.

Theorem 5.3.4: The cluster-mixing algorithm can be used to construct a positive-probability path connecting every pair ι_s, ι_t of possible values of ι, as long as $P(\iota|n, \mathbf{X})$, the distribution over ι given n and $\mathbf{X} \subset \mathbf{V}$ defined in the knowledge base, is non-zero for every state $\langle n, \iota, \mathbf{V} \rangle$ consistent with the knowledge base and the evidence.

Proof: Let us begin by defining a **minimal identity clustering** $\hat{\iota}$ as the unique identity clustering where only those terms that have been explicitly declared to be equal refer to the same object. We then propose the following path for moving from ι_s to ι_t:

- Split all the clusters in ι_s until $\hat{\iota}$ is reached.
- Merge all the clusters in $\hat{\iota}$ until ι_t is reached.

Let us now prove that each of the states encountered along the path is a consistent state: in this setting, this means that each of the ιs encountered along the way is consistent with the knowledge base and the evidence. We can demonstrate this for the first part of the path by showing that each such state is consistent with the knowledge base and the evidence; the constraint on $P(\iota|n, \mathbf{X})$ will then ensure that the probability of that state is non-zero. One of the ways in which an ι can conflict with the knowledge base or the evidence is by clustering together two terms that cannot be equal. However, since ι_s is consistent with both, we can be sure that ι_s never clusters together any such pair of terms; and since no state along this path can ever cluster together any two terms that are not clustered together in ι_s, this must hold for all these states as well. The other way in which an ι can clash with the evidence is by failing to cluster together terms that have been set to be equal. Since both $\hat{\iota}$ and ι_s are guaranteed to cluster such terms together, no state along this path ever needs to split them. Therefore, every state along the path is consistent with the knowledge base and the evidence, and is therefore a consistent state.

The proof for the second part of the path is symmetric: we can make the same claims for ι_t as for ι_s. Thus, every ι suggested along the path is consistent and so, as we proved above, has a positive acceptance probability. If every step along the path has a positive probability, then clearly so will the whole path. ■

Now that we know that our mechanism for suggesting new ιs can connect any two values of ι together, we can satisfy condition 3 of Theorem 5.3.3, and hence prove that a Markov chain built around the cluster-mixing algorithm can be used to perform approximate inference.

A Little Bit Of Experimental Fun

We have applied the cluster-mixing algorithm to a problem set in the example ball and urn domain of Figure 3.2. The query concerned the number of balls in the urn: it was phrased as a set of sentences stating that there are exactly l balls in the world, for $1 \leq i \leq n_{MAX}$. (Clearly, only one sentence was true at a time.) The evidence given always contained an even number of terms representing observed balls, half of which were black, and half of which were white. The prior over the number of balls was a uniform distribution over the integers $[1 \ldots 8]$.

In order to test our algorithm, we calculated the exact answer for up to forty observations. As we have explained in Section 2.3.1, the probability of a query is the sum of the probabilities of all the worlds that agree with both the query and the evidence over the sum of the probabilities of all the worlds that agree with just the evidence. So, we calculated the probabilities of the evidence at all values of $n \in [1, 8]$, and summed them to

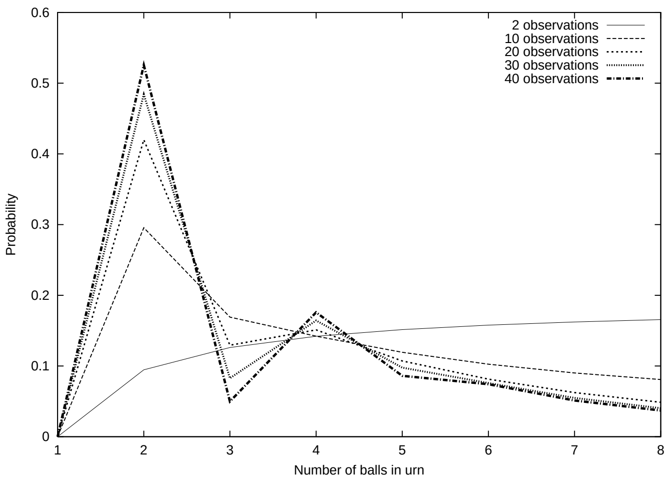


Figure 5.3. The exact posterior over the number of balls in the urn as the number of observations increases; in each set of observed instances, half the observations are white, and half are black.

get the normalization constant. In order to do the calculation, we have to remember that, according to Equation 3.3 and Equation 3.4.1, when o observations are drawn u.a.r. with replacement,

$$P(\iota|n) = \frac{n!}{(n-|\iota|)!}\frac{1}{n^o} .$$

Also, if a ball has an equal probability of being each colour,

$$P(\mathbf{V}_{\mathit{Colour}}|\iota) = 0.5^{|\iota|} .$$

Therefore, given that $\mathbf{I}_n$ represents the set of ιs corresponding to a given value of n, the summation looks as follows:

$$\begin{aligned} \sum_{\iota \in \mathbf{I}_n} P(n)P(\iota|n)P(\mathbf{V}_{\mathit{Colour}}|\iota) &= P(n)\sum_{\iota \in \mathbf{I}_n} \frac{n!}{(n-|\iota|)!}\frac{1}{n^o}0.5^{|\iota|} \\ &= P(n)\frac{n!}{n^o}\sum_{\iota \in \mathbf{I}_n}\frac{0.5^{|\iota|}}{(n-|\iota|)!} \qquad (5.2) \end{aligned}$$

The possible ιs can be (inefficiently) generated by generating all the possible partitions of the observations of each colour, dismissing those that contain more than n objects, and

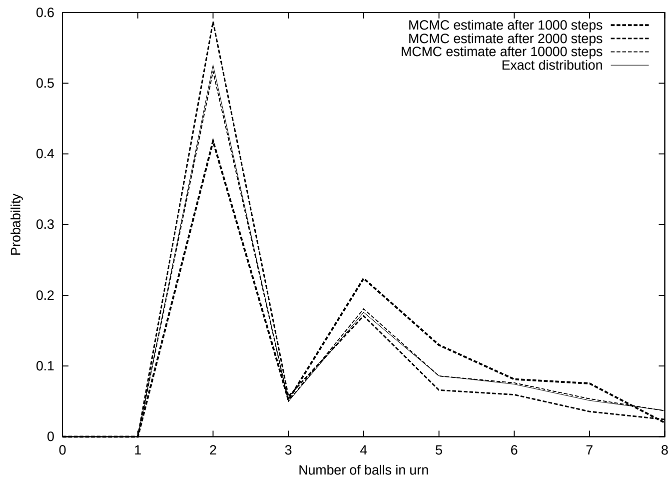


Figure 5.4. The posterior over the number of balls in the urn as approximated by our MCMC algorithm using increasing sample sizes

taking the cross-product of the two colours. The resulting posterior distributions are shown in Figure 5.3. Unsurprisingly, as the number of observations is increased, the posterior converges to a multi-modal distribution, with local maxima at the multiples of two. The results obtained when we applied algorithm to the forty-observation version of the problem are displayed in Figure 5.4. It should be clear that the estimated distribution tends to the exact distribution as the number of MCMC steps (and samples) increases.

5.3.2 Multipartite Matchings

Consider a scenario where we are asked to combine several databases, or several data sets obtained using several different sensors. There exist many problems that can be stated in those terms, from database merging to object tracking; Figure 3.3 shows a ball and urn example of such a scenario. More formally, we are given several sets of terms, where the terms within each set are known to be distinct from each other. We are then asked to reason about the ways in which these terms might co-refer, and the ways in which their attributes might be merged. The cluster-mixing algorithm will not work very well in this setting, since it will frequently propose clusters combining terms that are members with the same set. Instead, we will propose an alternative based on the bipartite matching algorithm presented in Section 4.2.4.

Bipartite Matchings

When there are only two sets of terms, $\mathbf{T}^1$ and $\mathbf{T}^2$, the algorithm can be set up as follows. We define $\mathbf{E} = \{(T^1, T^2) : T^1 \in \mathbf{T}^1, T^2 \in \mathbf{T}^2\}$ as the set of possible term-to-term pairings. Since some of these pairings might be inconsistent with the evidence, either because they pair together terms with conflicting attributes, or because they fail to pair together terms known to be equal, we also define $\mathbf{E_O} \subset \mathbf{E}$, the set of matchings consistent with the evidence. $\mathbf{E_O}$ encodes the set of edges of a bipartite graph with vertices $\mathbf{T}^1$ and $\mathbf{T}^2$, one where each bipartite matching $\mathbf{M}$ for this graph will correspond to an ι consistent with the evidence. We can use the algorithm presented in Figure 5.5 to propose new $\mathbf{M}$s, and hence new ιs.

It is easy to show that, if $\mathbf{M}'$ can be proposed given $\mathbf{M}$, then $\mathbf{M}$ can be proposed given $\mathbf{M}'$. If $\mathbf{M}'$ was constructed using one of the first two cases, then we can return to $\mathbf{M}$ by selecting the same (T^1, T^2); if the third case was used, we can do so by selecting M. Thus, the bipartite matching algorithm meets condition 4 of Lemma 5.3.1. If we assume that conditions 2 and 3 hold as well, we can, again, use that lemma to prove that any consistent new state $\omega' = \langle n, \iota', \mathbf{V}' \rangle$ constructed using the bipartite matching algorithm has a non-zero probability of being accepted, and then go on to prove the following theorem.

Theorem 5.3.5: The bipartite matching algorithm can be used to construct a positive-probability path connecting every pair ι_s, ι_t of possible values of ι, as long as $P(\iota|n, \mathbf{W})$ is non-zero for every state $\langle n, \iota, \mathbf{V} \rangle$ consistent with the knowledge base and the evidence.

```
Select a matching (T^1,T^2) ∈ E_O u.a.r.
if the matching (T^1,T^2) is in M then
  Remove it to get M' = M − (T^1,T^2)
else if Neither T^1 nor T^2 is matched in M then
  Add (T^1,T^2) in to get M' = M ∪ (T^1,T^2)
else if Either T^1 or T^2 is involved in a matching M ∈ M then
  Replace M with (T^1,T^2) to get M' = M − M ∪ (T^1,T^2)
else
  Let M' = M
end if
Propose M'
```

Figure 5.5. The bipartite matching algorithm. This algorithm is appropriate only when the set of all terms $\mathbf{T}$ can be split into two disjoint sets of distinct terms.

Proof: Consider two consistent states: starting state ι_s and end state ι_t. The path constructed will be very similar to the path proposed in the previous section: first, we will split all the pairs in ι_s until we reach the minimal identity clustering $\hat{\iota}$, and then we will add in all the pairs of ι_t until we reach that state. Every pairwise matching in every state along this path has to be part of either ι_s or ι_t, both of which are guaranteed to contain only pairs that are consistent both with the knowledge base and with the evidence. In addition, all the pairs that are forced to be equal by the evidence remain equal throughout. As a result, every state along this path is guaranteed to be consistent with the knowledge base and the evidence, and also, by the assumption made in the theorem itself, to have a non-zero probability: every state along this path must be consistent. We have just shown that transitions to consistent states always have positive acceptance probabilities. If every step along the path has a positive probability, then clearly so will the whole path.■

This result lets us satisfy condition 3 of Theorem 5.3.3, and hence prove that a Markov chain built around the bipartite-matching algorithm to propose all transitions in ι can be used to perform approximate inference—given that the set of terms $\mathbf{T}$ is split into two sets of distinct terms.

We can make some additional claims regarding the running time of the chain using the MCMC convergence result of Section 4.2.4. However, before we can use this result, we have to define the probability distribution over ι as the product of some *edge weights* corresponding to pairs of matched terms, and some *vertex weights* corresponding to single unmatched terms, and then guarantee that none of these weights are zero. Generally, these weights will correspond to factors of the distribution over δ, which can then be converted to a distribution over ι using Theorem 3.4.1.

In an ordinary record merging domain, where terms are assumed to be sampled from the set of objects, the probability that an object is present in one set does not directly depend on whether it is present or absent in other sets.[2] In this setting, the factorization isn't very interesting, since the edge weights are simply products of the corresponding vertex weights. Each "vertex weight" associated with an unmatched term can be rewritten as

$$w_V(T^t) = P(T^t \text{ is a sample of an object } i) = P(\delta^\omega(T^t)[i] = \{T^t\}|\mathbf{X}, n)$$

and each "edge weight" associated with a matching can be rewritten as

$$\begin{aligned} w_E(T^1, T^2) &= P\big(\text{both } T^1 \text{ and } T^2 \text{ are samples of } i\big) \\ &= P(\delta^\omega(T^1)[i] = \{T^1\}|\mathbf{X}, n)P(\delta^\omega(T^2)[i] = \{T^2\}|\mathbf{X}, n)\,. \end{aligned}$$

Dynamic domains, where the set of objects observed in the world can change from time-step to time-step, provide a more interesting example. Consider a setting with two

[2] Although an indirect dependency can exist, since the probability of being sampled can depend on some subset $\mathbf{X}$ the variables in $\mathbf{V}$.

time-steps, where $\mathbf{T}^1$ represents the terms observed at the first time step, and $\mathbf{T^2}$—those observed at the second. As in Section 3.6, we will let $\delta^t[i] \equiv \delta^\omega(\mathbf{T}^t)[i]$ represent the subset of the terms in $\mathbf{T}^t$ mapping to object i. This will be either a singleton, or an empty set, depending on whether the object has been observed at time t. The distribution over the δs can be factored as

$$\prod_{i=1}^{n} P(\delta^1[i]|\mathbf{X}, n)P(\delta^2[i]|\delta^1[i], \mathbf{X}, n) \, .$$

The first distribution is not relevant here, as it does not depend the $\mathbf{T}^1$-to-$\mathbf{T}^2$ matching. The weights are expressed in terms of the second distribution. The "vertex weights" can be written as

$$w_V(T^2) = P(\text{object } i \text{ enters}) = P(\delta^2[i] = \{T^2\}|\delta^1[i] = \{\}, \mathbf{X}, n)$$

and

$$w_V(T^1) = P(\text{object } i \text{ leaves}) = P(\delta^2[i] = \{\}|\delta^1[i] = \{T^1\}, \mathbf{X}, n) \, ,$$

and each "edge weight" associated with a matching can be rewritten as

$$w_E(T^1, T^2) = P(\text{object } i \text{ remains}) = P(\delta^2[i] = \{T^2\}|\delta^1[i] = \{T^1\}, \mathbf{X}, n)$$

As we proved in Section 4.2.4, our bound on the running time of a Markov chain over weighted bipartite matchings is polynomial in the size of the bipartite graph. In addition, it involves a ratio of edge and vertex weights. If $\hat{w}$ represents the maximum value a weight can take, and $\check{w}$ — the minimum values, then this ratio is

$$\frac{\hat{w}_E^3 \hat{w}_V}{\check{w}_E \check{w}_V^5} \, .$$

Thus, our bound suggests that the chain will run slower as the edge values get more extreme.

Increasing the Number of Sets

Of course, this algorithm applies only when there are exactly two sets. When the number of sets to be merged is greater, we suggest that the notation be expanded so that $\mathbf{E}^{ij}$ represents the set of pairings possible between the sets $\mathbf{T}^i$, $\mathbf{T}^j$ s.t. $i < j$, $\mathbf{E}^{ij}_O \subset \mathbf{E}^{ij}$ represents those pairings that are consistent with the evidence, and $\mathbf{M}^{ij}$ is a bipartite matching for a graph with edges $\mathbf{E}^{ij}_O$. There is an obvious relationship between the set of $\mathbf{M}^{ij}$s and an ι, which can be written as

$$\forall T^i \in \mathbf{T}^i, T^j \in \mathbf{T}^j \; s.t. \; \{T^i, T^j\} \in \iota \;\; \Rightarrow \;\; (T^i, T^j) \in \mathbf{M}^{ij} \; .$$

Note that, while every ι can be represented as a set of $\mathbf{M}^{ij}$s, not every set of $\mathbf{M}^{ij}$s will have an equivalent ι. It might, for example, be the case that two of the matchings impose a clustering that is inconsistent with a third: consider the situation where $(T^1, T^2) \in \mathbf{M}^{12}$ and $(T^1, T^3) \in \mathbf{M}^{13}$, but $(T^2, T^3) \notin \mathbf{M}^{23}$. We will call a set of $\mathbf{M}^{ij}$s with a corresponding ι **coherent**.

```
Pick two sets Tⁱ, Tʲ s.t.  i < j
Suggest a new Mⁱʲ using the bipartite matching algorithm
Construct an ι' by applying the suggested change to ι
if ι' is coherent and consistent with the evidence then
  Propose ι'
else
  Propose no changes
end if
```

Figure 5.6. A multipartite matching algorithm.

We then suggest algorithm of Figure 5.6,which applies the bipartite-matching algorithm to a randomly-chosen pair of sets and uses the proposed change to construct a new identity mapping. This new ι is constructed as follows:

- If the proposed change simply adds a matching (T^1, T^2), we attempt to merge the clusters currently containing T^1 and T^2. (This might easily lead to a non-coherent ι, if T^1 and T^2 currently have conflicting matches in some other set.)

- If the proposed change simply removes a matching (T^1, T^2), we take the cluster to which both T^1 and T^2 currently belong and split it into two separate clusters, one containing T_1, and one containing T_2. This can be done by deciding where each of the other elements of the cluster should go u.a.r. (The resulting ι will necessarily be coherent, but might be in conflict with the evidence if we separate a pair of terms known to be equal.)

- If the proposed change replaces one matching with another, we, again, split the cluster to which the old matching belongs into two new clusters, one containing the new matching, and one containing the newly unmatched vertex.

In order to prove that a chain based on this algorithm will approximate the correct distribution, we will, once again, have to show that the ι components of any two consistent states are connected by a positive-probability path. We will propose the usual type of path, one that passes through the minimal identity clustering $\hat{\iota}$: first, we apply as many splits as possible to the starting ι_s, and then we construct ι_t using merges. Using the same argument as before, if the starting state and the end state are both consistent with the knowledge base and the evidence, then so is every state along the path. Let us now make the additional assumption that, if some cluster (multipartite matching) of terms has a positive probability, then so does every subset of that cluster. Under that assumption, every state along the path will have a positive probability.

5.3.3 Reference Transitions

As we have discussed in Section 3.4, the distribution over the identity mappings of a functional term T can be expressed as a distribution over other terms with which it can co-refer. As an example, let us, again, decide each balls is decorated with a pictures of another ball, and augment the ball and urn language with the function *Picture*, which tells us which ball the picture represents. We might then be told that the pictures refer to B_1 and B_2 with equal probability, as in

$$P(\{\textit{Picture}(x) = B_1, \textit{Picture}(x) = B_2\}) = \{0.5,\ 0.5\}\ .$$

Pfeffer (2000) defines several distributions over reference slots in this manner. We will refer to the unique constant term associated with a functional term as its **assignment**. It should be clear that each constant term can be an assignment for any number of functional terms.

The algorithms proposed in the previous two sections can fail here. At any rate, the proofs we have just presented will not be applicable, because, in this setting, there is no unique minimal identity clustering: all the terms T can co-refer with affect the size of the clustering in the same way, and there is no way of choosing among them. Instead, we offer the algorithm of Figure 5.3.3, which suggests explicit changes to the assignments of the functional terms. It proposes a new assignment for a randomly selected functional term, and then moves the functional term from its current cluster to that of its new assignment. The algorithm is clearly reversible in the sense that satisfies condition 4 of Lemma 5.3.1: it is trivial to propose that a functional term's assignment be moved back to its previous value. Assuming that conditions 2 and 3 hold as well, we can, once more, use the lemma to prove that any consistent new state $\omega' = \langle n, \iota', \mathbf{V}' \rangle$ constructed using our ι-proposing mechanism has a non-zero probability of being accepted, and then go on to prove the following theorem.

```
Pick a functional term $F \in \mathbf{F}$, currently in cluster $\iota_f$
Pick a constant term $C \in Val[F]$, currently in cluster $\iota_c$
Let $\iota'_f = \iota_f - F$
Let $\iota'_c = \iota_c + F$
Suggest $\iota' = \iota - \iota_f - \iota_c + \iota'_f + \iota_c$
```

Figure 5.7. An algorithm for proposing new values of ι when identity uncertainty is modeled using reference distributions.

Theorem 5.3.6: The reference uncertainty algorithm can be used to construct a positive-probability path connecting every pair ι_s, ι_t of possible values of ι, as long as the following conditions hold for all the functional terms:

1. None of them occur in an *alldiff* statement in the knowledge base.

2. No pair of them is set to be equal in the evidence.

3. None of their attributes have their values set by the evidence.

Proof: Consider two consistent states: starting state ι_s and end state ι_t. We can construct a path from ι_s to ι_t by going through all the functional terms $F \in \mathbf{F}$ and making sure that the assignment of each one conforms to its ι_t value. We can show that every state along this path is consistent by showing that, given a consistent state, moving a functional term F to a new cluster does not make the state inconsistent. There are three ways in which the new state could introduce an inconsistency.

- It could contradict an equality statement. However, this cannot happen here. According to condition 2, functional terms cannot be equal to other functional terms, and, if a functional term is equal to a constant term, then it will already be in the correct cluster and there will be no need to move it.

- It could attempt to combine two terms forced to be unequal, whether because of conflicting variable assignments in the evidence, or because of an *alldiff* statement in the knowledge base. By conditions 1 and 3, neither of those two situations can occur here.

We have just shown that transitions to consistent states always have positive acceptance probabilities. If every step along the path has a positive probability, then clearly so will the whole path.■

This result lets us satisfy condition 3 of Theorem 5.3.3, and hence prove that a Markov chain built around the reference uncertainty algorithm to propose all transitions in ι can be used to perform approximate inference. Note, however, that we have made some rather strong assumptions about the contents of the evidence.

5.3.4 Conclusion

We have presented three different methods for proposing transitions in ι, and shown under what circumstances each method might be appropriate. It is worth noting that in complex domains where several different types of identity uncertainty are present, it might be appropriate to use a mixture of these methods. Moreover, it should be clear that there exist many other possible algorithms, different from the three described here. When there are few restrictions on ι, one alternative to the cluster-mixing algorithm might be to propose only merges and splits; and, in the multipartite matching scenario, we might consider changing several bipartite matchings at once. We hope that the proofs presented in this chapter might provide helpful guidelines for analyzing such alternative algorithms.

Chapter 6

Identity Uncertainty and Probabilistic Relational Models

Probabilistic Relational Models are a first-order probabilistic language (Koller & Pfeffer, 1998; Pfeffer, 2000; Getoor, 2001). Like semantic networks, they organize information in terms of object classes and object instances, and are thus a very appropriate representation for domains where we wish to reason about the properties of objects and their identities. In this chapter, we describe how our approach to identity uncertainty can be applied to PRMs. We also propose an MCMC-based inference algorithm for PRMs, including those which feature identity uncertainty, basing it on the algorithms of Chapter 5. We demonstrate that this algorithm is somewhat useful when reasoning about the complex Bayesian networks equivalent to generic PRMs, and then enormously useful once identity uncertainty—and, hence, the large state space of identity clusterings—enters the picture.

6.1 The General Approach

As we explained in Section 2.1.2, a relational knowledge base consists of a relational schema and a skeleton – or, alternatively, a typed relational language and a set of constraints. The possible worlds valid for this knowledge base are represented as sets of ground attribute values, and a probabilistic model over all these possible worlds is then defined in terms of local probabilistic models (quantified Bayesian network fragments) associated with the various attributes of the various classes. Our approach to identity uncertainty, as presented in Chapter 3, expands the definition of possible worlds, so that each possible world includes the number of objects and the term-to-object mapping as well as the values of the ground attributes. When adapting this general approach to PRMs, we take the classes into account by factoring both the number of objects, and their identity mappings or clusterings, by class. For now, we will disallow inheritance: our classes simply separate the terms and the objects into disjoint sets, so that terms of different classes cannot co-refer. With this in mind, we alter our definition of simple possible worlds, Definition 3.1, as follows:

Definition 6.1: A **simple possible world** for a PRM is a tuple $\omega = \langle \mathbf{n}, \delta, \mathbf{V} \rangle$, where

- $\mathbf{n} = \{n_{C1} \dots n_{C|\mathbf{C}|}\}$ is a set where each element n_C represents the cardinality of the class C.
- $\delta = \{\delta_{C1} \dots \delta_{C|\mathbf{C}|}\}$ is a set of identity mapping vectors. Each element δ_C is a vector of clusters of terms where the ith cluster contains the terms that refer to the ith object belonging to class C.
- $\mathbf{V}$ is a set of arrays keeping track of the value assignments for all the possible applications of all the attributes and reference slots of the language.

Our definition of compact possible worlds, Definition 3.2 is altered analogously.

Definition 6.2: A **compact possible world** for a PRM is a tuple $\omega = \langle \mathbf{n}, \iota, \mathbf{V} \rangle$, corresponding to a non-empty set of simple possible worlds, where

- $\mathbf{n} = \{n_{C1} \dots n_{C|\mathbf{C}|}\}$ is a set where each element n_C represents the cardinality of the class C.
- $\iota = \{\iota_{C1} \dots \iota_{C|\mathbf{C}|}\}$ is a set where each element ι_C represents the identity clusterings of the terms $\mathbf{T}_C$ belonging to class C.
- $\mathbf{V}$ is a set of arrays keeping track of the value assignments for all the possible applications of all the attributes of the language.

It is worth noting that, in a compact possible world, the values of the reference slots are not represented explicitly: the clusters in ι mix together functional terms and constants. However, the function mappings implied by ι are guaranteed to make sense, and can be recovered from ι quite easily.

Of course, the above definitions are incomplete without a specific underlying language defining the classes, instances, and attributes. In this chapter, we will use various versions of the typed relational languages of Getoor (2001) and Pfeffer (2000), all of which make the assumption that the set of language elements is finite. In addition, in order to be able to use compact possible worlds, we will be assuming that the set of terms of the language is finite. We enforce this assumption by imposing an ordering $\prec$ on all the reference slots. This ordering is defined as follows:

$$A_1 \prec A_2 \Leftrightarrow \mathit{Range}[A_1] = \mathit{Dom}[A_2] \; .$$

In other words, A_2 is a reference slot of a class to which reference slot A_1 refers.

Theorem 6.1.1: If the ordering $\prec$ contains no cycles, then the set of terms is finite.

Proof: By definition, for every legal slot chain $A_1.A_2. \; \dots \; .A_n$, $\mathit{Dom}[A_i]$ must match $\mathit{Range}[A_{i-1}]$ for every $i \geq 2$: in other words, it must be true that $A_i \prec A_{i-1}$. Thus, all legal chains must trace a path through the ordering $\prec$. When this ordering contains no cycles, all the paths, and hence all the chains, will be finite. Since the number of reference slots is also finite, it follows that there is a finite number of slot chains; and, since the number of constant terms is finite as well, there can be only a finite number of functional terms formed by applying slot chains to constants. ■

It is worth noting that, when we impose this ordering, we disallow recursive definitions and inverses.

In this chapter, knowledge bases all contain a full description of a PRM. Some might also, like the knowledge bases of Chapter 3, contain some *alldiff* statements, which can are applied to sets of terms and impose a full set of inequality constraints. Once again, we impose the restriction that each term can be part of only one such constraining statement; in addition, all of the terms a constraint is applied to must be of the same class. When an *alldiff* statement is applied to all the terms in some class C, it effectively imposes the unique names assumption on that class. Evidence is added in the form of a set of attribute and reference slot assignments; queries are ground facts. In Section 6.2, we consider PRMs without identity uncertainty. In Section 6.3, we introduce reference uncertainty, which is essentially identity uncertainty over functional terms, and discuss it in terms of our approach. In Section 6.4, we extend PRMs, allowing them to handle identity uncertainty over constant terms.

6.2 No Identity Uncertainty

We will begin by explaining how everything works when there is no uncertainty over n or ι. This can be enforced by applying *alldiff* statements to the instances of each class. Such a simple situation is still of interest, as it permits us to present an algorithm for proposing $\mathbf{V}$ transitions. It also draws attention to the fact that PRMs provide us with a

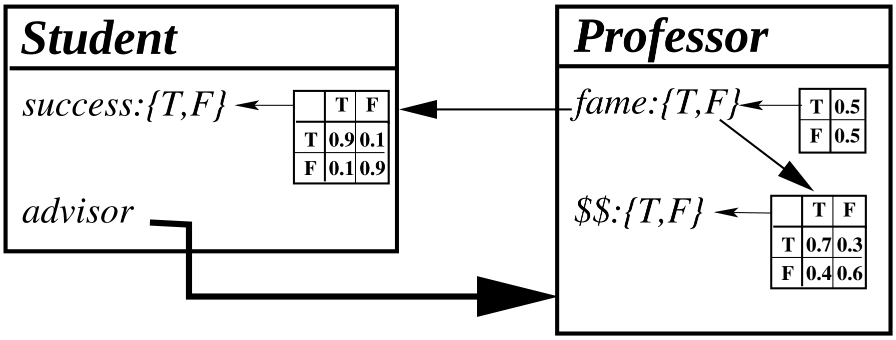


Figure 6.1. The simple PRM defined in the text, with all the associated conditional probability models.

compact and expressive method for modeling the distribution over $\mathbf{V}$. We do, of course, have to make sure that this model is well-defined: as we explained in Section 2.3, this can be done by checking the class dependency graph for acyclicity.

Example 6.1: As we return to PRMs, we also return to the University domain of Section 2.1.2, although we will now use a more complex example. Consider the probabilistic relational model shown in Figure 6.1. There are two classes, *Student* and *Professor*. The *Student* class has one reference slot, *Advisor*, which ranges over *Professor*s, and one Boolean descriptive attribute, *Success*. The *Professor* class has two Boolean attributes, *Fame* and *$$* (funding level). For any *Student* s, s.*Success* has one parent, s.*Advisor*.*Fame*, with an appropriate conditional distribution. For any *Professor* p, p.*Fame* has no parents and a simple prior distribution, while p.*$$* depends on p.*Fame*. We define a skeleton for this PRM by stating that there are two instances, $Student_1$ and $Prof_1$, and that $Student_1.Advisor = Prof_1$.

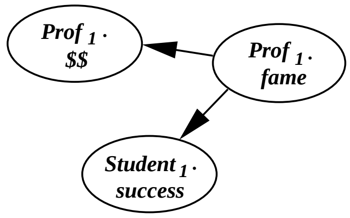


Figure 6.2. Bayesian network structure generated from the PRM in Figure 6.1 and the skeleton of Example 6.1. The *Advisor* relationship, being certain, doesn't appear.

6.2.1 Inference

When ι and n are known, we can represent the probability model over possible worlds using the single Bayesian network corresponding to $\mathbf{V}$. Since the structure of this network is fully defined, we say that there is no **structural uncertainty**. Inference is then simply inference on this Bayesian network, or at least those parts of it that are relevant. (Nodes that are not ancestors of the attributes referred to by the evidence and the query are not relevant.) Figure 6.2 shows the complete Bayesian network equivalent to our example.

Assuming that the Bayesian network contains the nodes $\mathbf{V} = V_1 \ldots V_n$, where $\mathbf{O}$ are the observed nodes corresponding to the evidence, and $\mathbf{H} = \mathbf{V} - \mathbf{O}$ are the hidden nodes, we can define a simple MCMC-based algorithm by picking a node in $\mathbf{H}$ and then applying Gibbs sampling to it, so that each step of the algorithm looks as follows:

```
Choose V_i ∈ H from random distribution q_H
Sample a new value for V_i from P(V_i|MB_i)
```

Consider the Markov chain $\mathcal{M}$, where the state space consists of all the values of $\mathbf{H}$ such that $P(\mathbf{H}|\mathbf{O}) > 0.0$, and the transition function is defined using this algorithm.

Lemma 6.2.1: Every pair of states in the state space of $\mathbf{M}$ can be connected by a path of non-zero probability, as long as none of the nodes in $\mathbf{V}$ that have parents have CPFs with zero values.

Proof: Let s and t be any pair of states. One possible way of defining a path from s to t is by flipping each variable in turn from its value in s to its value in t. The probability of every such path will be the product of all the flip probabilities. If any of the flips has a zero probability, then so will the path. Now, let us consider the probability of a single flip, one involving the variable V_i. This variable will be drawn from the distribution

$$P(V_i|MB_i) = P(V_i|Pa_i) \prod_{V_j \in Ch_i} P(V_j|Pa_j) \, .$$

We have to show that $P(V_i|MB_i)$ is non-zero for all possible values of $\mathbf{V}$. If none of the CPFs have zero values, this will clearly be the case. Moreover, permitting the CPFs of nodes without parents to have zero values will not change this, because these entries will never enter into this product. Parent-less nodes obviously cannot be counted among the children of V_i; if V_i itself has no parents, then the value it takes in s (and hence the value it is being flipped to) must have a non-zero probability. ■

Our reason for emphasizing that parent-less nodes can have zero CPF values is that we can transform any Bayesian network with zero CPF values into one where such values occur only in unconditional priors. This can be accomplished by merging every node which has zero CPF values with its parents to form a new node, one whose state space is the product of the state spaces of all the combined nodes, and whose CPF is calculated accordingly (and obviously still contains zeroes.) Eventually, we will run out of parents; at that point, the CPF will be an unconditional prior. However, the state space of the resulting node can be quite large in complex networks, so this approach is not always helpful.

At any rate, we can now prove the following:

Theorem 6.2.2: The Markov chain $\mathcal{M}$ can be used to approximate expectations over the distribution $\pi(\mathbf{H}|\mathbf{O})$, as long as none of the nodes in $\mathbf{V}$ that have parents have zero values in their CPFs.

Proof: As we explained in Chapter 4, we can approximate expectations over a distribution $\pi()$ using an ergodic Markov chain with $\pi()$ as its stationary distribution. In order to prove that a chain is ergodic with the appropriate stationary distribution, we need to prove that the detailed balance condition holds with respect to this distribution, that every state has a non-zero self-loop probability, and that the chain is irreducible (that every pair of possible states is connected by a path of non-zero probability.) Where $\mathcal{M}$ is concerned, we can show the following:

- The detailed balance condition holds for every randomized Gibbs sampler, including $\mathcal{M}$.
- Every Gibbs sampler has a built-in self-loop probability, since every variable is sampled from its current conditional distribution. Again, this includes $\mathcal{M}$.
- According to Lemma 6.2.1, every pair of states of the chain $\mathcal{M}$ is connected by a path of non-zero probability, and so $\mathcal{M}$ is irreducible. ■

While Theorem 6.2.2 proves that the algorithm works, it says nothing about the running time. In Section 4.2.3, we have provided a far more complicated proof demonstrating that, for a subset of discrete Bayesian networks, the running time is polynomial in the size of the network. It is also highly dependent on the ratio of maximal and minimal CPF values.

On a more practical level, one possible enhancement for this algorithm is *lazy construction* of the Bayesian network. The network can, e.g, be grown "from the query outwards", with nodes added as they are needed to sample a node currently in the network.

6.3 Reference Uncertainty

In the original PRM formulation, reference uncertainty arises when relations among objects, as described by reference slots, are not known: instead, there is uncertainty over

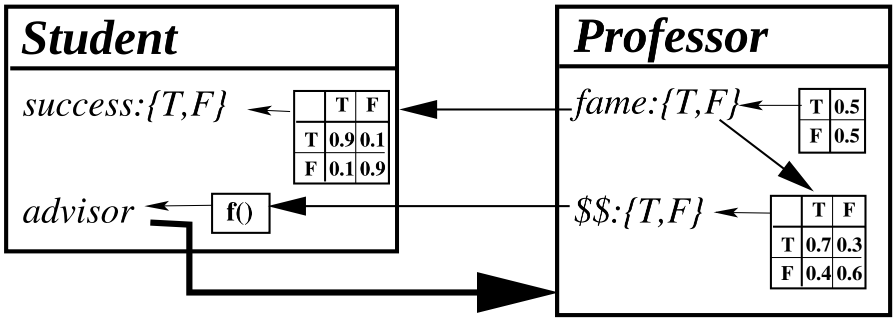


Figure 6.3. The simple PRM of Figure 6.1, augmented with reference uncertainty.

which object the reference slot refers to. This uncertainty is sometimes expressed as uncertainty over which constant term refers to the same object as the reference slot, but it can also be expressed using a function of the attributes of the candidate objects, as was done in Section 3.5.

For example, we may be unsure as to which of three professors is *Student*$_1$'s advisor, and have some distribution over the three possible professors. The simple example PRM of Figure 6.1 can be extended to handle this situation as shown in Figure 6.3. The language now contains three instances of the class *Professor*, *Prof*$_1$, *Prof*$_2$, and *Prof*$_3$. These are all known to be distinct, so the we place the statement

$$\mathit{alldiff}(\{\mathit{Prof}_1, \mathit{Prof}_2, \mathit{Prof}_3\})$$

in the knowledge base. The probability distribution over *Student.Advisor*, referred to using the cryptic function $f()$, ranges over all these professors. One possible way of defining this function would be to define a direct prior over (some subset of) the *Professor* instances: for example, by declaring that a student's advisor is equally likely to be *Prof*$_1$ and *Prof*$_2$. We would thus be stating that the advisor is equally likely to co-refer with *Prof*$_1$ and *Prof*$_2$. However, in the PRM of Figure 6.3 we have chosen to make the choice of a student's *advisor* dependent on the funding (*Prof.$$*) of each candidate. As a result, *Advisor.$$* is

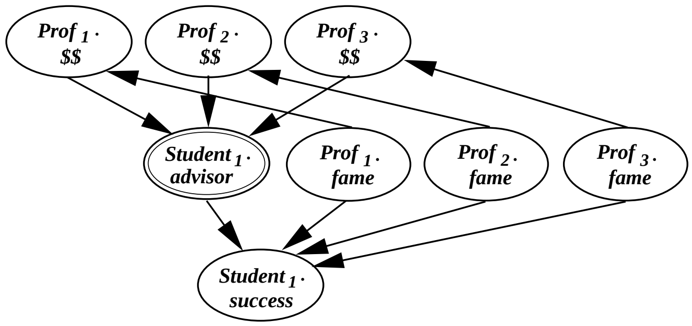


Figure 6.4. Bayesian network structure with reference uncertainty about *Student*$_1$.*Advisor*. The uncertain reference slot is shown in a double oval.

considered the parent of *Advisor*. We say that a professor with funding is three times more likely to be an advisor than one without. This can be stated using sampling potentials, as defined in Section 3.5.

When the value of a reference slot is uncertain and dependent on the attributes of the candidate instances, the number of dependencies in an equivalent Bayesian network can increase dramatically. Suppose that, according to the PRM, an attribute B depends on another attribute denoted by the slot chain $A_1.\cdots.A_n.B'$, where some of the reference slots A_i in the chain may be uncertain. Then, there is uncertainty over the actual object the slot chain $A_1.\cdots.A_n$ points to, and so over the object on whose B' attribute B should be dependent. As a result, when we construct the Bayesian network, B has to depend on the B' attributes of all the objects that might be reached by the all possible combinations of values of all the uncertain components of the slot chain $A_1.\cdots.A_n$, as well as on all the possible uncertain reference slots: these reference slots will be used to select the appropriate object and hence the appropriate B' attribute. Figure 6.4 presents the Bayesian network corresponding to the example of Figure 6.3. We are not sure who *Student*$_1$s advisor is, and so the parents of *Student*$_1$.*Success* have to include all the possible *Professor.Fame* attributes, as well as the *Advisor* slot, which is used to multiplex between them. Obviously, when

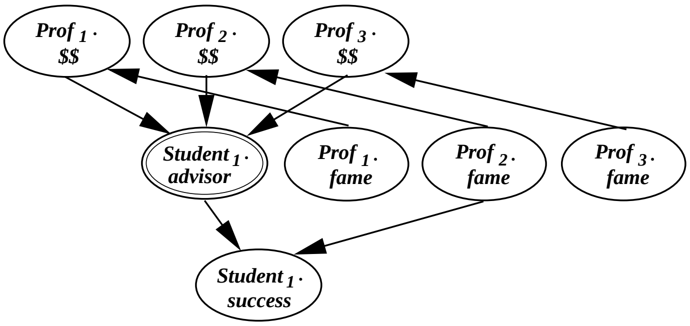


Figure 6.5. The network structure conditioned on $Student_1.Advisor = Prof_2$.

reference attributes have many possible values, very large implicit network structures can result.

It is worth noting that there does exist quite a lot of context-specific independence. Since the reference attributes act merely as multiplexors, when their values are set we know exactly which of the attribute chains matter, and the network dependencies look exactly as they would have looked without reference uncertainty. As an example, consider the network of Figure 6.5, where picking a value for the reference attribute $Student_1.Advisor$ has simplified the network of Figure 6.4.

6.3.1 Inference

To handle uncertain reference slots, we construct an inference algorithm using the proposal mechanism for reference transitions described in Section 5.3.3. When the constant terms are known to be distinct, and the distributions over all the uncertain functional terms are defined using reference distributions, we can simply use this mechanism to propose transitions in ι and the Gibbs steps of Section 6.2.1 to move through the space of $\mathbf{V}$. The formal results presented with the mechanism prove that, given our representation, this algorithm will approximate the correct distribution.

Although the algorithm was presented in the context of identity uncertainty, and of proposing changes to ι, it is perhaps even easier to view it simply in the context of uncertainty over function mappings. Suggesting a change to the "assignment" of one of the functional terms is equivalent to proposing a new value for the relevant reference slot. Either way, because the reference attribute values are known in every MCMC state, we can take advantage of the CSI present. When we access the Markov blanket of an attribute during a Gibbs step, we do not have to access all of $\mathbf{V}$, or even the part of ι corresponding to the mapping of the reference slots.

Consider an extended version of Figure 6.4 with n potential advisors, and let the current value of *Student*$_1$.*Advisor* be *Prof*$_i$. Suppose we want to apply Gibbs sampling to $\mathit{Prof}_j.f$ (*Prof*$_j$.*Fame*), where $j \neq i$. By Equation 4.2, the sampling distribution is

$$\begin{aligned}
&\alpha P(\mathit{Prof}_j.f)P(\mathit{Prof}_j.\$\$|\mathit{Prof}_j.f)P(\mathit{Student}_1.success|\mathit{Prof}_{1\ldots n}.f, \mathit{Student}_1.\mathit{Advisor}=\mathit{Prof}_i)\\
&= \alpha P(\mathit{Prof}_j.f)P(\mathit{Prof}_j.\$\$|\mathit{Prof}_j.f)P(\mathit{Student}_1.success|\mathit{Prof}_i.f, \mathit{Student}_1.\mathit{Advisor}=\mathit{Prof}_i)\\
&= \alpha' P(\mathit{Prof}_j.f)P(\mathit{Prof}_j.\$\$|\mathit{Prof}_j.f)\ .
\end{aligned}$$

The first step of this derivation uses the fact that $\mathit{Student}_1.success$ depends only on the fame of *Student*$_1$'s advisor, $\mathit{Prof}_i.f$, and not on other professors; the second step simply observes that $P(\mathit{Student}_1.success|\mathit{Prof}_i.f, \mathit{Student}_1.\mathit{Advisor}=\mathit{Prof}_i)$ is constant w.r.t. $\mathit{Prof}_j.f$. Thus, when the reference attribute is instantiated, sampling operates exactly as if the links from non-selected parents were nonexistent, as was shown inFigure 6.5. The same holds when sampling a child of the reference variable (e.g., $\mathit{Student}_1.success$).

Now suppose we want to change the value of *Student*$_1$.*Advisor*. Consider the transition from $\mathit{Student}_1.\mathit{Advisor}=\mathit{Prof}_i$ to $\mathit{Student}_1.\mathit{Advisor}=\mathit{Prof}_j$. If we assume that the proposal distribution is uniform and hence cancels, then, after the obvious cancellations, the relevant parts of the acceptance ratio will be

$$\frac{P(\mathit{Student}_1.\mathit{Advisor}=\mathit{Prof}_j|\mathit{Prof}_{1\ldots n}.\$\$)P(\mathit{Student}_1.success|\mathit{Prof}_j.f, \mathit{Student}_1.\mathit{Advisor}=\mathit{Prof}_j)}{P(\mathit{Student}_1.\mathit{Advisor}=\mathit{Prof}_i|\mathit{Prof}_{1\ldots n}.\$\$)P(\mathit{Student}_1.success|\mathit{Prof}_i.f, \mathit{Student}_1.\mathit{Advisor}=\mathit{Prof}_i)}\ .$$

In general, calculating this ratio can require accessing the current values of the funding levels of all possible candidate professors, or $\mathit{Prof}_1.\$\$, \ldots, \mathit{Prof}_n.\$\$$. After all, although the transition involves just two candidates, the probability of picking any one advisor *does* depend on the funding levels of all of them. However, because of the way we have defined the conditional distribution over the reference slot value, we can simplify the situation further. In general, if we force the distribution to take the form

$$P(\mathit{Student}_1.\mathit{Advisor} = \mathit{Prof}_i | \mathit{Prof}_1.\$\$, \ldots, \mathit{Prof}_n.\$\$) = \frac{f(\mathit{Prof}_i.\$\$)}{\sum_k f(\mathit{Prof}_k.\$\$)}$$

for any arbitrary function $f()$, we will find that the the summations $\sum_k f(\mathit{Prof}_k.\$\$)$ cancel in the transition probability ratio given above, leaving simply

$$\frac{f(\mathit{Prof}_j.\$\$)\mathit{Prof}(\mathit{Student}_1.success|\mathit{Prof}_j.f, \mathit{Student}_1.\mathit{Advisor} = \mathit{Prof}_j)}{f(\mathit{Prof}_i.\$\$)\mathit{Prof}(\mathit{Student}_1.success|\mathit{Prof}_i.f, \mathit{Student}_1.\mathit{Advisor} = \mathit{Prof}_i)}$$

which does not mention any values for the reference variable besides Prof_i and Prof_j. Sampling potentials take this form by definition; so do some other, well-known distributions, such as the *softmax* distribution,

$$P(X = i | Y_1 = y_1, \ldots, Y_n = y_n) = \frac{e^{y_i}}{\sum_k e^{y_k}} .$$

The softmax distribution, or rather a discretized version of it, might make a reasonable model for selecting an advisor based on funding.

A reasonably obvious way to implement all the context-specific independence is to keep track of the state using simplified networks, such as the one in Figure 6.5, keeping in mind that that the effective structure of the networks, and even the set of relevant nodes, can change whenever the value of the reference slots is altered, and updating the network structure as appropriate. If necessary, nodes may even be discarded when the attributes or slots they represent become irrelevant to the query under the current network structure. This chain might be viewed as moving through the the networks of a Bayesian multinet (Geiger & Heckerman, 1996).

Choosing new attribute values

As we explained in Chapter 5, whenever we propose a new ι, we must also propose new values for $\mathbf{U} \in \mathbf{V}$, the unobserved attributes of any objects who have been affected by the change in ι. One possible $q_{\mathbf{U}}(\mathbf{V}'|n, \iota', \mathbf{V})$ picks values for each $V_i \in \mathbf{U}$ u.a.r from $Val[V_i]$. This approach is simple to implement and yields an acceptance ratio that is easy to calculate. This is the algorithm that we test experimentally below. However, in general, this approach can be rather slow. We can, instead, pick new values from the distribution over $Val[V_i]$ given the current state, so that

$$q_{\mathbf{U}}(\mathbf{V}'|n, \iota', \mathbf{V}) = \prod_{V_i \in \mathbf{U}} P(V_i'|V_1' \dots V_{i-1}', V_{i+1} \dots V_{|\mathbf{V}|})$$

When calculating the acceptance probability, we will then have to evaluate the ratio

$$\frac{q_{\mathbf{U}}(\mathbf{V}|n, \iota, \mathbf{V}')}{q_{\mathbf{U}}(\mathbf{V}'|n, \iota', \mathbf{V})} = \prod_{V_i \in \mathbf{U}} \frac{P(V_i|V_1 \dots V_{i-1}, V_{i+1}' \dots V_{|\mathbf{V}|}')}{P(V_i'|V_1' \dots V_{i-1}', V_{i+1} \dots V_{|\mathbf{V}|})} \tag{6.1}$$

It is quite possible that little will cancel in the ratios corresponding to the individual V_is, since the numerators and the denominators do not have much in common. Also,

$$P(V_i|V_1 \dots V_{i-1}, V_{i+1} \dots V_{|\mathbf{V}|}) = \alpha P(V_i|Pa_i) \prod_{V_j \in Ch_i} P(V_j|Pa_j)$$

where calculating the normalizing constant α involves summing over all $v \in Val[V_i]$. Notice that the value of this normalizing constant will be different for $P(V_i|V_1 \dots V_{i-1}, V_{i+1}' \dots V_{|\mathbf{V}|}')$ and for $P(V_i'|V_1' \dots V_{i-1}', V_{i+1} \dots V_{|\mathbf{V}|})$, so that the two constants will *not* cancel in the ratio of Equation 6.1, and will need to be calculated explicitly. Performing all the summations – and hence calculating the ratio – may be quite time-consuming.

For example, consider a starting state where *Student*$_1$ refers to *Student* instance 1 while *Student*$_1$*.Advisor* refers to *Professor* instance 1, and a proposed state where the value of *Student*$_1$*.Advisor* is changed so that it now refers to *Professor* instance 2. When we propose this change to ι, we will simultaneously want to propose changes to the the uncertain attributes of these two objects. Let us say that everything about these professors is uncertain, so that we have to propose new values for both their fame and their funding values: that is, we have to propose new values for $\mathbf{V}'_{\mathit{\$\$}}[1]$, $\mathbf{V}'_{\mathit{\$\$}}[2]$, $\mathbf{V}'_{\mathit{Fame}}[1]$, and $\mathbf{V}'_{\mathit{Fame}}[2]$. Both the numerator and denominator in Equation 6.1 will include factors corresponding to all of these attributes. For $\mathbf{V}'_{\mathit{Fame}}[1]$, the factor in the numerator will be

$$\alpha P(\mathbf{V}_{\mathit{Fame}}[1])P(\mathbf{V}_{\mathit{Success}}[1]|\mathbf{V}_{\mathit{Fame}}[1])$$

and the factor in the denominator will be

$$\alpha P(\mathbf{V}'_{\mathit{Fame}}[1])P(\mathbf{V}_{\mathit{Success}}[1]|\mathbf{V}'_{\mathit{Fame}}[1])P(\mathbf{V}_{\mathit{\$\$}}[2]|\mathbf{V}'_{\mathit{Fame}}[1])\ .$$

It should be clear that the normalization constants will not cancel.

An alternative approach might be to run a "local" MCMC algorithm inside $q_{\mathbf{U}}()$ to generate attribute values that reflect the current values of the rest of $\mathbf{V}$ and hence are likely to be accepted. (When this method is used, care must be taken to ensure that the chain is run until convergence: stopping it prematurely may result in biased answers.)

6.3.2 Experimental Results

To check our algorithm, we applied it first to the knowledge base with reference uncertainty over three professors, where the algorithm generates and evaluates the network shown in Figure 6.4. We set values for fame or funding for each *Professor*, and queried the posterior probability of $\mathit{Student}_1.\mathit{Success}$. In this tiny model, we can compute the exact value for comparison. The MCMC estimates in Figure 6.6 clearly converge to the correct value.

We also tested the scalability of the algorithm by trying it out on various types of networks of increasing size. As an example, let us consider networks with n professors and $4n+1$ students, each of whom could have any one of the professors as advisor. We

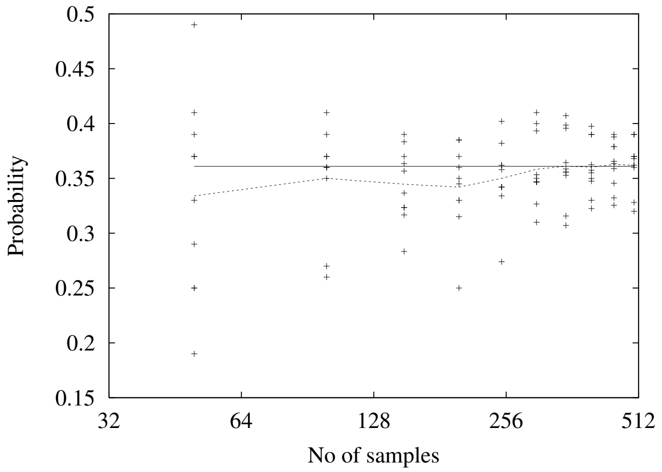


Figure 6.6. MCMC convergence on a simple example with three professors and one student. The exact probability is 0.361 (horizontal line). Each point represents the estimate from one of 10 Markov chains after a given number of samples (log scale). The dotted line shows the average of the 10 individual estimates.

specified the success (or otherwise) of $4n$ students and queried the success of the last. Figure 6.7 shows the inference cost as a function of the total network size. Because we cannot compute the exact values for all n, we use the standard convergence diagnostic due to (Gelman, 1996), checking against the exact values for small n. The resulting graph appears to be linear in the size of the network. It would appear that, for this type of network, the sampling algorithm scales well. Note also that cost is measured in terms of *single-variable* state transitions performed by the algorithm, hence the number of transitions *per variable* to reach convergence is approximately constant, regardless of network size.

6.4 Identity Uncertainty over Instances

Reference uncertainty is a type of identity uncertainty, one that permits us to define a distribution over ι using some of the probability distributions already associated with

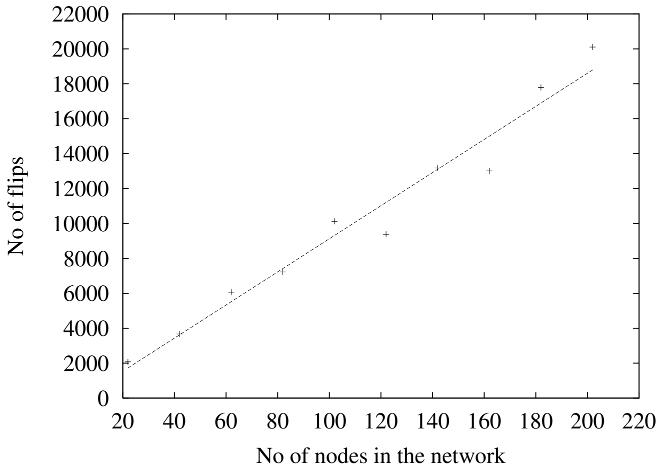


Figure 6.7. MCMC convergence as a function of network size. The y-axis measures the number of state transitions required to reach a preset convergence threshold for a standard diagnostic. The straight line is a regression fit to the data, which is averaged over 20 trials.

a PRM: the distributions over the values of the reference slots. Essentially, reference uncertainty is identity uncertainty over the identity mappings of the functional terms of the language. We would also like to be able to handle identity uncertainty over the identity mappings of the constant terms, or instances.

For example, we might want to remove the unique names assumption over the *Professor* class, and have uncertainty over how many professors there are in the world, as well as whether the various professors co-refer. If the language defines two *Professor* instances, $Prof_1$ and $Prof_2$, we might wonder whether ι_{Prof} is equal to $\{\{Prof_1, Prof_2\}\}$ or $\{\{Prof_1\}\{Prof_2\}\}$; as for the value of n_{Prof}, all we will know is that, if ι_{Prof} takes the first value, $n_{Prof} \geq 1$, and, otherwise, $n_{Prof} \geq 2$.

PRMs do not currently provide us with the means of defining distributions over ι_Cs and n_Cs, so we will have to extend the language and associate some appropriate definitions with each identity-uncertain class. These definitions can be expressed in any of the ways described in Chapter 3: the main difference is that, while the definitions of Chapter 3

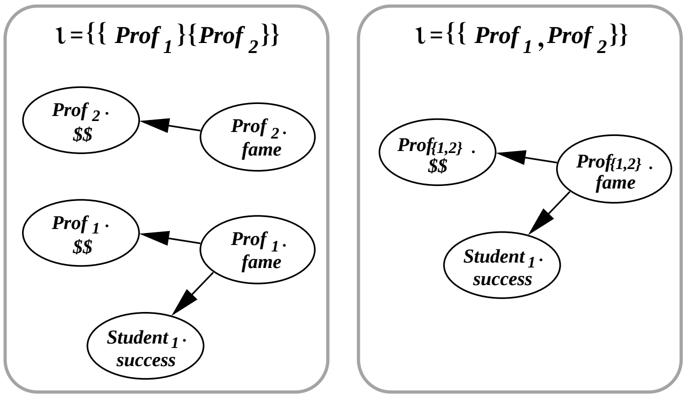


Figure 6.8. The two Bayesian network structures possible in the identity uncertain setting with two professors.

involve all of ι and n, in PRMs we will factor them into components corresponding to various classes. These components can then even depend on each other, at least as long as we avoid cycles. We might, for example, let the number of students at a university depend on the number of professors.

6.4.1 Inference

Our MCMC-based inference algorithm works in the usual way: at each time step, the chain proposes a new state a randomly-chosen element of a set of proposal mechanisms. These mechanisms include the randomized Gibbs sampler of Section 6.2.1, which, given fixed values of ι and n, runs just as before. They also include algorithms for suggesting new ι and n values, as defined in Chapter 5. Since we have shown that the algorithm of Section 6.2.1 is a reasonable mechanism for proposing changes to $\mathbf{V}$, the theorems of Chapter 5 carry over (with all the relevant conditions, of course.)

As with reference uncertainty, it is useful to note that, for any particular values of ι and n, we know exactly how the terms of the language are clustered together, and so we can express the interdependencies of all the attributes using a single Bayesian network.

For example, the distribution over the attributes of a two-professor world can always be represented using one of the two Bayesian networks of Figure 6.8. Thus, the algorithm can be thought of as moving through the space of Bayesian networks and then, for any given network, moving through the space of $\mathbf{V}$ assignments.

6.4.2 Experimental results

In the University domain, clustering two *Professor* instances is rather like merging their records, so we use a sampling-with-replacement type of distribution over the professors' identity mappings: given that there are n_{Prof} professor objects in the world, the probability that an instance refers to any particular object is $1/n_{Prof}$. We have implemented the cluster-merging proposal for ι_{Prof}. The scalability of the algorithm was tested as in Section 6.3.2, by, once more, applying it to randomly generated networks of increasing size, and measuring time to convergence using a standard diagnostic. The results shown in Figure 6.9 were obtained by constructing larger and larger sets of professors and students, with identity uncertainty over both, and requesting a posterior distribution over the number of students.

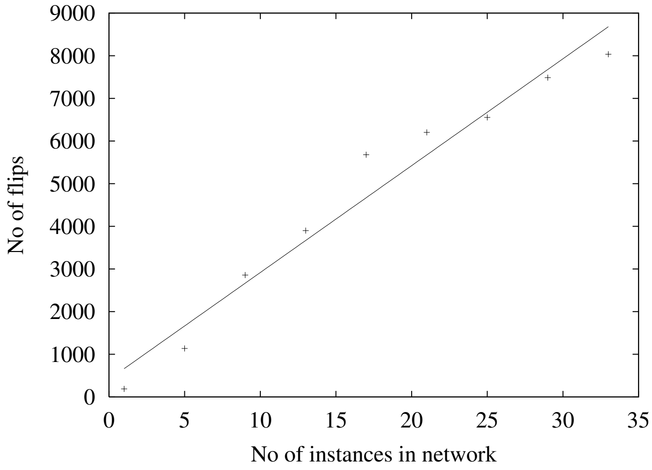


Figure 6.9. MCMC convergence as a function of network size. The y-axis measures the number of state transitions required to reach a preset convergence threshold for a standard diagnostic. The straight line is a regression fit to the data, which is averaged over 30 trials.

Chapter 7

Traffic Surveillance

The need to track multiple objects over time arises in many real-world domains, which are complicated by noisy sensors and unpredictable dynamics. The resulting problem has long been studied in the literature on **data association**. Since the problem involves reasoning about the underlying objects responsible for a set of observed objects, it is clearly an instance of the identity uncertainty problem. In this chapter, we explain how the approach we have developed in this dissertation can be applied to one data association domain, traffic surveillance.

Data association can be viewed as the problem of clustering sets of subsequent observations together to yield a set of "tracks" or object trajectories. It has long been noted (Sittler, 1964) that there are combinatorially many possible sets of trajectories, and so keeping explicit track of all the possibilities becomes infeasible as the number of objects to be tracked increases. Cox and Hingorani (1994) propose an efficient algorithm for keeping track of the m best possible hypotheses. Meanwhile, other approaches, such as that of (Bar-Shalom, 1992), abandon the idea of reasoning about full trajectories, and instead define the problem as that of associating a set of new observations with a set of previously constructed tracks, creating new tracks as needed. The problem has been tackled using several different EM-based approaches, where either the possible associations (Avitzour, 1992), or the state sequence (Logothetis, 1997) are treated as missing data. We approach the problem as an an example of identity uncertainty set in a dynamic domain, as discussed in Section 3.6: we reason about full trajectories, which are represented as aspects of possible worlds using clusters of objects, both old and new. We handle the combinatorial state space by proposing an MCMC-based algorithm.

7.1 The Application Domain

Huang and Russell (1998) provide a fairly general formulation of the vehicle tracking problem and describe an application to traffic surveillance. In this domain, the sensors are cameras or loop detectors located at various locations on a freeway network and the objects are vehicles. Many thousands of vehicles pass each camera, and the system must decide whether each vehicle is a new vehicle, or the same as one previously observed at a different location. Over time, these decisions give rise to hypothetical vehicle **trajectories**. Figure 7.1 displays a simple traffic surveillance domain, where vehicles driving down a short stretch of freeway are observed at three consecutive camera sites. It also shows three vehicle trajectories. Deriving a set of trajectories is the first step of many traffic surveillance applications, which might be interested in calculating quantities such as the average **link travel time** between locations or **origin-destination counts** along different routes through the system. Moreover, sudden changes in these quantities can be indicators of highway incidents, such as accidents or breakdowns.

Huang and Russell (1998) propose a threshold algorithm for object identification. We replace this algorithm with a polynomial-time approximation scheme based on Markov chain Monte Carlo simulation. We also replace their model, which imposed the Markov assumption on all the attributes associated with the vehicles, with one that keeps track of

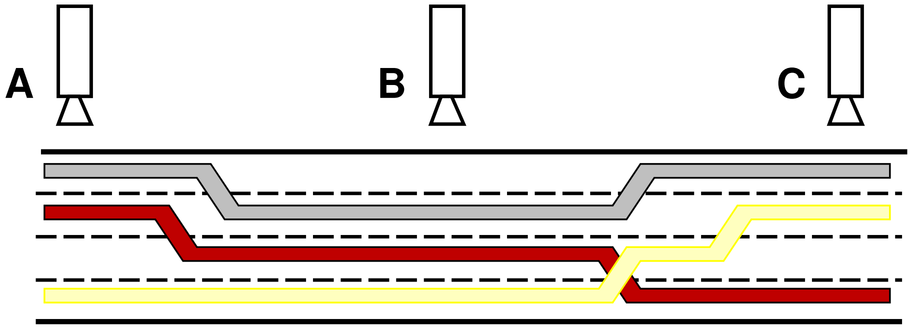


Figure 7.1. Schematic diagram showing three consecutive camera sites, A, B, and C, and three vehicle trajectories.

some of the persistent attributes (such as colour) using hidden variables. Using sensor data from a freeway traffic simulation, we show that this allows accurate estimation of long-range origin/destination information even when the individual links in the sensor chain are highly unreliable.

7.2 The Model

This domain differs differs from most other dynamic domains in that observations are made not at specific discrete time intervals, but, instead, at specific locations. While we could discretize time and attempt to estimate the vehicles' locations at every time-step, this would, in settings where the sensors are so widely dispersed that vehicles spend most of their time out of sensor range, require us to model many missed observations. This is likely to prove inefficient. Minimizing the number of missed observations by coarsening the discretization could lead to inaccuracy. Therefore, instead of indexing our observations by time, we index them by sensor name. language for the simple domain of Figure 7.1. There are two classes, *Vehicle* and *VehicleObservation*. The *Vehicle* class keeps track of the true values of the attributes of the vehicles. Some of these attributes, such as *Vehicle*

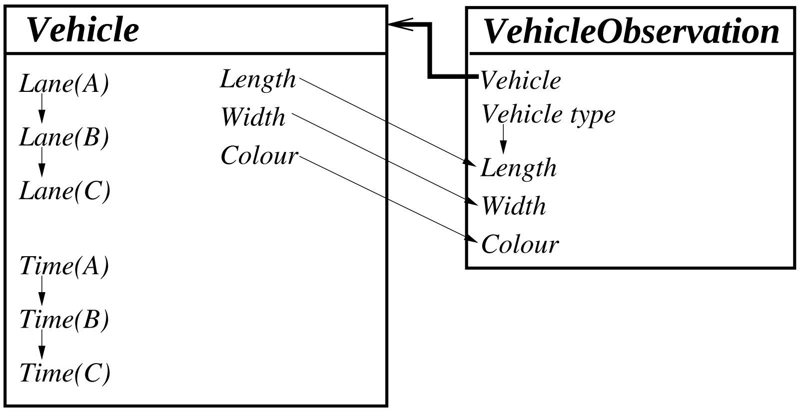


Figure 7.2. A relational schema for the traffic surveillance domain of Figure 7.1. The large rectangles represent classes: the dark arrows indicate the ranges of their reference slots, and the light arrows lay out all the probabilistic dependencies of their descriptive attributes.

type, *Colour*, *Width* and *Length*, are static; other, such as *Velocity* and *Lane* are dynamic, taking on different values at different sensors. We let observation time be one of these dynamic attributes. There is a full set of dynamic attributes associated with every sensor, and their interdependencies are based on the relative positions of the vehicle sensors. The *VehicleObservation* class keeps track of the observed values of the vehicle attributes, which are dependent on the corresponding true attributes of the relevant *vehicle*. Note that only the static attributes are included in the *VehicleObservation* class. We could add in the dynamic attributes, but we have made the assumption that they are observed accurately. We have also made the assumption that every vehicle passing a sensor is observed exactly once, so that there are no missed or duplicate observations. Every time a vehicle observation is made at a sensor s, it leads to the creation of a *VehicleObservation* instance and an associated *Vehicle* instance. All the relevant attributes are set to their observed values: these include all the *VehicleObservation* attributes, as well as the dynamic attributes associated with sensor s.

A simple possible world for this domain is the usual triple $\langle n, \delta, \mathbf{V} \rangle$, where

- n is the cardinality of the *Vehicle* class.
- δ is an identity clustering over the *Vehicle* class. (Also, δ^s represents the identity clustering over the *Vehicle*s associated with sensor s.)
- $\mathbf{V}$ is a set of arrays keeping track of all the possible applications of all the descriptive attributes.

We have made the very strong assumption that there are no extraneous objects in the world. As a result, compact possible worlds closely resemble simple possible worlds: n is a deterministic function of ι, and the only difference between the ιs and the δs is that the δs can be ordered arbitrarily, so that there are $n!$ δs corresponding to each ι. However, since we will not be calculating the actual probabilities of possible worlds, but (as usual with MCMC) only their ratios, the $n!$s will cancel, and we will not worry about them. This is not the only aspect of the probability distributions we will be ignoring in this section: as we explained in Section 3.6, our distributions over δ should be normalized to account for the fact that we know exactly how many observations were made, but we will not be calculating these normalization constants, since they cancel as well.

It is important to understand the role of the δ^ss. Since they keep track of the identity mappings of the *Vehicle* instances associated with the *VehicleObservation*s generated at sensor s, they also keep track of which *Vehicle* objects have no *Vehicle* instance associated with them, and therefore of which *Vehicle* objects have not been observed sensor s. Thus, given our assumptions that there are no missed observations, the δ^ss tell us which objects are present at each sensor. As an example, consider the three *Vehicle* instances observed at the three different sensors, $Vehicle^a_1$, $Vehicle^b_1$, and $Vehicle^c_1$. The identity mapping $\delta = \{\{Vehicle^a_1, Vehicle^b_1\}\{Vehicle^c_1\}\}$ then corresponds to the sensor-specific identity mappings $\delta^a = \{\{\mathit{Vehicle}_1^a\}\{\}\}$, $\delta^b = \{\{\mathit{Vehicle}_1^b\}\{\}\}$, and $\delta^c = \{\{\}\{\mathit{Vehicle}_1^c\}\}$, and describes the possible world where *Vehicle* object 1 drives past sensors a and b and leaves, and *Vehicle* object 2 enters before sensor c.

The distribution over simple possible worlds is defined using the factorization

$$P(n, \iota, \mathbf{V}) = P(n)P(\delta, \mathbf{V}_{\mathit{Lane}}|n)P(\mathbf{X}|\delta, n)$$

where $\mathbf{X} = \mathbf{V} - \mathbf{V}_{\mathit{Lane}}$. The factors are defined as follows. $P(n)$ is taken to be a uniform distribution, so that its contributions cancel in world probability ratios. As for $P(\mathbf{X}|\delta, n)$, the prior over *Vehicle type*, which ranges over $\{car, truck\}$, is encoded using a simple table; the remaining static attributes of the *Vehicle* class are all modeled as Gaussians, three dimensional in the case of *Colour*; their corresponding attributes of the *Vehicle Observation* class model the observation process by adding in some Gaussian noise; and the posterior distribution over the *Time* attributes is encoded as a beta distribution over the travel time between the two sensors. $P(\delta, \mathbf{V}_{\mathit{Lane}}|n)$ is more complicated: it is factored further, in a way that corresponds to the structure of the freeway network. For example, the network of Figure 7.1 leads to the factorization

$$\begin{aligned} P(\delta, \mathbf{V}_{\mathit{Lane}}|n) &= P(\mathbf{V}_{\mathit{LaneC}}|\mathbf{V}_{\mathit{LaneB}}, \delta^C, \delta^B, n)P(\delta^C|\delta^B, \mathbf{V}_{\mathit{LaneB}}, n) \\ &\quad P(\mathbf{V}_{\mathit{LaneB}}|\mathbf{V}_{\mathit{LaneA}}, \delta^B, \delta^A, n)P(\delta^B|\delta^A, \mathbf{V}_{\mathit{LaneA}}, n) \\ &\quad P(\mathbf{V}_{\mathit{LaneA}}|\delta^A, n)P(\delta^A|\mathbf{W}, n)\,. \end{aligned} \tag{7.1}$$

By intertwining the distributions over the δ^ss and the $\mathbf{V}_{\mathit{LaneS}}$ in this way, we are trying to capture the strong correlation between the lane a vehicle travels in and the probability that it is about to exit or has just entered. The *LaneS* attribute associated with each sensor S ranges over the integers $[1 \ldots l]$, where l is the number of lanes, as well as a special token representing that fact that the vehicle was not observed at S. Factors like $P(\mathbf{V}_{\mathit{LaneC}}|\mathbf{V}_{\mathit{LaneB}}, \delta^C, \delta^B, n)$ capture the lane-changing probabilities for vehicles that are

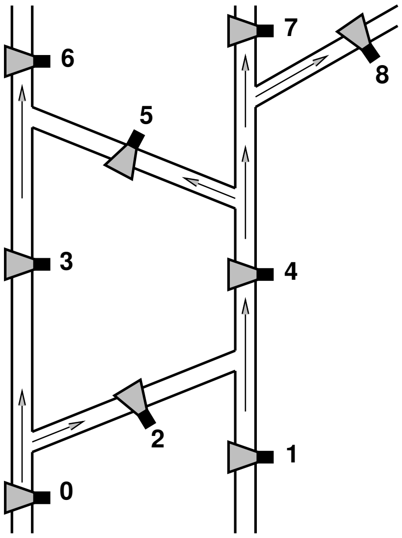


Figure 7.3. Schematic diagram of a simulated freeway network with nine cameras. We will be trying to approximate the origin-destination counts between the two entry points and the three exit points.

observed at both sensors, the prior over lanes for vehicles that have just entered, and also the deterministic constraint that the lane of vehicles that have just exit be set to the special token. Both the prior and the lane-changing posterior distributions are encoded using simple tables. Factors like $P(\delta^C|\delta^B, \mathbf{V}_{LaneB}, n)$ represent the probabilities of staying in the system, entering the system, or leaving it, as we discussed in Section 3.6.

Equation 7.1 is very simple. A more complicated, branching freeway network would lead to a more complicated factorization. Let us define a **section** as a contiguous fragment of the freeway network, demarcated by sensors. These sensors can be divided into two sets, an upstream set which observes the vehicles entering the section, and a downstream set which observes the vehicles leaving it. The movement of vehicles across each section is modeled using a distribution over the δs of the downstream sensors given the δs of the upstream sensors; the lane priors and lane-changing distributions are modeled using a distribution over the $\mathbf{V}_{Lane}$ values of the downstream sensors given the $\mathbf{V}_{Lane}$ values of the upstream sensors, as well as all the relevant δs. For example, the $P(\delta|\mathbf{W}, n)$ of the network presented in Figure 7.3 would be factored as

$$\begin{aligned} P(\delta|\mathbf{W}, n) \quad = \quad & P(\mathbf{V}_{Lane8}, \mathbf{V}_{Lane7}, \mathbf{V}_{Lane5}|\delta^8, \delta^7, \delta^5, \delta^4, \mathbf{V}_{Lane4}, n)P(\delta^8, \delta^7, \delta^5|\delta^4, \mathbf{V}_{Lane4}, n) \\ & P(\mathbf{V}_{Lane6}|\delta^6, \delta^5, \delta^3, \mathbf{V}_{Lane5}, \mathbf{V}_{Lane3}, n)P(\delta^6|\delta^5, \delta^3, \mathbf{V}_{Lane5}, \mathbf{V}_{Lane3}, n) \\ & P(\mathbf{V}_{Lane4}|\delta^4|\delta^1, \delta^2, \mathbf{V}_{Lane1}, \mathbf{V}_{Lane2}, n)P(\delta^4|\delta^1, \delta^2, \mathbf{V}_{Lane1}, \mathbf{V}_{Lane2}, n) \\ & P(\mathbf{V}_{Lane3}, \mathbf{V}_{Lane2}|\delta^3, \delta^2, \delta^0, \mathbf{V}_{Lane0}, n)P(\delta^3, \delta^2|\delta^0, \mathbf{V}_{Lane0}, n) \\ & P(\mathbf{V}_{Lane1}|\delta^1, n)P(\delta^1|n)P(\mathbf{V}_{Lane0}|\delta^0, n)P(\delta^0|n)\ . \end{aligned}$$

The first line corresponds to the section between cameras 4, 5, 7, and 8. Let us describe the part of the distribution over δ associated with that term more closely. We assume here that each object is observed at each sensor it passes once and only once. This implies that each trajectory can contain only one of the objects associated with a sensor. As in Section 3.6, we assume that, apart from this constraint, the trajectories are independent of each other, and so each of the terms can be factored further, into distributions associated with the particular trajectories. Clearly, we will have to model several possible distributions corresponding to all the possible sets of parent values. We will need one for each of the sensors that the object might have been at previously; in addition, if we permit objects to leave and enter the network in between sensors, we will also need a distribution to account for the possibility that the object has just entered. For example, to handle the second component, we would need to define the following distributions:

$P(\delta_i^6|\delta_i^5 = \{\}, \delta_i^3 = \{T\}, \mathbf{V}_{Lane3}, n)$

$P(\delta_i^6|\delta_i^5 = \{T\}, \delta^3 = \{\}, \mathbf{V}_{Lane5}, n)$

$P(\delta_i^6|\delta_i^5 = \{\}, \delta^3 = \{\}, \mathbf{W}, n)$

We do not have to define anything along the lines of

$P(\delta_i^6 = \{T\}|\delta_i^5 = \{T\}, \delta^3 = \{T\}, \mathbf{V}_{Lane}, n)$,

since that particular set of parents is impossible.

In each of the defined distributions, δ_i^6 can range over $\{T\}$ and $\{\}$, because any object T entering the section must either leave the network entirely or continue past sensor 6. In components where more than one sensor monitors the objects leaving, the distributions would range over all the possible ways in which T could be assigned to one (or none) of those sensors.

7.3 Algorithm

Our approximate, MCMC-based inference algorithm is built around the multipartite matching proposal of Section 5.3.2. One big difference is that, instead of considering bipartite matchings between pairs of disjoint sets, each corresponding to a single sensor, we consider bipartite matchings between the set of objects observed entering one of the section and the set of objects observed leaving it. As we apply the algorithm in an on-line setting, where new observations are continuously arriving, we handle the infinite stream of data by keeping around only a fixed number of observations. This is important not only because we have only finite storage space, but also because, while our models are assumed dynamic, we do not model their changes explicitly: instead, we use the latest data to continuously update the models using stochastic on-line EM. Thus, we hope that the changes in the models happen slowly enough that the distributions behind the set of observations we are keeping around can all be approximated using the same set of models.

The dynamic variable models at each link and the intrinsic variable models at each sensor are all learnt using stochastic EM, as shown in Figure 7.4. A model update is performed whenever a new object is observed at a sensor, and the MCMC process has been given time to converge. Each EM iteration proceeds as follows.

7.4 Learning the Models

```
while models have not converged do
  E: Run MCMC, sampling ι.  Calculate U.
  M: Update models using the sampled values
end while
```

During the approximate E step, we obtain samples of hidden variables ι, the identity mapping, and $\mathbf{U}$, the set of unobserved attribute values. Samples of ι are generated as the MCMC is run. The attribute values in $\mathbf{U}$ are not a part of our Markov chain state, so we must complete each sample by calculating $E(V_i|\iota, \mathbf{V} - V_i)$ for every unobserved attribute V_i. This involves integrating over a product of attribute values, attribute noise models, and the attribute prior. The fact that all of the relevant models are Gaussian lets us derive a relatively straightforward calculation which can be computed in time proportional to trajectory length. To give a simplified example, let's take a sequence of vehicle length measurements, and let's assume a uniform prior and a sequence of Gaussian noise models with a common variance. In this situation, the expected true vehicle length is just the mean of all the observations. Of course, the computation may become much more complicated if other models are used. In that case, sampling $\mathbf{U}$ as well as ι may be the best approach.

The M step is exact. It uses the ι and $\mathbf{U}$ values of each sample as if they were observed variables to perform conventional parameter learning.

We run EM whenever a new observation is made, using the old models as a starting point (in this way, information conveyed by discarded observations is not immediately lost.) Running EM to its convergence requires a Markov chain run for each EM iteration, resulting in many Markov chain runs. In practice, however, the chain converges very quickly once the algorithm is running, as the models change only slowly with each incoming observation. Moreover, Figure 7.4 demonstrates that the algorithm can some-

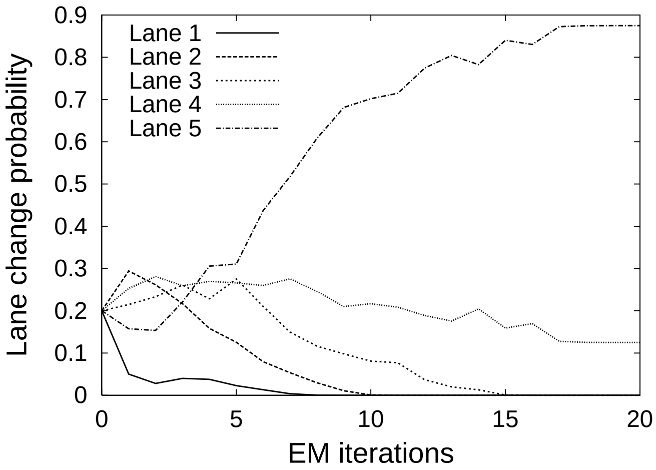


Figure 7.4. The distribution over the lane at the downstream given that the vehicle was in lane five at the upstream sensor, at various iterations of the EM algorithm. The observations used came from two loop sensors located a few miles apart in the Los Angeles freeway system; thus, no colour information was collected. The EM algorithm was still able to learn what seems to be a reasonable distribution: vehicles seem to stay in lane five with high probability, and, if they do change lanes, are most likely to end up in the adjacent lane, lane four.

times learn reasonable parameters quite rapidly even when starting off with an arbitrary distribution.

7.5 Results

We have performed two experiments comparing our approach to that of Huang and Russell (1998). Our data sets were created using a freeway simulator which allowed us to control road configurations, camera characteristics, and complex vehicle behaviour. Each set included the observations generated by approximately one hundred cars, and the models used in the experiments had been learnt by our EM-MCMC algorithm using data generated with the same parameter settings.

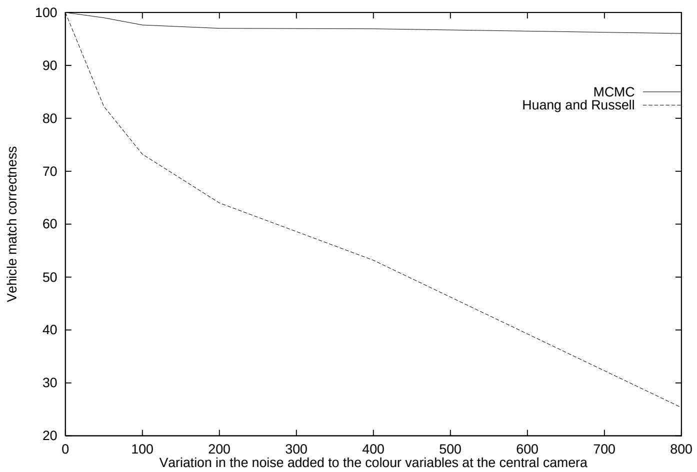


Figure 7.5. The vehicle matching accuracy of two algorithms as a function of the distortion at the central sensor. In the case of the Huang and Russell algorithm, which results in discrete matches, the accuracy is simply the percentage of correct A to C matches: in the case of the MCMC, it is the average percentage of correct matches across all the samples drawn.

7.5.1 Overcoming faulty sensors

In this experiment, both algorithms were applied to a three-camera network such as that in Figure 7.1. The outer two cameras, A and C, were completely reliable to the point where they could distinguish all cars, while the central camera, B, was subjected to increasing degrees of noise (The variance of the Gaussian error added to the colour information was increased.) Thus, the data set contained enough data for a complete matching between cameras A and C, but a pairwise appearance-based model, such as that of Huang and Russell (1998), could be expected to lose accuracy due to its hasty independence assumptions.

The results of the experiment, averaged over three runs, can be seen in Figure 7.5. The advantage of MCMC is clear: while the pairwise approach is highly sensitive to sensor

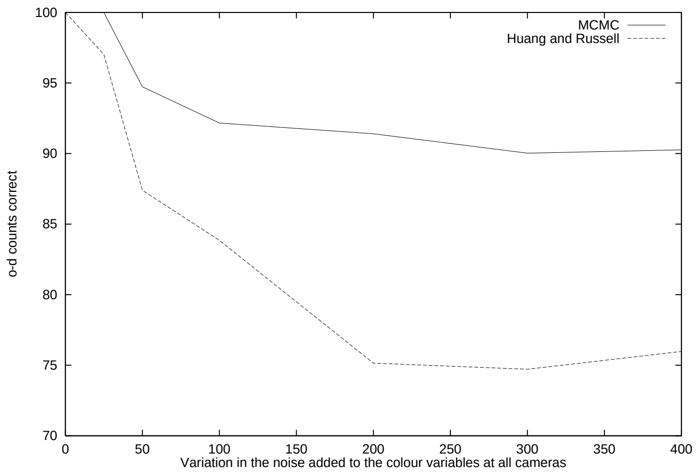


Figure 7.6. The origin-destination count accuracy of the two algorithms as functions of the distortion of multiple sensors. In the MCMC case, the quantity is averaged over all the samples.

noise, feature estimation helps to maintain an almost constant degree of accuracy.

7.5.2 Estimating origin/destination counts

We also compared the algorithms in a slightly more complicated setting, that of the larger network of Figure 7.3. The aim was to estimate the origin-destination counts between the two entry points and the three exit points. The color variance was manipulated as before, but at all eight cameras simultaneously. As we attempt to track vehicles through this system, the greater number of sensors should be a liability to the pair-wise model, as each pair of neighbouring sensors can make its own mistakes independently of the others. The MCMC-based algorithm, on the other hand, should benefit from the ability to observe each object several times to estimate intrinsics more reliably.

Figure 7.6 shows the percentage accuracy of the origin-destination counts as a function of the colour noise variance. Again, the performance deteriorates as the distortion increases; however, as expected, the MCMC-based approach again substantially outper-

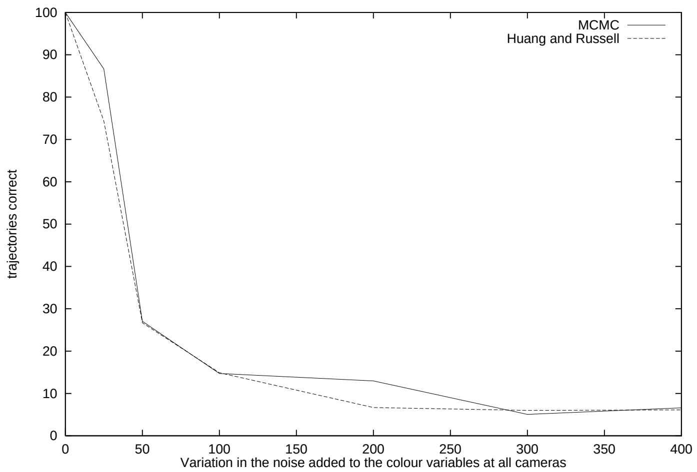


Figure 7.7. The origin-destination vehicle matching accuracy of the two algorithms as functions of the distortion of multiple sensors. Again, in the MCMC case, the quantity is averaged over all the samples.

forms the Huang-Russell algorithm, holding up even for levels of noise that essentially wipe out the colour altogether. On the other hand, Figure 7.7 shows that both methods are unable to find exact matches accurately for high levels of noise. The ability of the MCMC algorithm to recover reasonable counts despite the failure of individual matches suggests that its samples contain a reasonable amount of information about the ensemble behaviour of the vehicles: it seems quite probable that many of the incorrect matches involve swaps among similar vehicles traveling close together.

Chapter 8

Citeseer

Citation matching is the problem currently handled by systems such as Citeseer (Lawrence, Bollacker, & Giles, 1999).[1] Such systems process a large number of scientific publications to extract their citation lists. By grouping together all co-referring citations (and, if possible, linking to the actual cited paper), the system constructs a database of "paper" entities linked by the "$cites(p_1, p_2)$" relation. This is an example of the general problem of determining the existence of a set of objects, and their properties and relations, given a collection of "raw" perceptual data; this problem is faced by intelligence analysts and intelligent agents as well as by citation systems. A key aspect of this problem is determining when two observations describe the same object; only then can evidence be combined to develop a more complete description of the object. This is clearly a question of identity uncertainty.

Citations appear in many formats and are rife with errors of all kinds. Figure 8.1 shows two citations that probably refer to the same paper, despite many superficial differences. As a result, Citeseer—which is specifically designed to overcome such problems—currently lists more than 100 distinct AI textbooks published by Russell and Norvig on or

[1] See `citeseer.nj.nec.com`. Citeseer is now known as ResearchIndex.

```
[Lashkari et al 94] Collaborative Interface Agents, Yezdi
Lashkari, Max Metral, and Pattie Maes, Proceedings of the Twelfth
National Conference on Articial Intelligence, MIT Press, Cam-
bridge, MA, 1994.

Metral M. Lashkari, Y. and P. Maes.  Collaborative interface
agents.  In Conference of the American Association for Artificial
Intelligence, Seattle, WA, August 1994.
```

Figure 8.1. Two citations that probably refer to the same paper.

around 1995, from roughly 1000 citations.

This particular type of identity-uncertain domain has been approached in several different ways. *Record linkage* (Fellegi & Sunter, 1969) is a method for matching up the records in two files, as might be required when merging two databases. For each pair of records, a *comparison vector* is computed that encodes the ways in which the records do and do not match up. The probabilistic model consists of two conditional distributions over comparison vectors, one for matched pairs, and one for unmatched pairs. These distributions can be expressed as naive-Bayes distributions, and learnt using EM (Winkler, 1999). Linkage decisions are typically made by comparing the ratio of the two probabilities to some threshold values. The automatic acceptance of all matches whose ratios exceed the threshold can lead to inconsistencies (such as deciding that (A, B), (A, D), and (C, D) are matched pairs, but (B, D) is not.) Jaro (1984) proposed a method for overcoming this problem by searching for the single best one-to-one mapping; in the more general many-to-many case, matching decisions are often made in greedy fashion, which makes the overall process order-dependent and can lead to non-optimal global matchings. Overall, the approach does not provide for a general, principled way to combine matched records.

A richer probability model is developed by Cohen, Kautz, and McAllester (2000), who model the database as a combination of some "original" records that are correct and some number of erroneous versions. They give an efficient greedy algorithm for finding a single locally optimal assignment of records into groups. Citeseer is the best-known example of work on citation matching (Lawrence et al., 1999). The system groups citations using a form of greedy agglomerative clustering based on a text similarity metric. McCallum, Nigam, and Ungar (2000) use a similar technique, but also develop clustering algorithms designed to work well with large numbers of small clusters.

All of these preceding systems have used domain-specific algorithms and data structures to handle identity uncertainty over one set of objects. We use the citation matching domain as the first substantial application of our declarative, PRM-based representation and MCMC-based inference method, and include a method for generating effective proposals based on ideas from (McCallum et al., 2000). The system also incorporates an EM algorithm for learning the local probability models, such as the model of how author names are abbreviated, reordered, and misspelt in citations. Its performance is evaluated the on four datasets originally used to test the Citeseer algorithms (Lawrence et al., 1999). As well as providing significantly better performance, our system is able to reason simultaneously about papers, authors, titles, and, to some degree, publication types, and does a good job of extracting this information from the grouped citations. For example, an author's name can be identified more accurately by combining information from multiple citations of several different papers. The errors made by our system point to some interesting unmodeled aspects of the citation process.

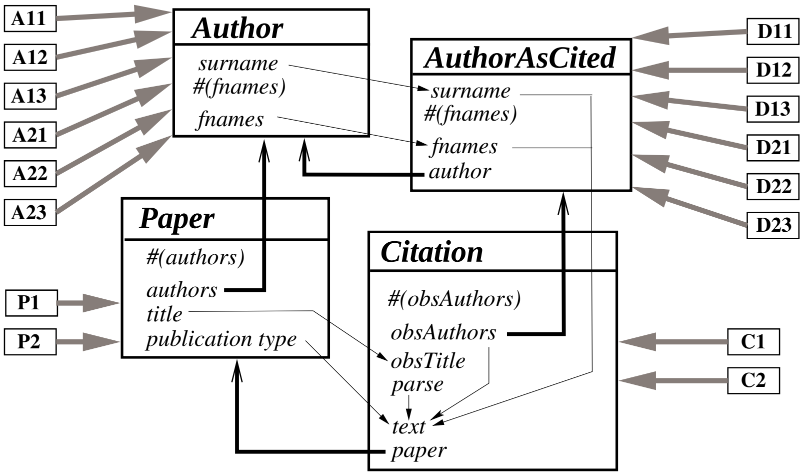


Figure 8.2. A relational schema for the example citations of Figure 8.1. The large rectangles represent classes: the dark arrows indicate the ranges of their reference slots, and the light arrows lay out all the probabilistic dependencies of their descriptive attributes. We have also included small rectangles representing instances, linked to their classes with thick grey arrows. In addition to these instances, our skeleton contains statements setting most of the reference slots.

8.1 The Language

Figure 8.2 presents a relational schema describing the example citations of Figure 8.1. There are four classes, the self-explanatory *Author*, *Paper*, and *Citation*, as well as *AuthorAsCited*, which represents not actual authors, but author names as they appear when cited. Each citation we wish to match leads to the creation of a *Citation* instance; instances of the remaining three classes are then added as needed to fill all the reference slots. As an example, consider the the first citation in Figure 8.1. To represent it, we would create a *Citation* instance C_1 and set its *text* attribute to the string "`Metral M. ...August 1994.`". We would then set its *paper* attribute to a newly created *Paper* instance, which we will call P_1, and introduce $\max(\#[author])$ (here only 3, for simplicity) *AuthorAsCited* instances

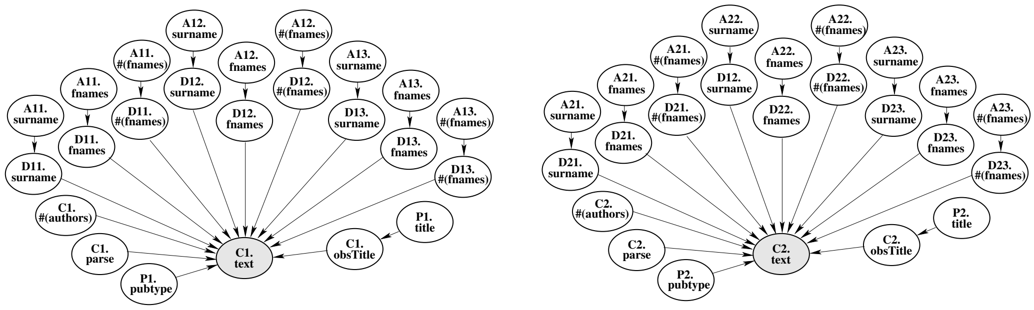


Figure 8.3. A Bayesian network modeling the dependencies encoded by the relational schema of Figure 8.2, given that $P_1 \neq P_2$.

(D_{11}, D_{12}, and D_{13}) to fill the P_1.*obsAuthors* (i.e., observed authors) attribute, as well as an equal number of *Author* instances (A_{11}, A_{12}, and A_{13}) to fill both the P_1.*authors[i]* and the D_{1i}.*author* attributes. The values of most of the reference slots are thus fixed, using equality statements placed in the evidence. The only uncertain reference slot is the *author* slot of the *AuthorAsCited* class, which ranges over all the relevant *Author* instances in order to model the possibility that the authors are out of order. Assuming (for now) that the value of C_1.*parse* is observed, we can set the values of all the descriptive attributes of the *Citation* and *AuthorAsCited* instances. Here, given the correct parse, D_{11}.*surname* would be set to `Lashkari`, and D_{12}.*fnames* would be set to $\{$`Max`$\}$. The remaining attributes — those of the *Paper* and *Author* instances — represent the "true" attributes of those objects, and their values are unobserved. One possible Bayesian network modeling the dependencies encoded by the schema of Figure 8.2 is shown in Figure 8.3.

8.2 Possible Worlds

In our application, any two citations may or may not be describing the same paper. This is modeled as identity uncertainty over the *Paper* class. Thus, for citations C_1 and C_2, we have uncertainty as to whether the corresponding papers P_1 and P_2 are distinct, or whether they refer to the same object. If they are the same, then, clearly, their respective lists of actual authors, $\mathbf{A}_1$ and $\mathbf{A}_2$ must also refer to the same list of objects. We also have identity uncertainty over the *Author* class. Authors associated with different papers can co-refer, but they can also be distinct, even if they have identical names. All the *Citation* and *AuthorAsCited* instances are constrained to be distinct. Since there is no identity uncertainty over those two classes, we ignore them when we define a possible world as a tuple $\langle n, \iota, \mathbf{V}\rangle$, where

- $\iota = \{\iota_{Paper}, \iota_{Author}\}$, where ι_C is an identity clustering over instances of the class C, represented as a set partition of the set of observed C instances.

- $n = \{n_{Paper}, n_{Author}\}$, where n_C is the cardinality of the class C, represented as some number $\geq |\iota_C|$.

- $\mathbf{V}$ is the set of attributes of the observed objects—all the objects referred to by at least one observed instance. It is true that when $n_C > |\iota_C|$ (where C is one of the two identity-uncertain classes), the world contains objects unreferenced by any of the terms; however, as we have discussed in Section 5.2.1, we do not need to worry about the attributes of unreferenced objects in domains where term-to-object mappings are not directly dependent on object attributes. Intuitively, the filler objects are irrelevant because there is no reference slot in the model whose value depends on the attributes of filler objects.

When defining the probability of a possible world, $P(\iota_{Paper}, \iota_{Author}, n_{Paper}, n_{Author}, \mathbf{V})$, we factor it as follows

$$P(\mathbf{V}|\iota_{Paper}, \iota_{Author})P(\iota_{Paper}|n_{Paper})P(\iota_{Author}|n_{Author})P(n_{Paper})P(n_{Author})$$

The distribution over $\mathbf{V}$ is provided by the PRM, given the ιs which define the relational structure. The $P(n_C)$s are specified using a high-variance log-normal distribution. As for the distributions over the ι_Cs, we have decided that each paper is *a priori* equally likely to be cited, and that each author is *a priori* equally likely to write a paper. Since, here, "*a priori*" means prior to obtaining *any* information about the object in question, the uniformity assumption is entirely reasonable. In all, we view instances of each of these classes as drawn from a uniform distribution over all the n_C objects of the relevant class, so that the probability of any particular mapping from k_C instances to n_C objects is $1/n_C^{k_C}$. If we use Theorem 3.4.1 to convert from mappings to identity clusterings, we get

$$P(\iota_C|n_C) = \frac{n_C!}{(n_C - |\iota_C|)!}\frac{1}{{n_C}^{k_C}} \; .$$

We could, of course, define a more complex model. For example, it would be reasonable to let the total number of papers depend on the total number of authors. Similarly, we could allow ι to depend on some of the attribute values—for example the frequency of citations to a given paper might depend on the topic of the paper, or on the fame of the authors.

8.3 The Probability Model

We will now fill in the details of the conditional probability models, which are constructed partly off-line and partly using stochastic EM, and draw on the following resources: the 1990 Census data on US names, a large A.I. BibTeX bibliography, and a hand-parsed collection of 500 citations.

First, let us consider *Citation.parse*, which cannot be an observed variable, since citation parsing, and even citation subfield extraction, is an unsolved problem. It is therefore fortunate that our approach lets us handle uncertainty over parses quite naturally. The state space of *Citation.parse* has two different components. First of all, it keeps track of the *citation style*, defined as the ordering of the author and title subfields (represented as a Boolean variable encoding whether the authors come first), as well as the format in which the author names are written (represented using three Boolean variables, encoding the surname/first-name ordering of the first author, the surname/first-name ordering of the remaining authors, and the style in which first names are written.) Secondly, it keeps track of our segmentation of *Citation.text*, which is divided into an author segment, a title segment, and three filler segments (one before, one after, and one in between the two meaningful segments.) The prior over styles is learned off-line, using our hand-segmented file. We assume a uniform distribution over segmentations.

One possible (and non-optimal) segmentation of the second sentence in Figure 8.1 might divide it up as follows:

- *Filler 1*: `Metral M.`
- *Authors*: `Lashkari, Y. and P. Maes.`
- *Filler 2*: `Collaborative interface agents.`
- *Title*: `In Conference of the American Association for Artificial Intelligence,`
- *Filler 3*: `Seattle, WA, August 1994.`

In this case, the citation style would be described as follows: authors precede titles, the first author's surname precedes the first name, the remaining author's surname follows the first name, and first names are written as initials.

Citation.parse greatly constrains *Citation.text*: the title segment of *Citation.text* must match the value of *Citation.obsTitle*, while the names found in its author segment must match the names of *Citation.obsAuthors*. The way in which these names are arranged is modeled using a set of context-free probabilistic grammars (one for each author name format) which are learnt off-line using the A.I. bibliography. The contents of the *Fillers* are modeled using hierarchical bigram models. Each of these models combines a letter-based model of words (a linear combination of a bigram model, a unigram model, and a dictionary) and a word-based model of the text segment itself (a linear combination of a bigram model and a unigram mode.) There are seven such models: one for *Filler 1*, one for *Filler 2*, and five for *Filler 3*, corresponding to the five possible publication types. All these models were, once again, learnt off-line using our pre-segmented file. They capture information such as the fact that the *Filler 2* segment tends to be short, and that, if the paper is an article, the *Filler 3* segment is very likely to contain the word `Journal`. Our incorrect segmentation, which has an unusually lengthy *Filler 2* and which places `Conference` in the *Title* rather that in *Filler 3*, does not score very highly according to these models.

Let us now turn to our "true" attribute priors, which are also learnt off-line. True titles are modeled exactly like *Filler*s, using a hierarchical bigram model constructed using the A.I. bibliography. The values of *Author.fnames* and *Author.surname* are modeled as having a a 0.9 chance of being drawn from the relevant US census file, and a 0.1 chance of being generated using a bigram model learned from that file. The distributions over *Author.#(fnames)*, *Paper.#(authors)*, and *Paper.pubType* [2] are derived from our hand-parsed file through simple counting.

Finally, there are the noise models. The conditional distributions of the "observed" variables given their true values (i.e., the corruption models of *Citation.obsTitle*, *AuthorAsCited.surname*, and *AuthorAsCited.fnames*) are modeled as noisy channels where each letter, or word, has a small probability of being deleted, or, alternatively, changed, and there is also a small probability of insertion. *AuthorAsCited.fnames* may also be abbreviated as an initial. The parameters of the corruption models are learnt on-line, using

[2] Publication types range over {article, conference paper, book, thesis, and tech report}

stochastic EM.

8.4 Inference

We use an MCMC inference algorithm of the type discussed in Chapter 5: one which, at each time-step, randomly chooses one of three proposal mechanisms, and then uses the usual acceptance ratio to decide whether to accept the newly proposed state. Of the three mechanisms, one proposes changes to the ns, one proposes changes to the ιs as well as new attribute values for the identity-uncertain objects, and one proposes changes to *parse*, one of the two unobserved attributes *not* associated with an identity-uncertain object. The *author* attribute of *AuthorAsCited* is the other, but this attribute is not included in the Markov chain state space. The set of *Authors* it ranges over is quite small, and so we sum it out. We also sum out *publication type*, another unobserved attribute with a small range. This works as follows: when calculating the probability of a *Filler 3* string, we take it to be the weighted sum of the five probabilities calculated with the bigram models associated with the five publication types.

Together, the three mechanisms propose changes to all the relevant aspects of the possible worlds. Here are a few more details on each mechanism:

- The n proposal uses a random walk.
- To construct the parse proposal, we pre-process each citation, organizing its plausible segmentations into a list ordered in terms of descending probability. At run-time, we simply sample from these discrete distributions. Since we assume that boundaries occur only at punctuation marks, and discard segmentations of probability $< 10^{-6}$, the lists are usually quite short.[3]

[3] It would also be possible to sample directly from a model such as a hierarchical HMM.

- Changes to ι_C are proposed using the cluster-mixing algorithm described in Section 5.3.1. Given a proposed ι'_C, suggesting values for the hidden attributes boils down to recovering their true values from (possibly) corrupt observations, e.g., guessing the true surname of the author currently known both as "Simth" and "Smith". Since our title and name noise models are symmetric, our basic strategy is to apply these noise models to one of the observed values. In the case of surnames, we have the additional resource of a dictionary of common names, so, some of the time, we instead pick one of the set of dictionary entries that are within a few corruptions of our observed names. (One must, of course, careful to account for this hybrid approach in our acceptance probability calculations.)

8.4.1 Scaling Up

One of the acknowledged flaws of the MCMC algorithm is that it often fails to scale. In this application, as the number of papers increases, the simplest approach — one where the two clusters to be mixed are picked u.a.r — is likely to lead to many rejected proposals, as most pairs of clusters will have little in common. The resulting Markov chain will mix slowly. Clearly, we would prefer to focus our proposals on those pairs of clusters which are actually likely to exchange their instances. We have implemented an approach based on the efficient clustering algorithm of McCallum et al (McCallum et al., 2000), where a cheap distance metric is used to pre-process a large dataset and fragment it into many *canopies*, or smaller, overlapping sets of elements that have a non-zero probability of matching. We do the same, using word-matching as our metric[4]. Then, at runtime, our $q(x'|x)$ function proposes first a canopy c, and then a pair of clusters u.a.r. from c. ($q(x|x')$ is calculated by summing over all the canopies which contain any of the elements of the

[4]The two thresholds required during canopy construction are set to 0.5 and 0.2.

	Face 349 citations, 242 papers	Reinforcement 406 citations, 148 papers	Reasoning 514 citations, 296 papers	Constraint 295 citations, 199 papers
Phrase matching	*94%*	*79%*	*86%*	*89%*
RPM + MCMC	*97%*	*94%*	*96%*	*93%*

Table 8.1. Results on four Citeseer data sets, for the text matching and MCMC algorithms. The metric used is the percentage of actual citation clusters recovered perfectly; for the MCMC-based algorithm, this is an average over all the MCMC-generated samples.

two clusters.)

8.5 Experimental Results

We have applied the MCMC-based algorithm to the hand-matched datasets used by Lawrence et al. (1999). Each of these datasets contains several hundred citations of machine learning papers, about half of them in clusters ranging in size from two to twenty-one citations. We have also implemented their *phrase matching* algorithm, a greedy agglomerative clustering method based on a metric that measures the degrees to which the words and phrases of any two citations overlap. (They obtain their "phrases" by segmenting each citation at all punctuation marks, and then taking all the bigrams of all the segments longer than two words.) The results of our comparison are displayed in Figure 8.1, in terms of the Citeseer error metric. Clearly, the algorithm we have developed easily beats our implementation of phrase matching.

We have also applied our algorithm to a large set of citations referring to the textbook *Artificial Intelligence: A Modern Approach.* It clusters most of them correctly, but there are a couple of notable exceptions. Whenever several citations share the same set of unlikely errors, they are placed together in a separate cluster. This occurs because we do not currently model the fact that erroneous citations are often copied from reference list to reference list, which could be handled by extending the model to include a *copiedFrom* attribute. Another possible extension would be the addition of a *topic* attribute to both papers and authors: tracking the authors' research topics might enable the system to distinguish between similarly-named authors working in different fields.

Chapter 9

Conclusion

In this dissertation, we have presented a framework for reasoning about identity uncertainty in first-order probabilistic languages. We began by outlining our general approach. We described several ways of extending such languages to enable them to represent identity uncertainty by encoding distributions over the identity mappings of the terms; here, we attempted to ensure that our approach is of practical use by considering different ways in which identity uncertainty can occur in real-world settings, such as record merging, feature correspondence, and tracking. We then formalized two representations for the identity-uncertain possible worlds themselves, and used them to show that our extended languages do indeed define a consistent and unique representation over these worlds. At that point, we turned to the problem of conditional inference. Since exact inference is, in general, intractable in this setting, we presented a family of MCMC-based algorithms, which are capable of approximating distributions over various aspects of the possible worlds by drawing samples from the space of these worlds. We described how such algorithms might be constructed in different identity-uncertain domains, and, in each case, proved that the algorithm is guaranteed to approximate the right distribution when certain conditions are met. We even provided running-time bounds for Markov chains over

Bayesian networks and weighed bipartite matches, two types of chains that can be used as subcomponents of our more general algorithms. In this way, we showed that our family of algorithms is well-grounded in MCMC theory. Finally, we applied our algorithms to several domains, including two toy domains, that of balls and urns and that of students and their advisors, and two more realistic domains, traffic surveillance and citation matching. The results were promising: in both cases, our approach was more accurate than previously applied methods. In this way, we have demonstrated that, as we stated in our thesis, our MCMC methods are a feasible approach to reasoning about identity uncertainty in first-order probabilistic settings.

9.1 Future Work

The work presented in this dissertation might be extended in a variety of ways. Here, we list some of the more interesting possibilities.

- One rather general extension we could make would be to develop the identity-uncertain approaches presented in Chapter 3 further, enabling them to handle more complicated settings and dependency structures. We might, for example, extend the noisy observation scenario to allow for missing observations, or let the number of objects in the world depend on other aspects of the world.

- We could work on making the MCMC-based algorithms faster and more efficient. One obvious way of accomplishing this would be to develop better proposal distributions, ones that connect different parts of the state space effectively, and also lead to optimal acceptance ratios. Clever proposals can be designed off-line (the canopy-based method we used in the Citeseer domain is an example of this), or they can be learnt adaptively while the algorithm runs. Other possible methods for speed-ing up the algorithm might include picking the initial starting state intelligently, and running several chains in parallel.

- As we improve the speed of the MCMC algorithm, we should also keep an eye on MCMC theory: any chain we design should come with a theoretical guarantee. We could also work on developing tighter running-time bounds on existing chains.

- A further possibility would be to work on approximate inference algorithms that do not involve MCMC. While we have shown that the MCMC-based approaches, which approximate a weighed sum over the possible identity assignments, work well when the world is full of noise, so that there exist many probable possible worlds, in settings where one possible world is the clear winner we might want to simply search for this one world.

- In this dissertation, we have downplayed one rather important problem, the problem of learning the identity-uncertain aspects of our representations. This problem becomes particularly interesting when the identity mappings in a world are dependent on other aspects of the world, so that we have to learn such probabilistic dependencies as well as distribution parameters. We are currently working on an approach that draws inspiration from PRM structure learning.

- There is no better way of testing and refining both representations and inference algorithms than applying them to a variety of new domains. Since we believe that identity uncertainty is a pervasive problem in many interesting, complex domains, it should not be difficult to find appropriate problem settings. Currently, we are attempting to apply our approach to the problem of extracting information from meeting-scheduling emails.

Generally speaking, it is our hope that the approach presented here will be useful not only as a formal background for our family of algorithms, but also as a general robust method for formulating the identity uncertainty problem regardless of the algorithm used. We do expect that, once a feasible learning algorithm has been developed, relational probabilistic languages with identity uncertainty will be a very useful tool for extracting knowledge from raw data gathered in domains where objects have no clear unique identifiers.

Bibliography

Avitzour, D. (1992). A maximum likelihood approach to data association. *IEEE Aerospace and Electronic Systems*.

Bar-Shalom, Y. (Ed.). (1992). *Multitarget multisensor tracking: Advanced applications*. Artech House, Norwood, Massachusetts.

Cohen, W., Kautz, H., & McAllester, D. (2000). Hardening soft information sources. In *KDD-2000*.

Cox, I. J., & Hingorani, S. (1994). An efficient implementation and evaluation of Reid's multiple hypothesis tracking algorithm for visual tracking. In *IAPR-94*.

Dagum, P., & Luby, M. (1993). Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, *60*, 141–153.

Dellaert, F. (2001). *Monte Carlo EM for Data-Association and its Applications in Computer Vision*. Ph.D. thesis, Carnegie-Mellon University.

Enderton, H. B. (1972). *A Mathematical Introduction to Logic*. Academic Press, New York.

Fellegi, I., & Sunter, A. (1969). A theory for record linkage. In *Journal of The American Statistical Association*.

Gaifman, H. (1964). Concerning measures in first order calculi. *Israel J. Math.*

Geiger, D., & Heckerman, D. (1996). Beyond bayesian networks: Similarity networks and bayesian multinets. *Artificial Intelligence*, *82*, 45–74.

Gelman, A. (1996). Inference and monitoring convergence. In Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (Eds.), *Markov Chain Monte Carlo in Practice*, pp. 131–143. Chapman and Hall, London.

Getoor, L. (2001). *Learning Statistical Models From Relational Data*. Ph.D. thesis, Stanford.

Getoor, L., Friedman, N., Koller, D., & Taskar, B. (2001a). Learning probabilistic models of relational structure. In *ICML-01*.

Getoor, L., Segal, E., Taskar, B., & Koller, D. (2001b). Probabilistic models of text and link structure for hypertext classification..

Gilks, W., Richardson, S., & Spiegelhalter, D. (1996). *Markov chain Monte Carlo in practice*. Chapman and Hall, London.

Huang, T., & Russell, S. (1998). Object identification in a Bayesian context. *Artificial Intelligence*, *103*, 1–17.

Jaro, M. A. (1984). Advances in record linkage methodology as applied to matching the 1985 census of Tampa, Florida. *JASA*.

Jerrum, M., & Sinclair, A. (1997). The Markov chain Monte Carlo method: an approach to approximate counting and integration. In Hochbaum, D. S. (Ed.), *Approximation Algorithms for NP-hard Problems*. PWS Publishing, Boston.

Kersting, K., & De Raedt, L. (2000). Bayesian logic programs. In Cussens, J., & Frisch, A. (Eds.), *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*.

Koller, D., & Pfeffer, A. (1998). Probabilistic frame-based systems. In *AAAI/IAAI*.

Lawrence, S., Bollacker, K., & Giles, C. L. (1999). Autonomous citation matching. In *Agents*.

Logothetis, A. (1997). *EM Algorithms for State and Parameter Estimation of Stochastic Dynamical Systems*. Ph.D. thesis.

McCallum, A., Nigam, K., & Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD-00*.

Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Tech. rep. CRG-TR-93-1, University of Toronto.

Ngo, L., & Haddawy, P. (1995). Probabilistic logic programming and bayesian networks. In *Proceedings of the 1995 Asian Computing Science Conference*, Pathumthani, Thailand.

Nilsson, N. J. (1986). Probabilistic logic. *Artificial Intelligence*, *28*, 71–88.

Odlyzko, A. M. (1995). Asymptotic enumeration methods. In et al, R. G. (Ed.), *Handbook of Combinatorics*. Elsevier.

Pasula, H., Marthi, B., Milch, B., Russell, S., & Shpitser, I. (2002). Identity uncertainty and citation matching. In *NIPS-02*.

Pasula, H., & Russell, S. (2001). Approximate inference for first-order probabilistic languages. In *Proc. IJCAI-01*.

Pasula, H., Russell, S., Ostland, M., & Ritov, Y. (1999). Tracking many objects with many sensors. In *Proc. IJCAI-99*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California.

Pfeffer, A. (2000). *Probabilistic Reasoning for Complex Systems*. Ph.D. thesis, Stanford.

Poole, D. (1992). Logic programming, abduction and probability. In *International Conference On Fifth Generation Computer Systems*, Vol. 64, Tokyo.

Reiter, R. (1980). Equality and domain closure in first order data bases. *J. ACM*.

Sato, T., & Kameya, Y. (1997). Prism: A symbolic-statistical modeling language. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pp. 1330–1335, Nagoya.

Sinclair, A. (1992). Improved bounds for mixing rates of Markov chains and multicommodity flow. *Combinatorics, Probability and Computing*, *130*, 351–370.

Sittler, R. W. (1964). An optimal data association problem in surveillance theory. *IEEE Military Electronics*.

Wellman, M. P., Breese, J. S., & Goldman, R. P. (1992). From knowledge bases to decision models. *Knowledge Engineering Review*, *7(1)*.

Winkler, W. E. (1999). The state of record linkage and current research problems..

Appendix A

A Few Graph Theoretic Terms

Tree Width

A **tree decomposition** of an undirected graph $G = (\mathbf{V}, \mathbf{E})$ is a pair $(\{X_i : i \in I\}, T)$ where $T = (I, F)$ is a tree and $\{X_i\}$ is a collection of subsets of V such that

- The subsets contain all of V, that is $\bigcup_{i \in I} X_i = V$
- Every pair of connected vertices occurs together in at least one subset, that is $\forall (v, w) \in E, \exists i \in I s.t.u, v \in X_i$
- The subsets containing each vertex $\forall v \in V$ form a connected subtree of T

The **tree width** of the decomposition is the one less than the size of the largest subset, that is $\max_{i \in I} |X_i| - 1$. The tree width of the graph in Figure A.1 is clearly 2.

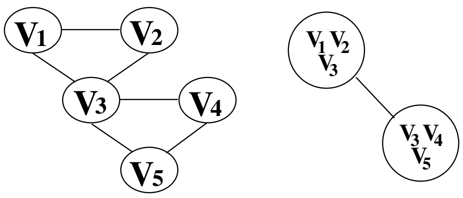


Figure A.1. A small undirected graph and its tree decomposition. The tree width is 2.

Linear Arrangements

Consider a graph $G = (\mathbf{V}, \mathbf{E})$, where the vertices are arranged according to some ordering $f : \mathbf{V} \rightarrow [1 \dots |\mathbf{V}|]$. Every integer point i of the ordering will cut across a certain number of edges,

$$|\{(u, v) \in \mathbf{E} : f(u) \leq i < f(v)\}|$$

For every ordering f this quality will be maximized at some point i. The **Minimum Cut Linear Arrangement** is the ordering which minimizes this maximum number of cut edges. This is a known problem of graph theory. It is easily extended to directed graphs: the **Minimum Cut Directed Linear Arrangement** will be the ordering which minimizes the maximum number of cut edges while respecting the ordering of the graph, i.e., an ordering where $f(u) < f(v)$ for all edges (u, v). In addition, for a directed graph, we can define a slightly different quantity. Instead of counting the number of edges being cut, we can count the number of vertices that form the end-points of these edges,

$$|\{v \in \mathbf{V} : \exists (u, v) \in \mathbf{E} : f(u) \leq i < f(v)\}|$$

This quantity might well be less than the cut defined above, since it is possible that several of the edges will lead to the same vertex. We call the ordering that minimizes the maximum number of such end-point vertices the **Minimum Destination Vertex Set Linear Arrangement**.

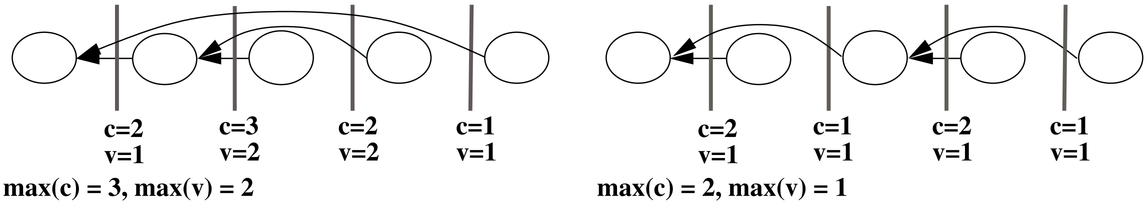


Figure A.2. Two different linear arrangements of the same directed graph. The cut (c) and the number of destination vertices (v) is marked at each integer point. It should be clear that the second arrangement is better (in fact, it is optimal.)

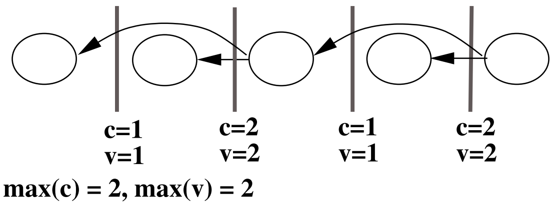


Figure A.3. The optimal linear arrangement for a fragment of a Hidden Markov Model. Both the cut (c) and the number of destination vertices (v) have two as their maximum value.

Appendix B

Set Partitions

A **set partition** of the set $\mathbf{X} = X_1, X_2, \ldots X_n$ is a collection $B_0, B_1, \ldots B_m$ of disjoint subsets of $\mathbf{X}$ whose union is $\mathbf{X}$. Each B_i is called a **block**.

The number of partitions of an n-set into m blocks is called a **Stirling number** of the second kind and is denoted $S(n, m)$. They satisfy the following recurrence relation:

$$S(n, m) = mS(n - 1, m) + S(n - 1, m - 1)$$

where $\forall m > n S(n, m) = 0$, and $S(n, 1) = 1$. Set of partitions is Σ.

The total number of partitions of an n set is a **Bell number**, denoted b_n. They satisfy the recurrence relation:

$$b_{n+1} = \sum_{i=1}^{n} \binom{n}{i} b_i$$

where $b_1 = 1$. In fact, for $n = 0, 1, \ldots 10$, the Bell number values are 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, and 115975. The asymptotic behaviour of Bell numbers is non-trivial (Odlyzko, 1995), nevertheless we can bound it as follows

$$O(2^n) < O(b_n) < O(n!)$$