Uncertainty with logical, procedural and relational languages

David Poole

Department of Computer Science,
University of British Columbia

UAI 2006 Tutorial
Outline

1. Background
   - Logic and Logic Programming
   - Knowledge Representation and Ontologies
   - Probability

2. First-order Probabilistic Models
   - Parametrized Networks and Plates
   - Procedural and Relational Probabilistic Languages
   - Inference and Learning

3. Identity, Existence and Ontologies
   - Identity Uncertainty
   - Existence Uncertainty
   - Uncertainty and Ontologies
Knowledge Representation

- Problem
  - Solve
  - Represent
- Representation
  - Compute
- Solution
  - Interpret
- Output

Informal to formal representation and solution.
What do we want in a representation?

We want a representation to be

- rich enough to express the knowledge needed to solve the problem.
- as close to the problem as possible: compact, natural and maintainable.
- amenable to efficient computation; able to express features of the problem we can exploit for computational gain.
- learnable from data and past experiences.
- able to trade off accuracy and computation time.
Notational Minefield

- Variable (probability and logic and programming languages)
- Model (probability and logic)
- Parameter (mathematics and statistics)
- Domain (science and logic and probability and mathematics)
- Grounding (logic and cognitive science)
- Object/class (object-oriented programming and ontologies)
- = (probability and logic)
- First-order (logic and dynamical systems)
First-order predicate calculus

in(alan,r123).
part_of(r123,cs_building).
in(X,Y) ←
    part_of(Z,Y) ^
in(X,Z).

alan
r123
r023
cs_building
in(•,•)
part_of(•,•)
person(•)
in(alan,cs_building)
Skolemization and Herbrand’s Theorem

**Skolemization:** give a name for an object said to exist

\( \forall x \exists y p(x, y) \) becomes \( p(x, f(x)) \)

**Herbrand’s theorem [1930]:**

- If a logical theory has a model it has a model where the domain is made of ground terms, and each term denotes itself.
- If a logical theory \( T \) is unsatisfiable, there is a finite set of ground instances of formulas of \( T \) which is unsatisfiable.
Logic Programming

definite clauses: \[
\begin{align*}
\text{part\_of}(r123, cs\_building). \\
in(alan, r123). \\
in(X, Y) &\leftarrow \text{part\_of}(Z, Y) \land in(X, Z)
\end{align*}
\]

A logic program can be interpreted:

- Logically
- Procedurally: non-deterministic, pattern matching language where predicate symbols are procedures and function symbols give data structures
- As a database language
Unique Names Assumption & Negation as Failure

- **Unique Names Assumption:**
  - different names denote different individuals
  - different ground terms denote different individuals

- **Negation as Failure:**
  - $g$ is false if it can’t be proven true
  - **Clark’s completion:**
    \[
    \forall X\forall Y \; \text{in}(X, Y) \iff (X = \text{alan} \land Y = \text{r123}) \lor \\
    (\exists Z \; \text{part\_of}(Z, Y) \land \text{in}(X, Z))
    \]
  - **stable model** is a minimal model $M$ such that an atom $g$ is true in $M$ if and only if there is a rule $g \leftarrow b$ where $b$ is true in $M$. 

---

This slide presents a detailed explanation of logical concepts, focusing on the Unique Names Assumption and Negation as Failure. It elaborates on the principles with formal expressions and clarifications, ensuring clarity in understanding these foundational logic concepts.
Acyclic Logic Programs

In acyclic logic programs

- All recursions are well-founded
- You can’t have:
  
  \[
  a \leftarrow \neg a. \\
  b \leftarrow \neg c, c \leftarrow \neg b. \\
  d \leftarrow \neg e, e \leftarrow \neg f, f \leftarrow \neg d. 
  \]

With acyclic logic programs:

- One stable model
- Clark's completion specifies what is true in that model
- Can conclude \( \neg g \) if \( g \) can’t be proved

Cyclic logic programs can have multiple stable models
- exploited by answer-set programming
Acyclic Logic Programs

In acyclic logic programs

- All recursions are well-founded
- You can’t have:
  \[ a \leftarrow \neg a. \]
  \[ b \leftarrow \neg c, \ c \leftarrow \neg b. \]
  \[ d \leftarrow \neg e, \ e \leftarrow \neg f, \ f \leftarrow \neg d. \]
- With acyclic logic programs:
  —One stable model
  —Clark’s completion specifies what is true in that model
  —Can conclude \( \neg g \) if \( g \) can’t be proved
- Cyclic logic programs can have multiple stable models
  —exploited by answer-set programming
Choosing Objects and Relations

How to represent: “Pen #7 is red.”
Choosing Objects and Relations

How to represent: “Pen #7 is red.”

- \textit{red}(\textit{pen}_7). It’s easy to ask “What’s red?”
  Can’t ask “what is the color of \textit{pen}_7?”
How to represent: “Pen #7 is red.”

- $\text{red}(\text{pen}_7)$. It’s easy to ask “What’s red?”
  Can’t ask “what is the color of $\text{pen}_7$?”

- $\text{color}(\text{pen}_7, \text{red})$. It’s easy to ask “What’s red?”
  It’s easy to ask “What is the color of $\text{pen}_7$?”
  Can’t ask “What property of $\text{pen}_7$ has value $\text{red}$?”
Choosing Objects and Relations

How to represent: “Pen #7 is red.”

- \( \text{red}(\text{pen}_7) \). It’s easy to ask “What’s red?”
  Can’t ask “what is the color of \( \text{pen}_7 \)?”

- \( \text{color}(\text{pen}_7, \text{red}) \). It’s easy to ask “What’s red?”
  It’s easy to ask “What is the color of \( \text{pen}_7 \)?”
  Can’t ask “What property of \( \text{pen}_7 \) has value \( \text{red} \)?”

- \( \text{prop}(\text{pen}_7, \text{color}, \text{red}) \). It’s easy to ask all these questions.
Choosing Objects and Relations

How to represent: “Pen #7 is red.”

- \textit{red}(\textit{pen}_7). It’s easy to ask “What’s red?”
  Can’t ask “what is the color of \textit{pen}_7?”

- \textit{color}(\textit{pen}_7, \textit{red}). It’s easy to ask “What’s red?”
  It’s easy to ask “What is the color of \textit{pen}_7?”
  Can’t ask “What property of \textit{pen}_7 has value \textit{red}?”

- \textit{prop}(\textit{pen}_7, \textit{color}, \textit{red}). It’s easy to ask all these questions.

\textit{prop(}\textit{Object, Property, Value)} is the only relation needed:
\textit{object-property-value representation, Semantic network, entity relationship model}
Universality of \( \text{prop} \)

To represent “a is a parcel”

- \( \text{prop}(a, \text{type}, \text{parcel}) \), where \( \text{type} \) is a special property
- \( \text{prop}(a, \text{parcel}, \text{true}) \), where \( \text{parcel} \) is a Boolean property
Reification

- To represent \( scheduled(cs422, 2, 1030, cc208) \). “section 2 of course cs422 is scheduled at 10:30 in room cc208.”
- Let \( b123 \) name the booking:
  - \( prop(b123, course, cs422) \).
  - \( prop(b123, section, 2) \).
  - \( prop(b123, time, 1030) \).
  - \( prop(b123, room, cc208) \).
- We have **reified** the booking.
- Reify means: to make into an object.
Triples and Semantics Networks

When you only have one relation, \textit{prop}, it can be omitted without loss of information.

\textit{prop}(\textit{Obj}, \textit{Att}, \textit{Value}) can be depicted as \langle \textit{Obj}, \textit{Att}, \textit{Val} \rangle or

\[ \begin{array}{c}
\textit{Obj} \\
\textit{Att} \\
\textit{Val}
\end{array} \]

\text{David Poole}
Uncertainty with logical, procedural and relational languages
Triples and Semantics Networks

When you only have one relation, prop, it can be omitted without loss of information. prop(Obj, Att, Value) can be depicted as \( \langle \text{Obj}, \text{Att}, \text{Val} \rangle \) or

\[
\begin{align*}
\text{Obj} & \quad \text{Att} & \quad \text{Val} \\
\end{align*}
\]
Frames

The properties and values for a single object can be grouped together into a frame.
We can write this as a list of property : value or slot : filler.

```
[owned_by : craig,
 deliver_to : ming,
 model : lemon_laptop_10000,
 brand : lemon_computer,
 logo : lemon_disc,
 color : brown,
 ...]
```
Classes

- A class is a set of individuals. E.g., house, building, officeBuilding
- Objects can be grouped into classes and subclasses
- Property values can be inherited
- Multiple inheritance is a problem if an object can be in multiple classes (no satisfactory solution)
- Need to distinguish class properties from properties of objects in the class
Knowledge Sharing

- If more than one person is building a knowledge base, they must be able to share the conceptualization.

- A conceptualization is a map from the problem domain into the representation. A conceptualization specifies:
  - What sorts of objects are being modeled
  - The vocabulary for specifying objects, relations and properties
  - The meaning or intention of the relations or properties

- An ontology is a specification of a conceptualization.
Ontologies

- Philosophy:
  - Study of existence

- AI:
  - “Specification of a Conceptualization”
  - Map: Concepts in head ↔ symbols in computer
  - Allow some inference and consistency checking
Shared Conceptualization
Semantic Web Ontology Languages

- **RDF** — language for triples in XML. Everything is a resource (with URI)
- **RDF Schema** — define resources in terms of each other: type, subClassOf, subPropertyOf
- **OWL** — allows for equality statements, restricting domains and ranges of properties, transitivity, cardinality...
- **OWL-Lite, OWL-DL, OWL-Full**
Three views of KR

- **KR as semantics** We want to devise logics in which you can state whatever you want, and derive their logical conclusions.
  
  **Examples:** Logics of Bacchus and Halpern

- **KR as common-sense reasoning** We want something where you can throw in any knowledge and get out ‘reasonable’ answers.
  
  **Examples:** non-monotonic reasoning, maximum entropy.

- **KR as modelling** We want a symbolic modelling language for ‘natural’ modelling of domains.
  
  **Examples:** logic programming, Bayesian networks.
Logic and Uncertainty

Choice:

- Rich logic including all of first-order predicate logic — use both probability and disjunction to represent uncertainty.
- Weaker logic where all uncertainty is handled by Bayesian decision theory. The underlying logic is weaker. You need to make assumptions explicit.
Logic and Uncertainty

tell \( a \lor b \)

ask \( P(a) \)

- Rich logics try to give an answer:
  \[
P(a) = \frac{2}{3}
  \]
  \[
P(a) \in [0.5, 0.75]
  \]

- Weaker logics: you have not specified the model enough.

\[
\begin{array}{ccc}
A & \lor & B \\
\text{P(a)=2/3} & & \text{P(a)=1/2} \\
\text{P(a)=3/4} & & \\
\end{array}
\]
Probability over possible worlds or individuals

To mix probability and logic, two main approaches:

- a probability distribution over possible worlds
  — a possible world is like an interpretation but can have other properties.
  — measure over sets of possible worlds where the sets are described by finite logical formulae
To mix probability and logic, two main approaches:

- a probability distribution over possible worlds
  — a possible world is like an interpretation but can have other properties.
  — measure over sets of possible worlds where the sets are described by finite logical formulae

- a probability distribution over individuals
  — proportion of individuals obeys the axioms of probability.
1. Background
   - Logic and Logic Programming
   - Knowledge Representation and Ontologies
   - Probability

2. First-order Probabilistic Models
   - Parametrized Networks and Plates
   - Procedural and Relational Probabilistic Languages
   - Inference and Learning

3. Identity, Existence and Ontologies
   - Identity Uncertainty
   - Existence Uncertainty
   - Uncertainty and Ontologies
Parametrized Bayesian networks / Plates

Parametrized Bayes Net:

\[ r(X) \]

\[ X \]

Individuals:
\[ i_1, \ldots, i_k \]

Bayes Net:

\[ r(i_1) \]
\[ \ldots \]
\[ r(i_k) \]

David Poole
Uncertainty with logical, procedural and relational languages
Parametrized Bayesian networks / Plates (2)

Individuals: $i_1, \ldots, i_k$

$X$

$r(X)\rightarrow s(X)\rightarrow t$

$q\rightarrow r(i_1)\rightarrow s(i_1)\rightarrow t$

$q\rightarrow \cdots\rightarrow q\rightarrow r(i_k)\rightarrow s(i_k)\rightarrow t$
Creating Dependencies

Common Parents

Observed Children

(q)

(r(X))

X

q

r(i_1) \ldots r(i_k)

q

Common Parents

Observed Children
Creating Dependencies: Exploit Domain Structure

\[ r(X) \]
\[ s(X) \]

\[ r(i_1) \quad r(i_2) \quad r(i_3) \quad r(i_4) \]
\[ s(i_1) \quad s(i_2) \quad s(i_3) \]

\[ X \]
\[ s(i_1) \quad s(i_2) \quad s(i_3) \]

\[ \cdots \cdots \]
Creating Dependencies: Relational Structure

\[ a \neq_k a \forall p \in P \]

\[ \forall a_i \in A \forall a_k \in A a_i \neq_k a_k \forall p \in P \]

\[ \text{collaborators}(a_i, a_k) \]

\[ \text{author}(a_i, p_j) \]

\[ \text{author}(a_k, p_j) \]

\[ \text{author}(A, P) \]

\[ \text{collaborators}(A, A') \]
Probabilistic Relational Models

- In the object-property-value representation, there is a random variable:
  - for each object-property pair for each functional property
    - The range of the property is the domain of the variable.
  - for each object-property-value there is a Boolean random variable for non-functional properties
- Plate for each class.
Probabilistic Relational Model Example

```
Job
  Job-id
  Employer
  Employee
  job class

Person
  Person-id
  Name
  town-id
  commute transport
  house size

City
  town-id
  name
  state
  density
```

David Poole

Uncertainty with logical, procedural and relational languages
A Bayesian network can be represented as a deterministic system with (independent) stochastic inputs.

<table>
<thead>
<tr>
<th>Independent Inputs</th>
<th>Deterministic System</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b \leftrightarrow (a \land bifa)$</td>
</tr>
<tr>
<td>$bifa$</td>
<td>$b \lor (\neg a \land bifna)$</td>
</tr>
<tr>
<td>$bifna$</td>
<td>$c \leftrightarrow (b \land cifb)$</td>
</tr>
<tr>
<td>$cifb$</td>
<td>$c \lor (\neg b \land cifnb)$</td>
</tr>
<tr>
<td>$cifnb$</td>
<td></td>
</tr>
</tbody>
</table>
A choice space is a set of random variables. Each random variable has a domain. [A set of the exclusive propositions corresponding to a random variable is an alternative.]

There is a possible world for each assignment of a value to each random variable. [or from each selection of one proposition from each alternative.]

The deterministic system specifies what is true in the possible world.

You can also represent decision/game theory by having multiple agents making choices.
Meaningless Example

Alternatives: \( \{ c_1, c_2, c_3 \}, \{ b_1, b_2 \} \)

\[
\begin{align*}
P_0(c_1) &= 0.5 & P_0(c_2) &= 0.3 & P_0(c_3) &= 0.2 \\
P_0(b_1) &= 0.9 & P_0(b_2) &= 0.1
\end{align*}
\]

\( f \leftrightarrow (c_1 \land b_1) \lor (c_3 \land b_2), \ d \leftrightarrow c_1 \lor (\neg c_2 \land b_1), \ e \leftrightarrow f \lor \neg d \)

Possible Worlds:

\[
\begin{array}{cccccc}
w_1 & | & c_1 & b_1 & f & d & e & \rightarrow \ & P(w_1) = 0.45 \\
w_2 & | & c_2 & b_1 & \neg f & \neg d & e & \rightarrow \ & P(w_2) = 0.27 \\
w_3 & | & c_3 & b_1 & \neg f & d & \neg e & \rightarrow \ & P(w_3) = 0.18 \\
w_4 & | & c_1 & b_2 & \neg f & d & \neg e & \rightarrow \ & P(w_4) = 0.05 \\
w_5 & | & c_2 & b_2 & \neg f & \neg d & e & \rightarrow \ & P(w_5) = 0.03 \\
w_6 & | & c_3 & b_2 & f & \neg d & e & \rightarrow \ & P(w_6) = 0.02 \\
\end{array}
\]

\[P(e) = 0.45 + 0.27 + 0.03 + 0.02 = 0.77\]
Some Representation Languages

- Independent Choice Logic (ICL): deterministic system is given by an acyclic logic program
- IBAL: deterministic system is given by a ML-like functional programming language
- A-Lisp: deterministic system is given in Lisp
- CES: deterministic system is given in a C-like language
Diagnosing students errors

\[ \begin{array}{c}
\text{x}_2 & \text{x}_1 \\
\text{y}_2 & \text{y}_1 \\
\hline
\text{z}_3 & \text{z}_2 & \text{z}_1
\end{array} \]

\[ + \]

\[ \text{carry}_3 \]

\[ \text{carry}_2 \]

\[ \text{knows addition} \]

\[ \text{knows carry} \]

\[ \text{z}_3 \]

\[ \text{z}_2 \]

\[ \text{z}_1 \]
Diagnosing students errors

\[ \begin{array}{c}
\text{carry} \\
+ \begin{array}{c}
  x_2 \\
  x_1 \\
\end{array} \\
\text{carry} \\
\hline
\text{addition} \\
\begin{array}{c}
  y_2 \\
  y_1 \\
\end{array} \\
\text{carry} \\
\hline
\begin{array}{c}
  z_3 \\
  z_2 \\
  z_1 \\
\end{array}
\end{array} \]

What if there were multiple digits
Diagnosing students errors

\[
\begin{array}{c}
  + \\
  \hline \\
  x_2 & x_1 \\
  y_2 & y_1 \\
  z_3 & z_2 & z_1 \\
\end{array}
\]

What if there were multiple digits, problems
Diagnosing students errors

What if there were multiple digits, problems, students
Diagnosing students errors

What if there were multiple digits, problems, students, times?
Example: Multi-digit addition

\[
\begin{array}{ccc}
x_j & \cdots & x_2 & x_1 \\
+ & \cdots & y_2 & y_1 \\
z_j & \cdots & z_2 & z_1 \\
\end{array}
\]

\[x \xrightarrow{\text{carry}} z\]

Student

Time

knows

addition

knows

carry

Digit

Problem

\[x \xrightarrow{\text{carry}} z\]

\[y \xrightarrow{\text{carry}} z\]
ICL rules for multi-digit addition

\[
\begin{align*}
z(D, P, S, T) &= V \leftarrow x(D, P) = Vx \land \\
y(D, P) &= Vy \land \\
carry(D, P, S, T) &= Vc \land \\
knowsAddition(S, T) &\land \\
\neg mistake(D, P, S, T) &\land \\
V \text{ is } (Vx + Vy + Vc) \text{ div } 10.
\end{align*}
\]

Alternatives:
\[
\forall DPST \{ noMistake(D, P, S, T), mistake(D, P, S, T) \}
\]
\[
\forall DPST \{ selectDig(D, P, S, T) = V \mid V \in \{0..9\} \}
\]
First-order Probabilistic Inference

- Ground the representation to a ground Bayes net
- Carry out inference in the lifted representation (without grounding unless necessary)
- Compile to secondary structure, where first-order representations lead to structure sharing.
Lifted Inference Example

Suppose we observe:

- Joe has purple hair, a purple car, and has big feet.
- A person with purple hair, a purple car, and who is very tall was seen committing a crime.

What is the probability that Joe is guilty?
Background parametrized belief network

\[
\begin{align*}
town\_conservativeness \\
\downarrow \\
hair\_colour(X) \\
\downarrow \\
sex(X) \\
\downarrow \\
height(X) \\
\downarrow \\
shoe\_size(X) \\
\downarrow \\
guilty(X) \\
\end{align*}
\]

\[X:person\]
Observing information about Joe

- sex(joe)
- height(joe)
- shoe_size(joe)
- hair_colour(joe)
- car_colour(joe)
- guilty(joe)

X: person, X ≠ joe
Observing Joe and the crime

- hair_colour(joe)
- height(joe)
- shoe_size(joe)
- guilty(joe)
- sex(joe)
- town_conservativeness
- car_colour(X)
- descn(X)
- witness
- X:person, X≠joe

Variables:
- sex
- height
- shoe_size
- hair_colour
- car_colour
- guilty
- town_conservativeness
- descn
- witness

Parameters:
- X:person, X≠joe
Guilty as a function of population

![Guilty as a function of population graph]

The graph shows the probability of guilt, $P(guilty(joe))$, as a function of the population size. As the population increases, the probability of guilt decreases.
Learning

- Although there can be an unbounded number of variables, parameter sharing means that are only a finite number of distribution parameters to learn.
- You can also define a score on structure and search for the optimal structure.
1 Background
- Logic and Logic Programming
- Knowledge Representation and Ontologies
- Probability

2 First-order Probabilistic Models
- Parametrized Networks and Plates
- Procedural and Relational Probabilistic Languages
- Inference and Learning

3 Identity, Existence and Ontologies
- Identity Uncertainty
- Existence Uncertainty
- Uncertainty and Ontologies
Identity Uncertainty

- Is this reference to the same paper as another reference?
- Is this the person who committed the crime?
- Is this patient the same as the patient who was here last week?
- Is this car the same car that was identified 3km ago?
Symbol Denotations

In logic, \( x = y \) is true if \( x \) and \( y \) refer to the same individual.

\( a \neq b, b = c, b = f(a), d = e, d \neq b, \ldots \)
In logic, $x = y$ is true if $x$ and $y$ refer to the same individual. $a \neq b$, $b = c$, $b = f(a)$, $d = e$, $d \neq b$, ...
Equality can be axiomatized with:

- $x = x$
- $x = y \Rightarrow y = x$
- $x = y \land y = z \Rightarrow x = z$
- $y = z \Rightarrow f(x_1, \ldots, y, \ldots, x_n) = f(x_1, \ldots, z, \ldots, x_n)$
- $y = z \land p(x_1, \ldots, y, \ldots, x_n) \Rightarrow p(x_1, \ldots, z, \ldots, x_n)$
Symbol Partitioning

Constants/Terms  Individuals

\[ a \]

\[ b \]

\[ c \]

\[ f(a) \]

\[ d \]

\[ e \]
Probability and Identity

- Have a probability distribution over partitions of the terms
- The number of partitions grows faster than any exponential (Bell number)
- The most common method is to use MCMC: one step is to move a term to a new or different partition.
Existence Uncertainty

- What is the probability there is a plane in this area?
- What is the probability there is a large gold reserve in some region?
- What is the probability that there is a third bathroom given there are two bedrooms?
- What is the probability that there are three bathrooms given there are two bedrooms?
Two approaches:

- **BLOG**: you have a distribution over the number of objects, then for each number you can reason about the correspondence.

- **NP-BLOG**: keep asking: is there one more? e.g., if you observe a radar blip, there are three hypotheses:
  - the blip was produced by plane you already hypothesized
  - the blip was produced by another plane
  - the blip wasn’t produced by a plane
Existence Example

- false alarm
- plane
- observe blip
- another blip
- third blip
- plane
- another plane
- same plane
- first plane
- second plane
- another plane
- another plane
- same plane
- another plane
- false alarm
Uncertainty and Ontologies

- We need to share conceptualizations.
  — People providing models and observations need to have common vocabulary.

- We need hierarchical type systems.
  — Probabilistic models may be at different levels of detail and abstraction than observations.

- ... therefore we need ontologies.
Potential Confusions

- Object-oriented programming provides valuable tools for data/code sharing, abstraction and organization.
- Use the notion of class and object:
  ```java
  class person {
      int height;
  }
  ```
  An instance of this is not a person!
- You cannot be uncertain about your own data structures!
- The notion of class and instance means something different in ontologies — this difference matters when you have uncertainty.
Ontologies and Uncertainty

- A community develops an ontology to allow semantic interoperability.
- People build probabilistic and/or preference models using this ontology.
- People describe the world using the ontology.

  e.g., models of apartments, geohazards (e.g., where is it possible that there will be a toxic spill?),...
Conclusions

- There has been much progress over 20 years.
- We don’t yet have the “Prolog” of first-order probabilistic reasoning.
- We need more experience with real applications to see what we really need.