Graphical Probability Models for Inference and Decision Making

Unit 3: Representing Knowledge in an Uncertain World

Instructor: Kathryn Blackmond Laskey
Spring 2012
Learning Objectives

• Describe
  – The elements of a knowledge representation
  – The difference between propositional and first-order logics
  – Why first-order logics are important for intelligent reasoning
• Name and describe some major first-order languages for probabilistic reasoning
  – OOBNs; PRMs; MEBN; RBN, MLN
• Define a Multi-Entity Bayesian Network
  – Reusable model components
  – Basis for knowledge-based model construction
• Define a situation-specific Bayesian network.
• Given a simple MEBN model and a query, construct a situation-specific BN to respond to the query.
Unit 3 Outline

• Knowledge Representation
• Logic and Ontologies
• Expressive Probabilistic Languages
• Probabilistic Ontology
Representing The World

• Representation requires:
  – A representing system
  – A represented system
  – A correspondence between the representing system and the represented system

• Intelligent systems use representations to respond intelligently to their environments
  – Sense
  – Recognize
  – Plan and act

• A good representation should:
  – Contain features corresponding to important properties of the represented system
  – Capture how structure & behavior of represented system gives rise to observable evidence
  – Improve with experience
  – Rest on a mathematically sound and scientifically principled logical foundation

http://www.oz-q.com/brent/pic/
Advantages of a Good Knowledge Representation

• Problems that are difficult in one representation become easy in another
  – Find MCMXLIV – (CDXLIV + MD)
  – Find 1944 – (444 + 1500)

• Knowledge is different from data or information
  – *Information*: stimuli that has meaning in some context to its receiver (techtarget.com)
  – *Data*: information that has been translated into a form that is more convenient to move or process (techtarget.com)
  – *Knowledge*: Expertise and skills acquired through experience or education; the theoretical or practical understanding of a subject. (Oxford English Dictionary)

• Formal knowledge representation represents *semantics* of a domain
  – Knowledge structures reflect structure of the domain
  – Facilitates maintenance and reuse
  – Supports sharing and semantic interoperability
  – Supports efficient reasoning
Knowledge Representation
(Davis, 1993)

• A knowledge representation is a surrogate
  – Cannot store physical objects and processes in a computer
  – Symbols and links form model of an external system
    » Variables serve as surrogates for entities they designate
    » Variables are transformed to simulate behavior of system

• A knowledge representation is a set of ontological commitments
  – Ontology determines categories of things that can exist in the model

• A knowledge representation is a fragmentary theory of intelligent reasoning
  – Describes things, behavior, interactions
  – Declarative: stated as explicit axioms & allowable transformations
  – Procedural: compiled into executable programs

• A knowledge representation is a medium for efficient computation
  – Must encode knowledge in form that can be processed efficiently

• A knowledge representation is a medium of human expression
  – Vehicle of communication between knowledge engineers and domain experts
Components of Computational Representation

• Vocabulary
  – Variables, constants, functions, connectives

• Syntax
  – Rules for composing legal expressions
  – Organization into higher level structures or patterns
    » Frames
    » Objects
    » Scripts

• Proof rules (operational semantics)
  – Rules for deriving expressions from other expressions
  – Corresponds to operational semantics of computer language

• Semantics - characterizes meaning of expressions
  – Ontology or theory of reference (denotational semantics)
  – Theory of truth (axiomatic semantics)
Unit 3 Outline

- Knowledge Representation
- Logic and Ontologies
- Expressive Probabilistic Languages
- Probabilistic Ontology
Logic

• Logic is the study of precise patterns of reasoning
  – Aristotle developed classical syllogisms
  – Boole developed Boolean logic (19th century)
  – Leibnitz formalized Aristotle’s syllogisms (17th century)
  – Frege and Pierce developed first-order logic (late 19th century)
  – Undecidability results, higher order logics, modal logics, (20th century)

• Russell and Norvig (2002) define a logic as:
  – A formal language for expressing knowledge
    » Must have precisely defined syntax and semantics
  – A means of carrying out reasoning in such a language
    » Must have precisely defined reasoning processes that correspond to semantics of the language

• Logic concerns structure not content
  – A valid syllogism:
    » All querxsms are frjplantes
    » Morxengro is a querxsms
    » Therefore, Morxengro is a frjplantes
  – Logic formalizes reasoning so it can be performed mechanistically
Expressive Power

• *Propositional* logic can express particular facts but not generalizations
  – Language has statements but no variables
  – We can say “If Jack owns a dog then Jack is an animal lover”
  – We cannot say “Any person who owns a dog is an animal lover”
• *First-order* logic can express generalizations about objects in the domain of application
  – Variables can refer to objects
  – Quantifiers can express generalizations about objects in the domain and state the existence of objects having given properties
    » For all numbers \( n \) and \( m \), \( n+m \) is equal to \( m+n \)
    » There is a fire station in every town
• *Higher-order* logic can express generalizations about sets of objects in the domain, functions defined on the domain, or properties of objects
  – Some things can be said in higher-order logic that cannot be said in first-order logic (such as the full principle of mathematical induction)
• *Modal* logics can reason not just about truth and falsehood, but also about necessity, possibility, desirability, permissibility, and other non truth-functional attributes of propositions
  – Modal logics are also strictly more expressive than first-order logic
Did Curiosity Kill the Cat?  
(an artificial FOL example due to Russell and Norvig)

- Our knowledge (in English):
  1) Jack owns a dog.
  2) Every dog owner is an animal lover.
  3) No animal lover kills an animal.
  4) Either Jack or Curiosity killed the cat, who is named Tuna.

- Our knowledge (translated into first-order logic)
  0) $\forall x \text{ Cat}(x) \Rightarrow \text{Animal}(x)$  
     (background knowledge: all cats are animals)
  1) $\exists x \text{ Dog}(x) \land \text{Owns}(\text{Jack},x)$
  2) $\forall x (\exists y \text{ Dog}(y) \land \text{Owns}(x,y)) \Rightarrow \text{AnimalLover}(x)$
  3) $\forall x (\text{AnimalLover}(x) \Rightarrow \forall y \text{ Animal}(y) \Rightarrow \neg \text{Kills}(x,y))$
  4a) $\text{Kills}(\text{Jack},\text{Tuna}) \lor \text{Kills}(\text{Curiosity},\text{Tuna})$
  4b) $\text{Cat}(\text{Tuna})$

- Can you prove that Curiosity killed the cat?
Propositional and First-Order Expressive Power

• Recall our Curiosity example:
  0) \( \forall x \text{Cat}(x) \Rightarrow \text{Animal}(x) \)
  1) \( \exists x \text{Dog}(x) \land \text{Owns}(\text{Jack},x) \)
  2) \( \forall x (\exists y \text{Dog}(y) \land \text{Owns}(x,y)) \Rightarrow \text{AnimalLover}(x) \)
  3) \( \forall x (\text{AnimalLover}(x) \Rightarrow \forall y \text{Animal}(y) \Rightarrow \neg \text{Kills}(x,y)) \)
  4a) \( \text{Kills}(\text{Jack},\text{Tuna}) \lor \text{Kills}(\text{Curiosity},\text{Tuna}) \)
  4b) \( \text{Cat}(\text{Tuna}) \)

• Can we represent this example in propositional logic?
  – We would have to replace each generalization with a list of instances, e.g.,
    2a) \( \text{Dog}_D0 \land \text{Owns}_\text{Jack}_D0 \Rightarrow \text{AnimalLover}_\text{Jack} \)
    2b) \( \text{Dog}_D1 \land \text{Owns}_\text{Fred}_D1 \Rightarrow \text{AnimalLover}_\text{Fred} \)
    2c) …

• First-order logic is to propositional logic as algebra is to arithmetic
Classical First-Order Logic

• Vocabulary:
  – Constants (stand for particular named objects)
  – Variables (stand for generic unnamed objects)
  – Functions (represent attributes of objects or sets of objects)
    » Location(x); MotherOf(y); TeacherOf(c,s)
  – Predicates (represent hypotheses that can be true or false)
    » Guilty(s)
    » Near(John,GroceryStore32)
  – Connectives
    » Quantifiers, conjunction, disjunction, implication, negation, equality

• Syntax:
  – Atomic sentences
  – Composition rules for forming compound sentences from atomic sentences

• Semantics
  – Possible worlds are abstract structures that specify truth-values of sentences
  – A sentence is valid if it is true in all possible worlds
  – A sentence follows logically from a set of axioms if it is true in every possible world in which the axioms are true

• Proof rules
  – Natural deduction
  – Resolution with refutation
Privileged Status of FOL

• Has been proposed as unifying language for
  – Defining extended logics
  – Interchanging knowledge
• Many common KR systems have theoretical basis in FOL
• Common Logic issued as ISO standard October 2007
  – Family of first-order logic syntaxes sharing common semantics
  – Designed for knowledge interchange
• Issues:
  – Cannot express generalizations about sets, predicates, functions
  – No built-in structures for
    » Categories
    » Time and space
    » Causality
    » Action
    » Events
    » Value / utility
  – Cannot represent gradations of plausibility
Ontology

- Ontology connects formal language to things in the world
  - Categories of entities organized hierarchically into types / subtypes
    » Objects of a given type have:
      • Similar structure (part-whole composition)
      • Similar behavior (processes)
      • Similar associations
  - Attributes entities can have
  - Relationships entities can have to each other

- Specifying an ontology:
  - Formal - defined by logical rules
  - Informal - specified via prototypical instances

- Ontology for a first-order language defines:
  - Allowable predicates and functions
  - Types of entities variable symbols can refer to
  - Entities denoted by constant symbols

- Explicit computational specification of ontology facilitates interoperability and reuse
Ontology of Curiosity’s World

• Types of things that can exist
  – Animals, Dogs, Cats, People

• Properties (characteristics that can be true or false)
  – People can have the property AnimalLover

• Relationships
  – Subtype: Dogs, Cats and People are subtypes of Animal
  – Killing: Animals can kill other animals
  – Ownership: People can own animals (other than people)

• Attributes (can take one of a set of possible values)
  – Because “Owns” is a functional relationship we can define “Owner” attribute of animals
Ontology Supports Logical Reasoning

Concepts
- animal, carnivore, herbivore

Relationships
- carnivore is-a animal
- herbivore is-a animal
- carnivore eats herbivore
- lion is-a carnivore
- zebra is-a herbivore
Ontology Supports Logical Reasoning

Concepts
animal, carnivore, herbivore

Relationships
- carnivore is-a animal
- herbivore is-a animal
- carnivore eats herbivore
- lion is-a carnivore
- zebra is-a herbivore

Reasoning
- lion eats zebra
Logical Reasoning
may be inadequate

• Our knowledge base:
  – All birds lay eggs
  – All aquatic birds can swim
  – All aquatic birds can hold their breath
  – Most aquatic birds have duck-like bills
  – Most aquatic birds have webbed feet

• The problem-specific data:
  – Pamela lays eggs
  – Pamela has a duck-like bill
  – Pamela has webbed feet
  – Pamela can swim
  – Pamela can hold her breath

• Therefore: Pamela is a…
Logical Reasoning may be inadequate

• Our knowledge base:
  – All birds lay eggs
  – All aquatic birds can swim
  – All aquatic birds can hold their breath
  – Most aquatic birds have duck-like bills
  – Most aquatic birds have webbed feet

• The problem-specific data:
  – Pamela lays eggs
  – Pamela has a duck-like bill
  – Pamela has webbed feet
  – Pamela can swim
  – Pamela can hold her breath

• Therefore: Pamela is a…

  Mammal!
Unit 3 Outline

- Knowledge Representation
- Logic and Ontologies
- Expressive Probabilistic Languages
- Probabilistic Ontology
Possible and Probable Worlds

• A classical logic KB consists of sentences called axioms
• The axioms implicitly define a set of possible worlds
• To reason about a given problem:
  – We supply some additional sentences to represent problem-specific facts
  – We pose a query to infer the truth-value of a sentence of interest
• In classical logic the possible results are:
  – We may find a proof that the query sentence is true;
  – We may find a proof that the query sentence is false;
  – No proof may exist (either truth-value may be consistent with the axioms)
• First-order logic cannot rank plausibility of statements that cannot be proven true or false
• A probabilistic logic assigns probabilities to possible worlds
  – Can respond to a query with a probability even if truth-value for sentence is not determined by the axioms
  – Provable sentences have probability 1; sentence that contradict the axioms have probability 0
The Trouble with BNs

• Standard Bayesian networks:
  – Designed for a single instance
    » e.g., single vehicle at a single time
• Many problems require greater representation power:
  – Multiple entities of interest
    » Many vehicles
    » Many reports coming in at different times
  – Entities are related
    » Which vehicles go with which reports?
    » Are there unreported vehicles? Spurious reports?
  – Situation evolves in time
    » Vehicles move
    » New reports arrive
• Greater expressive power is needed

©Kathryn Blackmond Laskey
History of Probabilistic KR

• Mid to late 80’s: Heuristic KBMC
  – Levitt, Binford, Ettinger applied KBLC to computer vision and automated target recognition
  – Breese PhD dissertation on rule-based KBMC
  – Goldman PhD dissertation on KBMC for natural language understanding

• 1990’s: Probabilistic logics
  – Charniak & Goldman (1993) was the first true first-order probabilistic logic based on BN fragments
  – Halpern, Bacchus & collaborators - theoretical work in first-order probabilistic representations

• 2000 and beyond
  – Convergence between BN, database, and logic programming communities
  – Intensive development of probabilistic languages with first-order expressive power
  – Software support is beginning to appear
Extending Expressiveness of Graphical Models

• Standard graphical probability model includes random variables and dependence relationships expressed as conditional probability statements
  – \( \Pr(A \mid B, C, D) = \text{belief\_table} \)

• A “probabilistic rule” makes it explicit that this probability statement applies to any individual in a class of individuals
  – \( (\forall x) \left[ \Pr(A(x) \mid B(x), C(x), D(x)) = \text{belief\_table} \right] \)

• First-order Bayesian logic:
  – Variables and constants represent entities in domain
  – Predicates and functions represent attributes and relationships
  – Specify probability distribution over values of predicates and functions
    » Axioms are conditional probability statements, e.g.:
      • \( (\forall x) \left[ \Pr(A(x) \mid B(x), C(x), D(x)) = \text{belief\_table} \right] \)
      • \( B(R17) = b2 \)
  – A probabilistic KB implicitly represents probabilities for possible worlds
  – Rules of inference
    » Probability calculus
    » Derive probability statements from other probability statements
Expressive Probabilistic Languages

• Using different metaphors
  – Object-oriented Bayesian networks
  – Probabilistic relational models
  – Multi-entity Bayesian networks
  – Markov Logic Networks
  – Bayesian Logic Programs
  – Plates
  – … and many more

• They appeal to people from different communities
• All attempt to combine expressive language with ability to represent and manipulate uncertainty
Knowledge-Based Model Construction

• Represent probabilistic knowledge in an expressive base representation
  – Represent knowledge as fragments of a Bayesian network
  – Fragments capture stable patterns of probabilistic interrelationships
  – Fragments can be replicated

• At problem solving time
  – Bring fragments together to construct a problem-specific Bayesian network
  – Use constructed model to process queries

• A KBMC system must contain at least the following elements:
  – A base representation that represents domain dependencies, constraints, etc.
  – A model construction procedure that maps a context and/or query into a target model

• Advantages of more expressive representation
  – Understandability
  – Maintainability
  – Knowledge reuse (agile modeling)
  – Exploit repeated structure

(Breese, Goldman and Wellman, 1992)
Object-Oriented Knowledge Representation

• Recent work has suggested objects as a representation for belief network knowledge bases

• Advantages of object-oriented view
  – Ability to represent abstract data types
    » Classes represent types of object
    » Instances represent individual objects in a class
    » Objects of a given type share structure and behavior
    » Subclasses inherit structure and behavior from superclasses
  – Encapsulation
    » Public structure and methods are “seen” externally
    » Private structure and methods are available only to the object itself
  – Facilitates knowledge base development, maintenance, reuse
Object-Oriented Bayesian Networks

- Classes represent types of object
  - Attributes for a class are represented as OOBN nodes
  - *Input* nodes refer to instances of another class
  - *Output* nodes can be referred to by other classes
  - *Encapsulated* nodes are private
    » Conditionally independent of other objects given input and output nodes

- Classes may have subclasses
  - Subclass inherits attributes from superclass
  - Subclass may have additional attributes not in superclass

- Classes may be *instantiated*
  - Instances represent particular members of the class

*Hugin*® has support for OOBNs
OOBN Example: Starships and Reports

- Use the “Instance” tool to make instances of Starship MFrag and Report MFrag
- Connect output nodes in Starship MFrag to input nodes in Report MFrag
- We can see encapsulated nodes in left-hand panel but not in OOBN diagram
- Hugin does not have support for reference uncertainty (which report goes with which starship)

SS Class is output node of Starship MFragment

SS Class is input node of Report MFragment

Hugin Help has tutorial on OOBNs
Example: BNs for Forensic Genetics

- f & m are instances of the type “founder” (no parents represented)
- c is an instance of type “child” (has parents represented)

Reference: Dawid, et al. (2005)
Probabilistic Relational Models

• Elements of PRM
  – Relational schema - Represents classes, attributes and relationships (corresponds to table in relational DB schema)
  – PRM structure - Represents probabilistic dependencies & numerical probabilities
  – Skeleton - Unique identifier and template for each instance
  – Data - Fills in values for instances

• An instance of a relational schema consists of a set of objects
  – Each object belongs to one of the classes
  – A value is specified for each descriptive attribute
  – An object of the appropriate type is specified for each reference attribute

• PRM structure represents probabilistic information
  – Allows representation of repeated structure
  – Can be viewed as a set of BN fragments
  – Can be learned from data
PRM Example: Publishing Papers
(Geoor and Pfeffer, 2005)

PRM Example: Starships and Reports

- OOBN and PRM represent:
  - Classes and instances
  - Attributes
- PRM relational schema represents Subject relationship between reports and starships

PRM structure

Relational schema

Relational skeleton
Multi-Entity Bayesian Networks

- Syntax similar to first-order predicate calculus
- *MEBN fragments* represent probabilistic dependencies among related random variables
  - Random variable syntax is similar to first-order logic notation
    » RVName(variable1, variable2, ..., variablen)
    » (Ordinary) variables are placeholders for entity instances
  - Inserting instance names for the ordinary variables creates an instance of the MFrags
  - There are built-in MEBN fragments for standard logical operators (and, or..) and quantifiers
  - *Influence combination* rules specify how influences from arbitrarily many parents are combined
- *MEBN theory* is a collection of MEBN fragments that satisfies global consistency conditions
- Inference: Situation-specific Bayesian network
  - Constructed from MEBN knowledge base
  - Contains instances of MEBN fragments
- Unlike OOBNs and PRMS, MEBN has native representation for n-ary relations
Multi-Entity Bayesian Networks: Example MEBN Theory
Example SSBN (aka “ground BN”)
MEBN Specifics

• M_frag
  – Contains random variables and a fragment graph
  – Random variables may be resident, input, or context
  – Input random variables are roots
  – Context random variables are shown as isolated nodes

• Random variables
  – Every random variable is resident in exactly one M_frag, called its home M_frag
  – Random variables may be input or context random variables in any number of M_frags
  – A random variable’s distribution is defined in its home M_frag
    » Local distribution specifies how to construct a belief table for an instance of a resident random variable
    » Context random variables specify a context in which the influences defined in the fragment graph hold

• M_frag instances
  – Substitute entity identifiers for variables in the M_frag for which it is possible for all context random variables to have value True
  – Context random variables known to be true or false do not have to be represented explicitly
  – Context random variables that are uncertain are parents to all resident random variables
  – Each M_frag instance specifies a parents -> child influence
  – M_frag specifies a rule for combining influences when a random variable is resident in more than one M_frag instance
MEBN Example: Starships and Reports

There are built-in MFrags for any first-order logic sentence.

- Random variable
  - $\diamond(e)$
  - Type(e)
  - Isa($tl,e$)
- Home Mfrag
  - Type
  - Type
  - Isa
- Starship Mfrag
  - OperatorSpecies(st)
  - StarshipClass(st)
- Report Mfrag
  - Subject(rp)
  - ReportedClass(rp)

The Type, IsA, and Report Subject MFrags explicitly represent aspects of the problem that are left implicit in OOBNs and PRMs. These aspects (types, instances, relationships) are treated in a standard way in all MEBN theories.

©Kathryn Blackmond Laskey
### Starship MEBN Theory: Details

<table>
<thead>
<tr>
<th>Random Variable</th>
<th>Home MFrags</th>
<th>Also Appears In</th>
<th>States</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{(}e)$</td>
<td>Type</td>
<td>Isa</td>
<td>Unique Identifier Symbols</td>
<td>If $e$ is a unique identifier, $\hat{(}e)$ is equal to $e$ or Absurd; MFragment specifies a distribution for other constants</td>
</tr>
<tr>
<td>Type($e$)</td>
<td>Type</td>
<td>Isa</td>
<td>Type labels</td>
<td>MFragment specifies a type for each unique identifier</td>
</tr>
<tr>
<td>Isa($tl$, $e$)</td>
<td>Isa</td>
<td>Starship Report Report Subject</td>
<td>True, False</td>
<td>If $tl$ is the type of $e$ then value is True, else False</td>
</tr>
<tr>
<td>Subject(rp)</td>
<td>Report Subject</td>
<td>Report</td>
<td>Starship identifiers</td>
<td>Uniform</td>
</tr>
<tr>
<td>OperatorSpecies(st)</td>
<td>Starship</td>
<td>Report</td>
<td>Cardassian, Friend, Klingon, Romulan, Unknown</td>
<td>See table</td>
</tr>
<tr>
<td>StarshipClass(st)</td>
<td>Starship</td>
<td>Report</td>
<td>Warbird, Cruiser, Explorer, Frigate, Freighter</td>
<td>See table</td>
</tr>
<tr>
<td>ReportedClass(rp)</td>
<td>Report</td>
<td>Warbird, Cruiser, Explorer, Frigate, Freighter</td>
<td>Multiplexor; see table for single-parent distribution</td>
<td></td>
</tr>
</tbody>
</table>

### Belief Tables

#### StarshipClass(st) | OperatorSpecies(st)

<table>
<thead>
<tr>
<th>OperatorSpecies(st)</th>
<th>StarshipClass(st)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardassian</td>
<td>40.000</td>
</tr>
<tr>
<td>Friend</td>
<td>10.000</td>
</tr>
<tr>
<td>Klingon</td>
<td>22.000</td>
</tr>
<tr>
<td>Romulan</td>
<td>23.000</td>
</tr>
<tr>
<td>Unknown</td>
<td>10.000</td>
</tr>
</tbody>
</table>

#### ReportedClass(rp) | OperatorSpecies(Subject(rp))

<table>
<thead>
<tr>
<th>ReportedClass(rp)</th>
<th>OperatorSpecies(Subject(rp))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warbird</td>
<td>80.000 5.000 5.000 5.000 5.000</td>
</tr>
<tr>
<td>Cruiser</td>
<td>5.000 80.000 5.000 5.000 5.000</td>
</tr>
<tr>
<td>Explorer</td>
<td>5.000 5.000 80.000 5.000 5.000</td>
</tr>
<tr>
<td>Frigate</td>
<td>5.000 5.000 5.000 80.000 5.000</td>
</tr>
<tr>
<td>Freighter</td>
<td>5.000 5.000 5.000 5.000 80.000</td>
</tr>
</tbody>
</table>
Random Variable Classes and Instances

- **OperatorSpecies(st)** and **StarshipClass(st)** are *random variable classes*
  - Ordinary variable \( st \) denotes a placeholder for a generic entity
  - Ordinary variables begin with lowercase letters

- **OperatorSpecies(Starship1)** is an *instance* of **OperatorSpecies(st)**
  - *Unique identifiers* replace ordinary variables
  - Entity identifiers begin with uppercase letters
  - In Laskey (2005) unique identifiers begin with exclamation points (can’t do in Netica)

- Instances for all type-consistent bindings of entities to variables are exchangeable (they have the same belief tables)

The local distribution for StarshipClass and ReportedClass is given in the table. The BN diagram contains 1 instance of Starship Mfrag and 2 instances of Report Mfrag.
Open-source software developed in collaboration between GMU and University of Brasilia
Situation-Specific BN

- A situation specific network is a Bayesian network constructed specifically for a problem instance
  - Represents only relevant parts of a larger model
  - “Top down” construction - prune explicitly represented larger network to obtain smaller situation-specific network
  - “Bottom up” construction builds a situation-specific network from a knowledge base of component knowledge elements

- Key ideas for situation-specific network construction
  - Construction is query and evidence driven
    » Compute $P(\text{Target Nodes} | \text{Evidence Nodes})$
  - Network must contain query and evidence random variables
  - Random variables not computationally relevant to query
    » d-separated from target variables given evidence variables
    » “barren” nodes – have no descendents that are either evidence or target random variables
  - Nuisance nodes
    » May be computationally relevant to query
    » Can be marginalized out prior to evidence propagation without affecting posterior distribution of target random variables
Situation-Specific Bayesian Network

- **E**: Evidence node
- **T**: Target node
- **I**: Internal node
- **N**: Nuisance node
- **B**: Barren node*
- **D**: d-separated node

* B1, B3, B4, B5 are barren and also are d-separated from target nodes given evidence nodes
SSBN for Starship Example

• Evidence (findings):
  – ReportedClass(Report1)=WarBird
  – ReportedClass(Report2)=Cruiser
  – ReportedClass(Report3)=Cruiser
  – Subject(Report1)=Starship1
  – Subject(Report2)=Starship2
  – Subject(Report3)=Starship2

• Targets (what we want to know)
  – OperatorSpecies(Starship1)
  – OperatorSpecies(Starship2)

Rules for SSBN construction imply we do not need to represent:
• Isa and Type random variables when we have no type uncertainty
• Subject random variable when there is no uncertainty about the subject of a report
SSBN Construction

- **BN/DG Fragment KB**
  - Retrieve model fragments
  - Match variables
  - Attach evidence to variables

- **Streaming Evidence**

- **Query / Suggestors**
  - Combine fragments into situation-specific model
  - Update inferences and decisions

- **Query Responses**

- **Model Workspace**
Reference Uncertainty

- StarshipClass(st) in the Report MFrag is a reference random variable
  - Its argument st refers to an attribute of a different entity from the argument rp of its parent
  - We may be uncertain about which starship generated a given report
  - This is called reference uncertainty
  - An SSBN with reference uncertainty has a StarshipClass parent for each starship that could have produced the report, and a reference parent telling us which starship produced the report

- When there is reference uncertainty, the belief table for the child is specified using the multiplexor influence combination rule
  - We use the belief table specified in the MFrag and the local distribution from the parent that actually generated the report

Subject(Report0) is a parent of ReportedClass(Report0) because context random variables Starship1=Subject(Report0) and Starship2=Subject(Report0) are uncertain
Example of Multiplexor Local Distribution

Partial SSBN belief table for ReportedClass(Report0):

- Distribution for ReportedClass(rp) is based on the species operating the subject of the report
- ReportedClass(rp) does not depend on OperatorSpecies(st) for non-subject starships
Reference Uncertainty Example: Prior Model

- Two starships and three reports
- Association of reports to starships is unknown
Reference Uncertainty Example: First Report

- Subject is Starship1
- ReportedClass is Warbird
Reference Uncertainty Example: Second Report

- No information about Subject
- ReportedClass is Cruiser
Reference Uncertainty Example: Third Report

- No information about Subject
- ReportedClass is Warbird
SSBN with Reference Uncertainty

• Evidence (findings):
  – ReportedClass(Report0)=WarBird
  – ReportedClass(Report1)=Cruiser
  – ReportedClass(Report2)=WarBird
  – Subject(Report0)=Starship1
    – (Subject(Report1)=Starship1) ∨ (Subject(Report1)=Starship2)
    – (Subject(Report2)=Starship1) ∨ (Subject(Report2)=Starship2)

• Targets (what we want to know)
  – OperatorSpecies(Starship1)
  – OperatorSpecies(Starship2)

Rules for SSBN construction imply we do not need to represent:
  • Isa and Type random variables when we have no type uncertainty
  • Subject random variable when there is no uncertainty about the subject of a report
Existence Uncertainty

• Reports are fallible
  – Sensor may produce spurious reports
  – Starship may fail to be reported
• Report instance generates hypothetical starship
  – Attributes of nonexistent starship have value “NA”

Local Distribution for OperatorSpecies(Starship1)

Local Distribution for StarshipClass(Starship1)

Local Distribution for ReportedClass(Report1) and ReportedClass(Report2)
Multiple Confirming Reports
Resolve Existence Uncertainty

IsA(Starship, Starship1)
- True: 63.0%
- False: 47.0%

OperatorSpecies(Starship1)
- Cardassian: 22.5%
- Friend: 11.5%
- Klingon: 11.1%
- Romulan: 3.45%
- Unknown: 4.28%
- NA: 47.0%

StarshipClass(Starship1)
- WarBird: 42.8%
- Cruiser: 3.02%
- Explorer: 2.62%
- Frigate: 2.39%
- Freighter: 2.21%
- NA: 47.0%

ReportedClass(Report1)
- WarBird: 100%
- Cruiser: 0%
- Explorer: 0%
- Frigate: 0%
- Freighter: 0%
- NoReport: 0%

ReportedClass(Report2)
- WarBird: 35.3%
- Cruiser: 6.81%
- Explorer: 5.51%
- Frigate: 6.34%
- Freighter: 5.21%
- NoReport: 42.8%

ReportedClass(Report1)
- WarBird: 100%
- Cruiser: 0%
- Explorer: 0%
- Frigate: 0%
- Freighter: 0%
- NoReport: 0%

ReportedClass(Report2)
- WarBird: 100%
- Cruiser: 0%
- Explorer: 0%
- Frigate: 0%
- Freighter: 0%
- NoReport: 0%
Existence and Miss/False Alarm Probabilities

- Probability of detect given NA is false alarm probability
- Probability of NoDetect given not-NA is miss probability
  - Typically higher for objects that are not near the sensor
Types of Uncertainty

• First-order uncertainty
  – Attribute value uncertainty
    » TempLight(M) = ?
  – Type uncertainty
    » Type(M) = ?
  – Existence uncertainty
    » Exists(M) = ?
  – Reference uncertainty
    » MachineLocation(M1) = ?

• Higher-order uncertainty
  – Parameter uncertainty
    » Pr(RoomTemp(r)=HIGH | ACStatus(r)=BROKEN) = ?
  – Structural uncertainty
    » Is there an arc from RoomTemp(r) to ACStatus(r)?
  – Entity-Relationship uncertainty
    » Does a relationship exist between two types of entity?
Subtasks in SSBN Construction with Existence and Reference Uncertainty

• Data association
  – Given: a new report and a set of hypothesized entities
  – Task: identify which entity gave rise to the report
  – Approaches:
    » “Hard” assignment to best-fit entity
    » “Soft” assignment to multiple entities
    » Multiple-hypothesis assignment

• Hypothesis management
  – Hypothesize new entities when reports don’t match existing entities
  – Prune hypotheses that have too little support to be maintained
  – Combine similar hypotheses

• MEBN fragment retrieval and model construction

• Inference and projection
  – Declare reports as evidence for the hypotheses they support
  – Infer properties of entities and relationships among entities
  – Project forward to time of next report
Reasoning About Time

- **Dynamic Bayesian Networks (DBNs)** are commonly used to reason in domains where temporal effects are present.
- Random variables are repeated in time
  - Special case: Kalman filter
  - Special case: Hidden Markov model
- More complex temporal effects can also be modeled
- Active area of research
- We will do a unit on dynamic models

DBN Example
Markov Logic Networks

- “Just add weights” to first-order logic sentences
- Yields first-order undirected graphical model

![Diagram](image)

*Figure 1. Ground Markov network obtained by applying the last two formulas in Table I to the constants Anna(A) and Bob(B).*

<table>
<thead>
<tr>
<th>English</th>
<th>First-Order Logic</th>
<th>Clausal Form</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friends of friends are friends.</td>
<td>$\forall x \forall y \forall z \text{Fr}(x, y) \land \text{Fr}(y, z) \Rightarrow \text{Fr}(x, z)$</td>
<td>$\neg \text{Fr}(x, y) \lor \neg \text{Fr}(y, z) \lor \text{Fr}(x, z)$</td>
<td>0.7</td>
</tr>
<tr>
<td>Friendless people smoke.</td>
<td>$\forall x (\neg (\exists y \text{Fr}(x, y)) \Rightarrow \text{Sm}(x))$</td>
<td>$\text{Fr}(x, g(x)) \lor \text{Sm}(x)$</td>
<td>2.3</td>
</tr>
<tr>
<td>Smoking causes cancer.</td>
<td>$\forall x \text{Sm}(x) \Rightarrow \text{Ca}(x)$</td>
<td>$\neg \text{Sm}(x) \lor \text{Ca}(x)$</td>
<td>1.5</td>
</tr>
<tr>
<td>If two people are friends, either both smoke or</td>
<td></td>
<td>$\neg \neg \text{Fr}(x, y) \lor \text{Sm}(x) \lor \neg \text{Sm}(y)$</td>
<td>1.1</td>
</tr>
<tr>
<td>neither does.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table I. Example of a first-order knowledge base and MLN. Fr() is short for Friends(), Sm() for Smokes(), and Ca() for Cancer().
Software for MLNs

• Alchemy
  – http://alchemy.cs.washington.edu/

• Tuffy
  – http://research.cs.wisc.edu/hazy/tuffy/
Unit 3 Outline

• Knowledge Representation

• Logic and Ontologies

• Expressive Probabilistic Languages

• Probabilistic Ontology
Ontology

• An ontology is an explicit, formal representation of knowledge about a domain of application. This includes:
  – Types of entities that exist in the domain; Fighter, APV, Missile, ...
  – Properties of those entities; hasMaxSpeed, hasNetWeight, hasMaxGRate, ...
  – Relationships among entities; isCommissionedAt, supports, hasLaunchBase, ...
  – Processes and events that happen with those entities; participate in mission ...

where the term entity refers to any concept (real or fictitious, concrete or abstract) that can be described and reasoned about within the domain of application.

Probabilistic Ontology

- A probabilistic ontology is an explicit, formal representation of knowledge about a domain of application. This includes:
  - Types of entities that exist in the domain; Fighter, APV, Missile, ...
  - Properties of those entities; hasMaxSpeed, hasNetWeight, hasMaxGRate, ...
  - Relationships among entities; isCommissionedAt, supports, hasLaunchBase, ...
  - Processes and events that happen with those entities; participate in mission ...
  - Statistical regularities that characterize the domain; P(isEnemyIncursur| hasIFF = False, groundSpeed = >420Kt, isFormationMember = True) = 90%, ...
  - Inconclusive, ambiguous, incomplete, unreliable and dissonant knowledge related to entities of the domain;
  - Uncertainty about all the above forms of knowledge;

where the term entity refers to any concept (real or fictitious, concrete or abstract) that can be described and reasoned about within the domain of application.

PR-OWL Probabilistic Ontology Language

• Expresses MEBN theories in World Wide Web Consortium (W3C) recommended OWL ontology language.
  – Some aspects of MEBN are not supported by OWL
  – PR-OWL reasoners are expected to respect these conditions although they cannot be handled by OWL reasoners

• Open-source, freely available solution for representing knowledge and associated uncertainty.

• UnBBayes-MEBN (developed in collaboration between GMU and University of Brasilia)
  – Represents and reasons with MEBN theories
  – Stores MFrags in PR-OWL format
Summary and Synthesis

- **First-order logic**
  - Basis for standard symbolic AI
  - Expressive
  - Cannot represent ambiguity or uncertainty

- **Probability (traditional)**
  - Moving rapidly into mainstream AI
  - Propositional representational power (no mechanisms for reasoning about classes of individuals)
  - Represents uncertainty
  - Non-modular

- **Statistics (traditional)**
  - Probability theory applied to classes of individuals
  - Limited expressive power

- **First-order probabilistic logics synthesize logic, probability & statistics**
  - First-order expressive power
  - Represent uncertainty
  - Modular elements with global consistency constraint
  - Learning theory based on Bayesian statistics
Some References for Unit 3