First-Order Probabilistic Languages: Into the Unknown

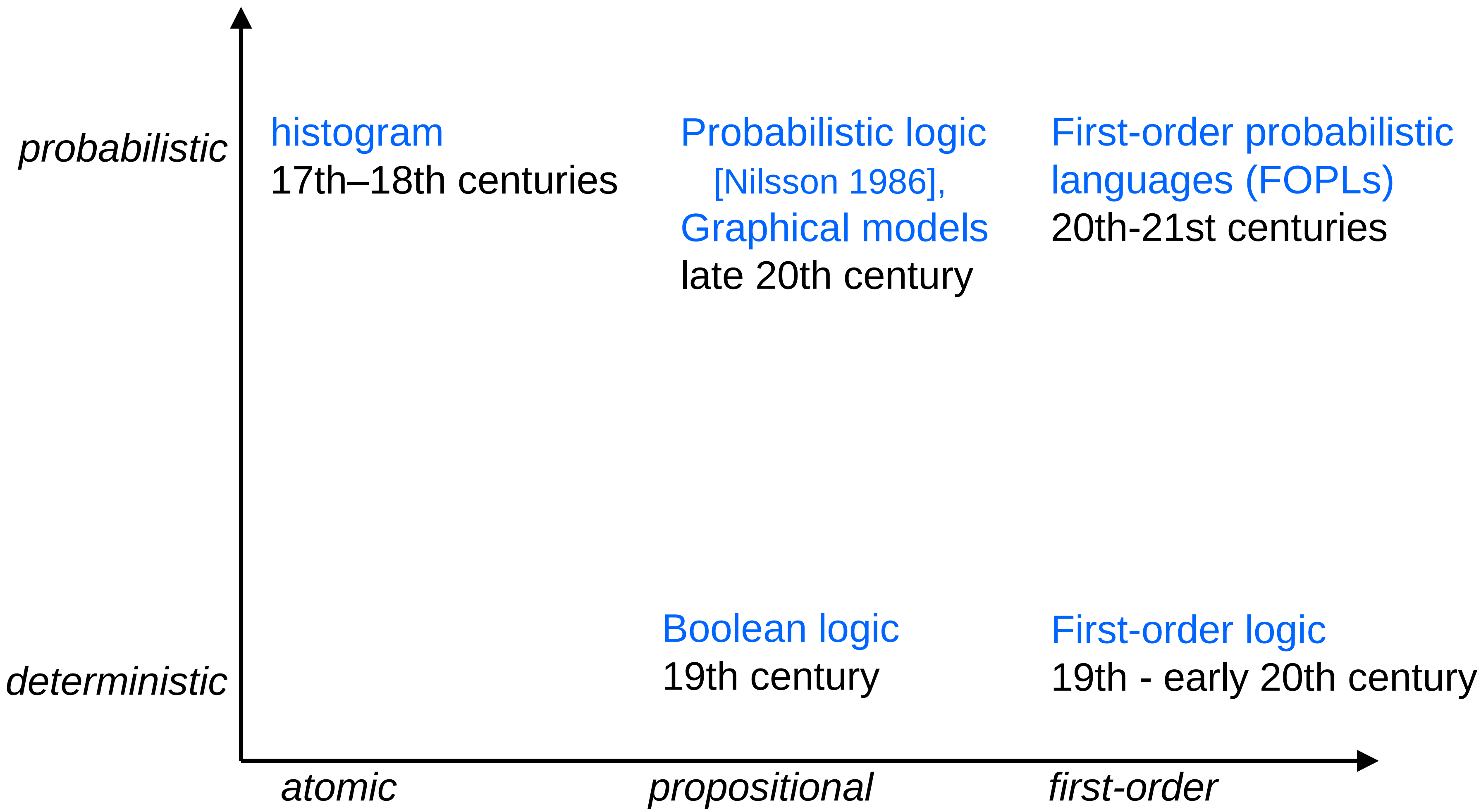
Brian Milch and Stuart Russell

University of California at Berkeley, USA

August 27, 2006

Based on joint work with
Bhaskara Marthi, David Sontag, Andrey Kolobov, Daniel L. Ong

Knowledge Representation



First-Order Probabilistic Languages (FOPLs)

Probabilistic Horn Abduction

Probabilistic Logic Programs

ProbLog

Probabilistic Entity-Relationship Models

Markov Logic Networks

Relational Bayes Nets

IBAL

Bayesian Logic Programs

PRISM

Multi-Entity Bayes Nets

Object-Oriented Bayes Nets

BUGS/Plates

Relational Markov Networks

Probabilistic Relational Models

SPOOK

Stochastic Logic Programs

Bayesian Logic

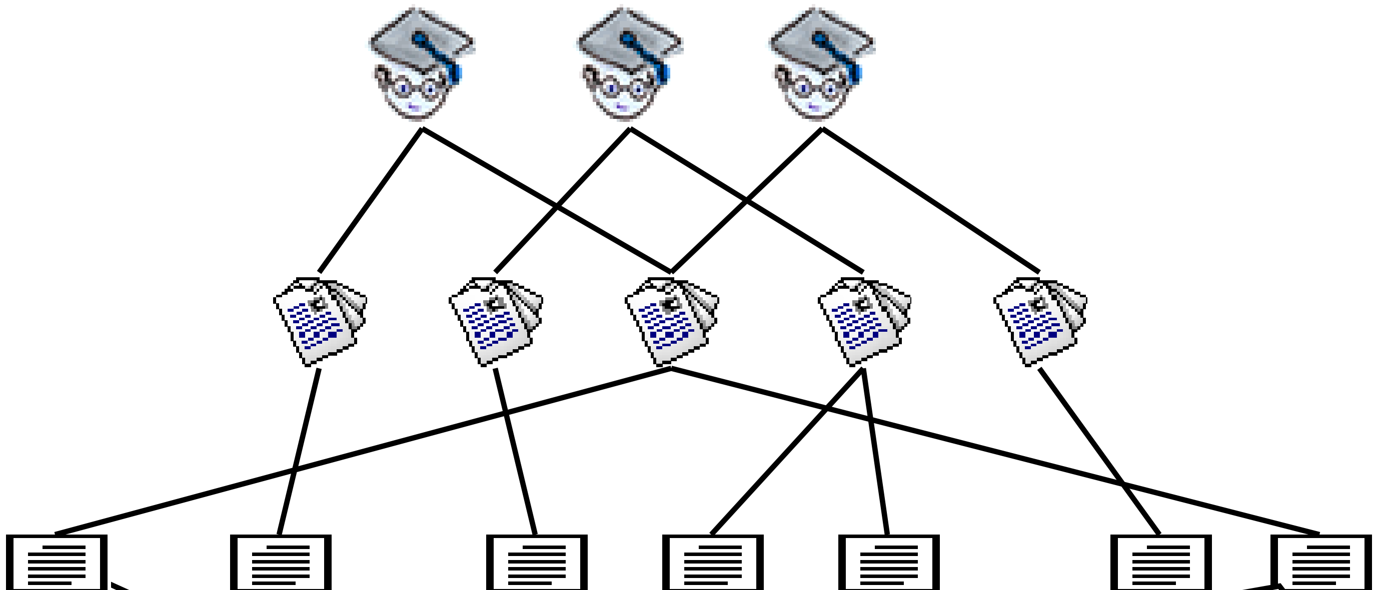
Logic Programs with Annotated Disjunctions

Logical Bayesian Networks

This Talk

- Taxonomy of FOPLs
- Design of a FOPL: Bayesian logic (BLOG)
- Inference in infinite Bayes nets
- Open problems in structure learning

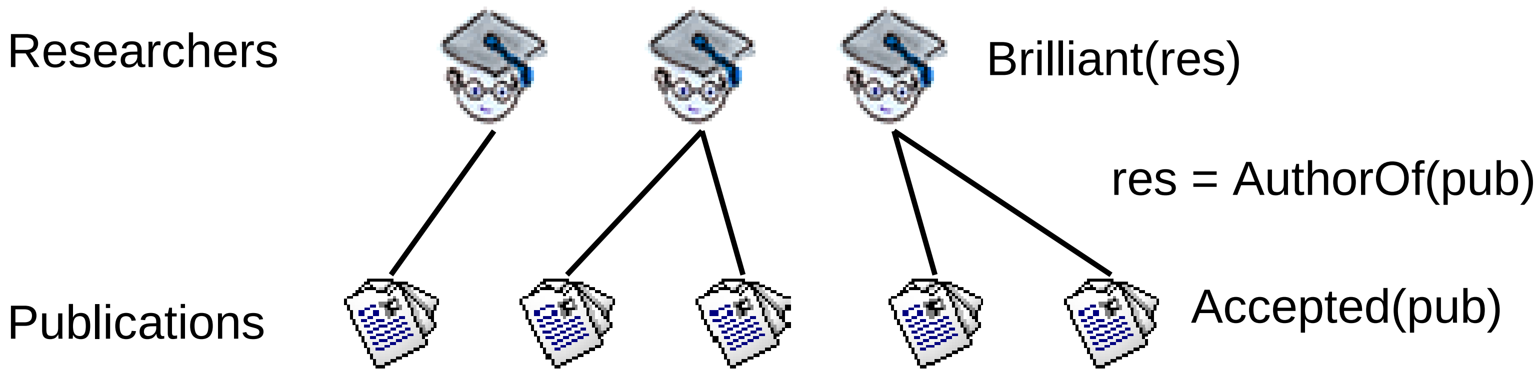
Motivating Problem: Bibliographies



Russell, Stuart and Norvig, Peter. Articial Intelligence. Prentice-Hall, 1995.

S. Russel and P. Norvig (1995). Artificial Intelligence: A Modern Approach. Upper Saddle River, NJ: Prentice Hall.

Pedagogical Example



- Tasks:
  - Infer who is brilliant
  - Predict paper acceptances
  - Infer who wrote a paper

Relational Structures

- Possible worlds are relational structures
  - Set of objects
    e.g., {Jones, Pub1, Pub2}
  - Relations and functions defined on the objects
    e.g., AuthorOf = {(Pub1, Jones), (Pub2, Jones)}
    Brilliant = {(Jones)}
    Accepted = {(Pub2)}
- Also known as: logical models / interpretations, relational databases

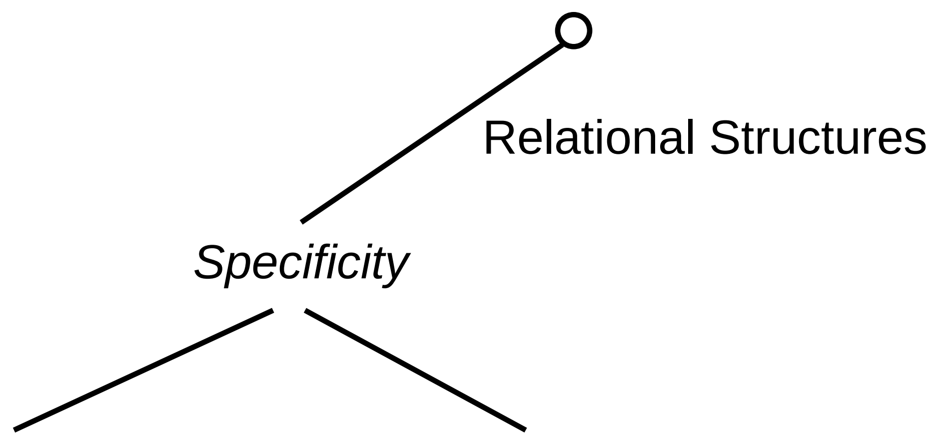
How can we define probability distributions over relational structures?

Taxonomy of FOPLs, first level

Outcome Space

- Relational structures
  - First-order interpretations
    - PHA [Poole 1992]
    - RBNs [Jaeger 1997]
    - PRISM [Sato 1997]
    - MLNs [Domingos & Richardson 2004]
    - BLOG [Milch et al. 2004]
    - ...
  - Relational databases
    - PRMs [Koller & Pfeffer 1998]
    - RMNs [Taskar et al. 2002]
    - DAPER models [Heckerman et al. 2004]
  - Instantiations of random variables
    - Early KBMC
    - BUGS/Plates [Gilks *et al.* 1994]
    - BLPs [Kersting & De Raedt 2001]
  - Nested data structures
    - IBAL [Pfeffer 2001]
- Proofs (and hence logical atoms)
  - SLPs [Muggleton 1996]

Full Specification versus Constraints



Model specifies constraints on distribution

Model specifies full distribution

e.g., $\forall x \; P(\text{Brilliant}(x)) = 0.3$

Halpern's logic of probability [1990]
PLP [Ng & Subrahmanian 1992]

Conditional Probabilities versus Weights

Relational Structures

Full Distribution

Parameterization

Conditional probability distributions (CPDs)

Define directed graph (Bayesian network)

BUGS/Plates, PHA, PRISM, RBNs, PRMs, BLPs, DAPER, BLOG, MEBN

Potentials or feature weights

Define undirected graph (Markov network)

RMNs, MLNs

Directed Models

Probability model

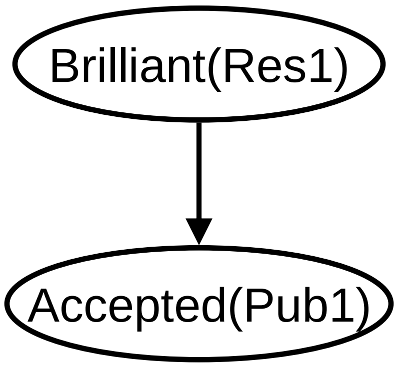
Brilliant(res) ~

P(b)	P(¬b)
0.2	0.8

Accepted(pub) ~

Brilliant (AuthorOf(pub))	P(a)	P(¬a)
b	0.8	0.2
¬b	0.3	0.7

Bayesian network (BN)



- Parameters easy to interpret
- CPDs can be estimated separately
- But need to ensure BN is acyclic

Relational skeleton

Researcher = {Res1}
Publication = {Pub1}
AuthorOf = {(Pub1, Res1)}

Directed Models

Probability model

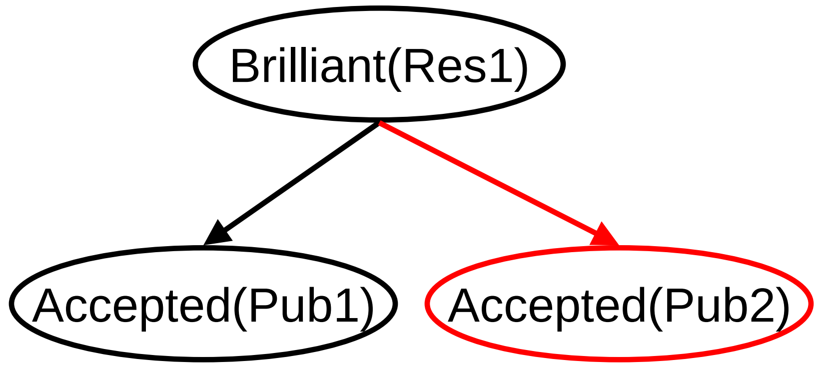
Brilliant(res) ~

P(b)	P(¬b)
0.2	0.8

Accepted(pub) ~

Brilliant (AuthorOf(pub))	P(a)	P(¬a)
b	0.8	0.2
¬b	0.3	0.7

Bayesian network (BN)



Relational skeleton

Researcher = {Res1}
Publication = {Pub1, Pub2}
AuthorOf = {(Pub1, Res1),
(Pub2, Res1)}

- Parameters easy to interpret
- CPDs can be estimated separately
- But need to ensure BN is acyclic
- Changing relational skeleton doesn't change optimal parameters

Undirected Models

Probability model

$\forall$ res,

Brilliant(res)	
b	¬b
0.2	0.8

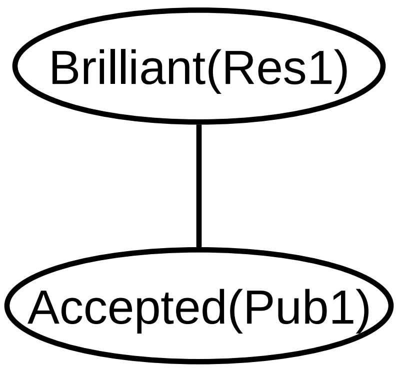
$\forall$ res, pub : res = AuthorOf(pub) $\rightarrow$

	Accepted(pub)	
Brilliant(res)	a	¬a
b	0.8	0.2
¬b	0.3	0.7

Relational skeleton

Researcher = {Res1}
Publication = {Pub1}
AuthorOf = {(Pub1, Res1)}

Markov network



- No acyclicity constraints
- But parameters harder to interpret
- Estimating parameters requires inference over whole model

Undirected Models

Probability model

$\forall$ res,

Brilliant(res)	
b	¬b
1	1

same distribution,
different parameters

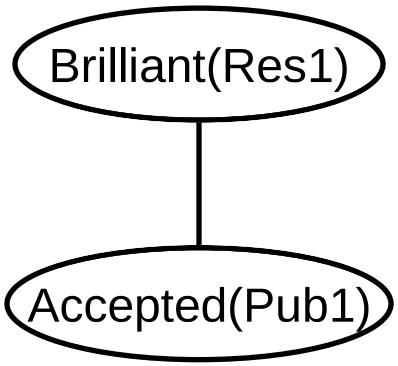
$\forall$ res, pub : res = AuthorOf(pub) $\rightarrow$

	Accepted(pub)	
Brilliant(res)	a	¬a
b	0.16	0.04
¬b	0.24	0.56

Relational skeleton

Researcher = {Res1}
Publication = {Pub1}
AuthorOf = {(Pub1, Res1)}

Markov network



- No acyclicity constraints
- But parameters harder to interpret
- Estimating parameters requires inference over whole model

Undirected Models

Probability model

∀ res,

Brilliant(res)	
b	¬b
1	1

∀ res, pub : res = AuthorOf(pub) →

	Accepted(pub)	
Brilliant(res)	a	¬a
b	0.16	0.04
¬b	0.24	0.56

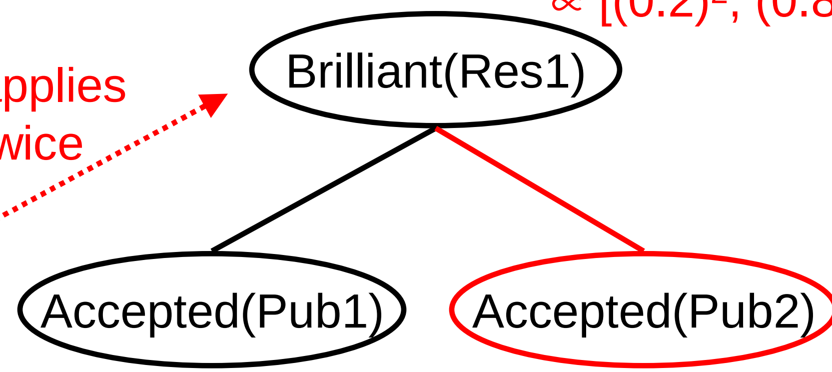
Relational skeleton

Researcher = {Res1}
Publication = {Pub1, Pub2}
AuthorOf = {(Pub1, Res1), (Pub2, Res1)}

applies twice

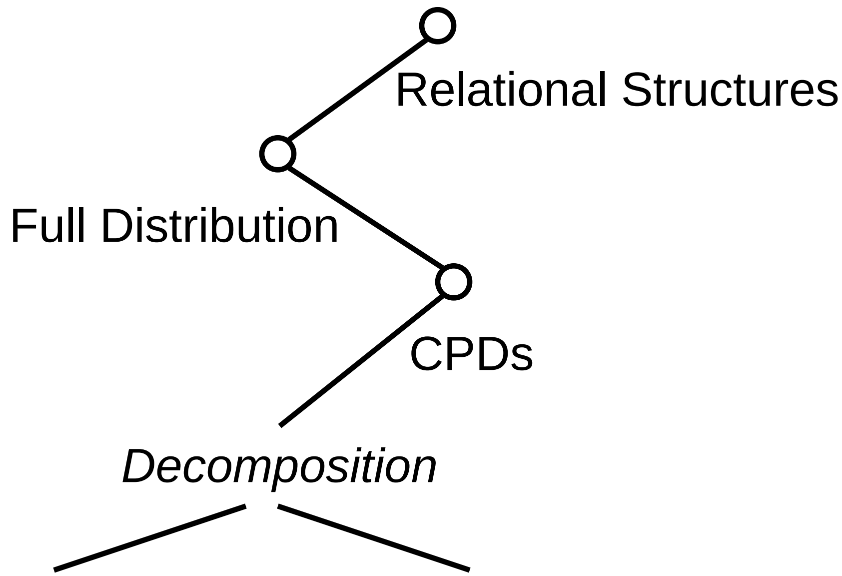
Markov network

marginal now $\propto [(0.2)^2, (0.8)^2]$



- No acyclicity constraints
- But parameters harder to interpret
- Estimating parameters requires inference over whole model
- Changing relational skeleton may change optimality of parameters

Independent Choices versus Probabilistic Dependencies



Model is decomposed into independent, random "coin flips" and logical rules

PHA, PRISM
Independent Choice Logic [Poole 1997]

Model defines probabilistic dependencies between parent and child variables

BUGS/Plates, RBNs, PRMs, BLPs, DAPER, BLOG, MEBN

Making All Random Choices Independent

- With dependent choices: Flip coin for Accepted(pub) with bias determined by Brilliant(AuthorOf(pub))
- With independent choices:
  - Flip coins for all possible values of Brilliant(AuthorOf(pub))

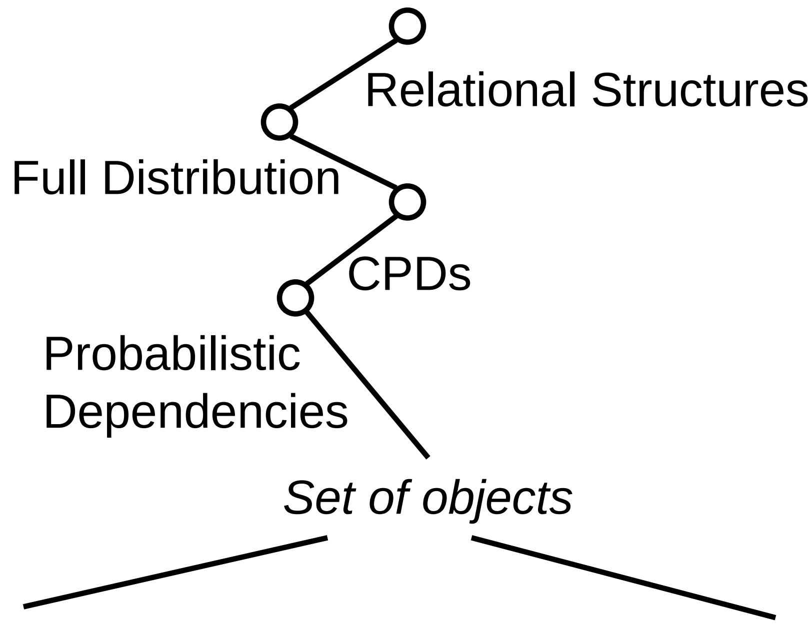
    ∀ pub Accepted_given_Brilliant(pub, True) ~ Bernoulli[0.8, 0.2]
    ∀ pub Accepted_given_Brilliant(pub, False) ~ Bernoulli[0.3, 0.7]

  - Choose which flip to use based on actual value of Brilliant(AuthorOf(pub))

    ∀ pub Accepted(pub)
    = Accepted_given_Brilliant(pub, Brilliant(AuthorOf(pub)))

- Makes algorithms more elegant, but representation more cumbersome

Known versus Unknown Objects



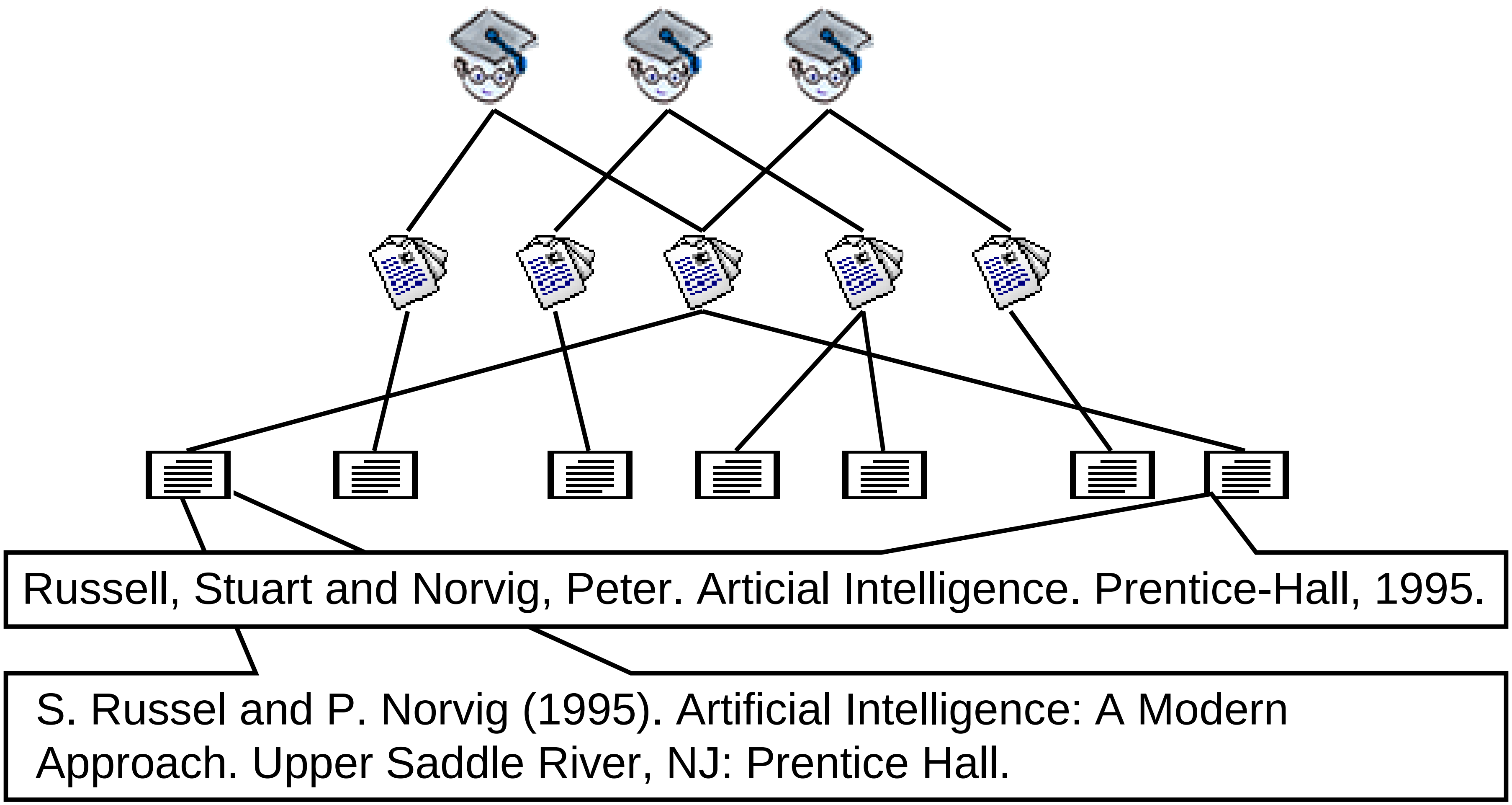
Fixed for all possible worlds, in one-to-one correspondence with symbols (e.g., Herbrand universe)

BUGS/Plates, RBNs, DAPER, BLPs

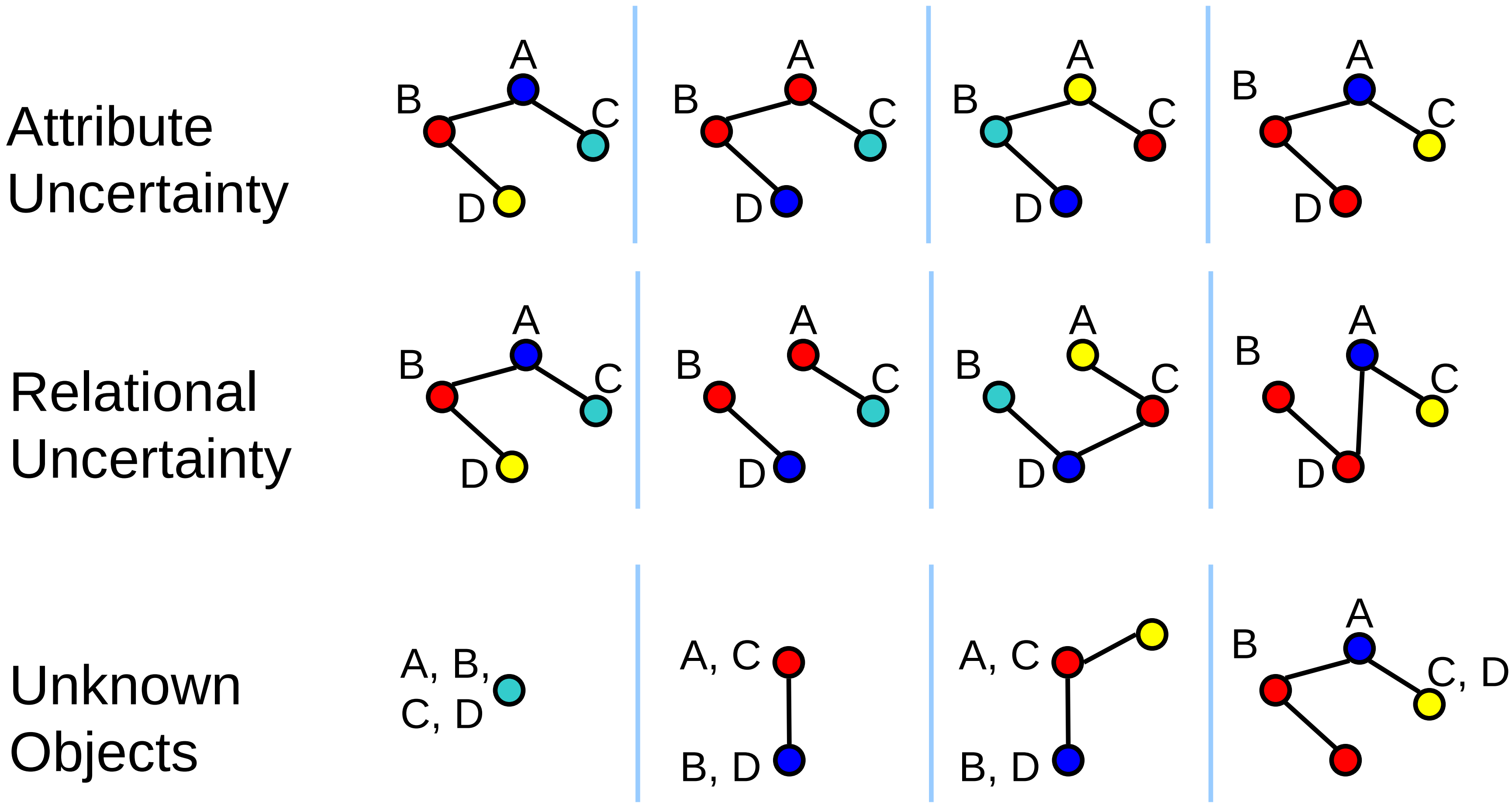
Varies from world to world, with uncertainty about symbol-object mapping

(PRMs), MEBN, BLOG

Example Again: Bibliographies



Levels of Uncertainty



Bayesian Logic (BLOG)

[Milch et al., IJCAI 2005]

- Completely defines probability distribution over model structures with varying sets of objects
- Intuition: Stochastic generative process with two kinds of steps:
  - Set the value of a function on a tuple of arguments
  - Add some number of objects to the world

BLOG Model for Bibliographies

```
guaranteed Citation Cit1, Cit2, Cit3, Cit4;

#Res ~ NumResearchersPrior();

String Name(Res r) ~ NamePrior();

#Pub ~ NumPubsPrior();

NaturalNum NumAuthors(Pub p) ~ NumAuthorsPrior();

Res NthAuthor(Pub p, NaturalNum n)
    if (n < NumAuthors(p)) then ~ Uniform({Res r});

String Title(Pub p) ~ TitlePrior();

Pub PubCited(Citation c) ~ Uniform({Pub p});

String Text(Citation c) ~ CitationCPD
       (Title(PubCited(c)),
        {Name(NthAuthor(PubCited(c), n)) for
           NaturalNum n : n < NumAuthors(PubCited(c))});
```

BLOG Model for Bibliographies

```
guaranteed Citation Cit1, Cit2, Cit3, Cit4;

#Res ~ NumResearchersPrior();

String Name(Res r) ~ NamePrior();

#Pub ~ NumPubsPrior();

NaturalNum NumAuthors(Pub p) ~ NumAuthorsPrior();

Res NthAuthor(Pub p, NaturalNum n)
    if (n < NumAuthors(p)) then ~ Uniform({Res r});

String Title(Pub p) ~ TitlePrior();

Pub PubCited(Citation c) ~ Uniform({Pub p});

String Text(Citation c) ~ CitationCPD
       (Title(PubCited(c)),
        {Name(NthAuthor(PubCited(c), n)) for
           NaturalNum n : n < NumAuthors(PubCited(c))});
```

Number statements

Dependency statements

BLOG Model for Bibliographies

```
guaranteed Citation Cit1, Cit2, Cit3, Cit4;

#Res ~ NumResearchersPrior();

String Name(Res r) ~ NamePrior();

#Pub ~ NumPubsPrior();

NaturalNum NumAuthors(Pub p) ~ NumAuthorsPrior();

Res NthAuthor(Pub p, NaturalNum n)
    if (n < NumAuthors(p)) then ~ Uniform({Res r});

String Title(Pub p) ~ TitlePrior();

Pub PubCited(Citation c) ~ Uniform({Pub p});

String Text(Citation c) ~ CitationCPD
       (Title(PubCited(c)),
        {Name(NthAuthor(PubCited(c), n)) for
           NaturalNum n : n < NumAuthors(PubCited(c))});
```

Elementary CPDs

BLOG Model for Bibliographies

```
guaranteed Citation Cit1, Cit2, Cit3, Cit4;

#Res ~ NumResearchersPrior();

String Name(Res r) ~ NamePrior();

#Pub ~ NumPubsPrior();

NaturalNum NumAuthors(Pub p) ~ NumAuthorsPrior();

Res NthAuthor(Pub p, NaturalNum n)
    if (n < NumAuthors(p)) then ~ Uniform({Res r});

String Title(Pub p) ~ TitlePrior();

Pub PubCited(Citation c) ~ Uniform({Pub p});

String Text(Citation c) ~ CitationCPD
       (Title(PubCited(c)),
        {Name(NthAuthor(PubCited(c), n)) for
           NaturalNum n : n < NumAuthors(PubCited(c))});
```

CPD arguments

Syntax of Dependency Statements

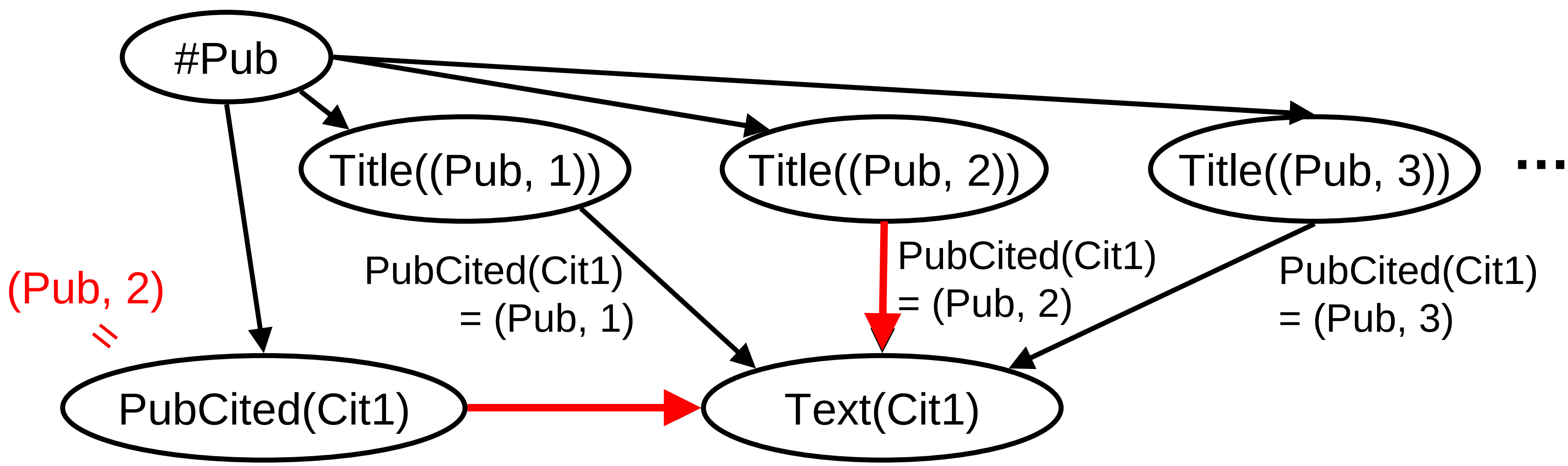
<*RetType*> F(<*ArgType*> x_1, ..., <*ArgType*> x_k)
    if <*Cond*> then ~ <*ElemCPD*>(<*Arg*>, ..., <*Arg*>)
    elseif <*Cond*> then ~ <*ElemCPD*>(<*Arg*>, ..., <*Arg*>)
    ...
    else ~ <*ElemCPD*>(<*Arg*>, ..., <*Arg*>);

- Conditions are arbitrary first-order formulas
- Elementary CPDs are names of Java classes
- Arguments can be terms or set expressions
- Number statements: same except that their headers have the form #<*Type*>

Semantics: Contingent BN

[Milch *et al.*, AI/Stats 2005]

- Each BLOG model defines a contingent BN



- *Theorem:* Every BLOG model that satisfies certain conditions (analogous to BN acyclicity) fully defines a distribution
  [see Milch *et al.*, IJCAI 2005]

Design of BLOG: Choosing Function Values

- Choosing values for functions, not just predicates

```
Pub PubCited(Citation c) ~ Uniform({Pub p});
```

- Removes unique names assumption

$$\text{PubCited(Cit1)} \stackrel{?}{=} \text{PubCited(Cit2)}$$

- Alternative in logic: relation PubCited(*c*, *p*)
  - But then BN has many Boolean PubCited nodes for each citation
  - Need to force relation to be functional

Design of BLOG: Contingent Dependencies

- Arguments passed to CPDs are determined by other variables, which can also be random

```
String Text(c) ~ CitationCPD(Title(PubCited(c)));
```

- Contrast with BLPs, where BN contains all edges that are active in any context

```
Text(c) :- Title(p), PubCited(c, p).
```

- Also contrast with languages that make context explicit, but require it to be non-random
  [Ngo & Haddawy 1997; Fierens *et al.* 2005]

```
Text(c) | Title(p) ← PubCited(c, p).
```

Design of BLOG: Explicit Aggregation

- One dependency statement per random function
  - Can have if-then-else clauses

```
String Title(Pub p)
    if Type(p) = Proceedings then ~ ProcTitlePrior
    else ~ OrdinaryTitlePrior;
```

  - Can pass multisets into CPDs

```
String Text(Citation c) ~ CitationCPD
       (Title(PubCited(c)),
        {Name(NthAuthor(PubCited(c), n)) for
           NaturalNum n : n < NumAuthors(PubCited(c))});
```

- Contrast with combining rules in BLPs, etc.

Design of BLOG: Number Statements

```
#Pub ~ NumPubsPrior();
```

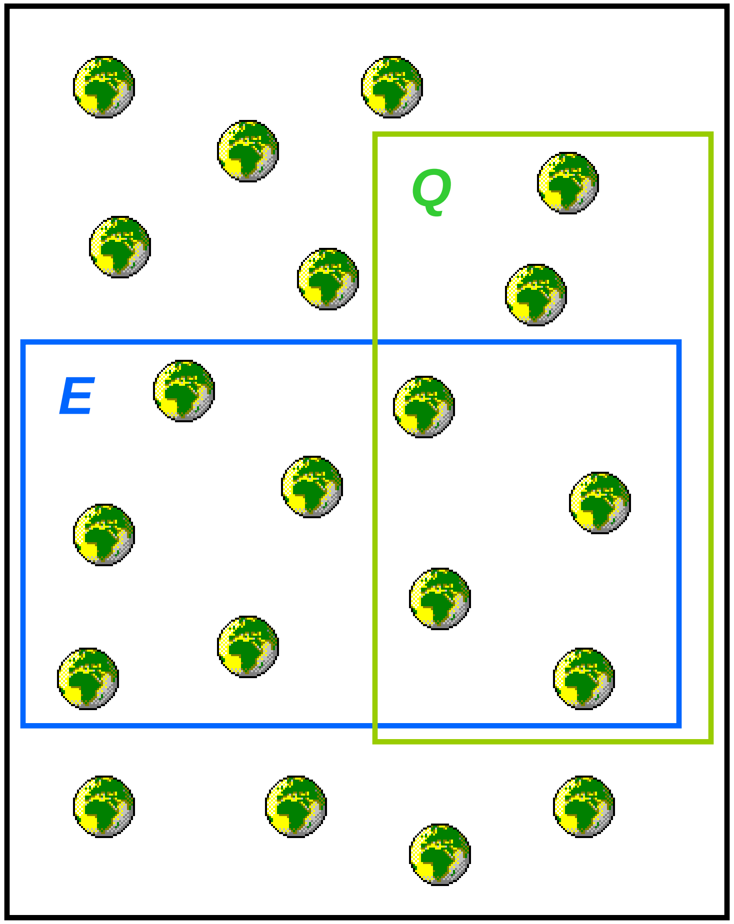
- Distribution for number of objects of a type
  - Can also have objects generating objects, e.g., aircraft generating radar blips
- Contrast with existence variables in MEBN [Laskey & Costa 2005]
  - Easier to have one number variable than sequence of existence variables
  - Number statements make interchangeability explicit
    - Can be exploited in inference; see [Milch & Russell, UAI ’06]

Inference

- Task: Find posterior probability that query ***Q*** is true given evidence ***E***

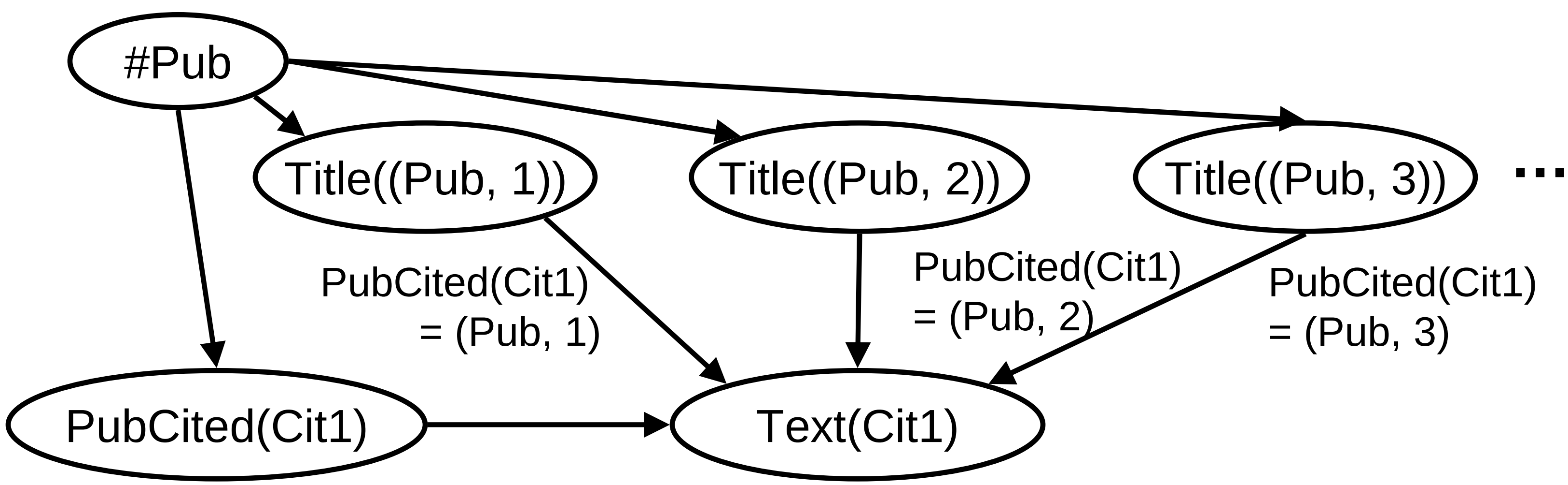
$$P(Q \mid E) = \frac{P(E \cap Q)}{P(E)}$$

- Naive solution involves summing probabilities of worlds in ***E*** and in ***E*** $\cap$ ***Q***



Inference on BNs

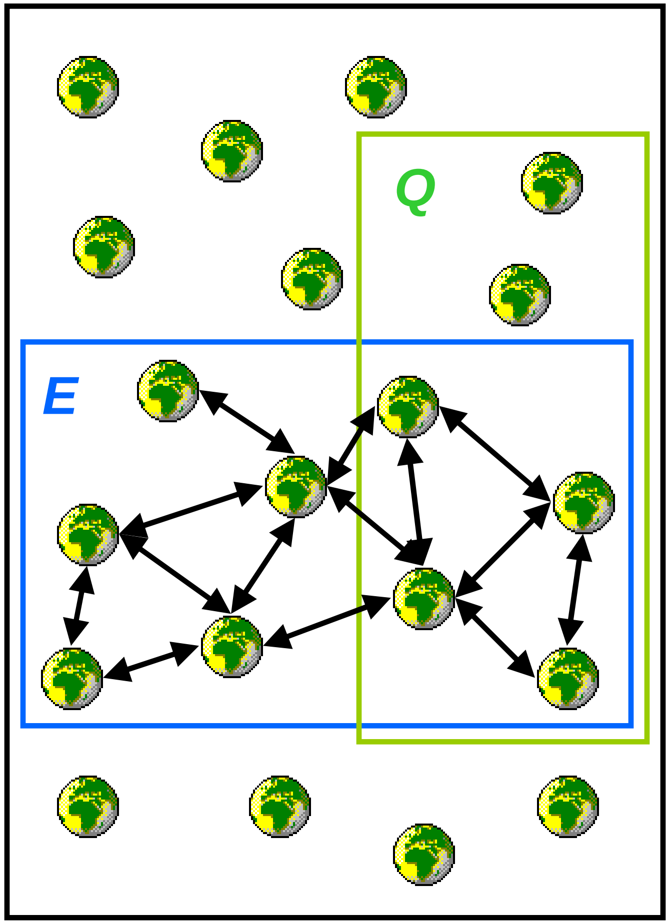
- Most common FOPL inference method:
  – Construct BN defined by model
  – Perform exact or approximate inference on BN
- But many BLOG models define infinite BNs



Exploiting Context-Specific Relevance

- Sampling algorithms only need to instantiate finite set of context-specifically relevant variables
  - Rejection sampling [Milch *et al.*, IJCAI 2005]
  - Likelihood weighting [Milch *et al.*, AI/Stats 2005]
  - Metropolis-Hastings MCMC [Milch & Russell, UAI 2006]
- *Theorem:* For structurally well-defined BLOG models, sampling algorithms converge to correct probability for any query, using finite time per sampling step

Metropolis-Hastings MCMC



- Let $\boldsymbol{s}_1$ be arbitrary state in $\boldsymbol{E}$
- For $\boldsymbol{n}$ = 1 to $\boldsymbol{N}$
  - Sample $\boldsymbol{s}' \in \boldsymbol{E}$ from proposal distribution $\boldsymbol{q}(\boldsymbol{s}' \mid \boldsymbol{s}_n)$
  - Compute acceptance probability

$$\alpha = \max\left(1, \frac{p(s')\, q(s_n \mid s')}{p(s_n)\, q(s' \mid s_n)}\right)$$

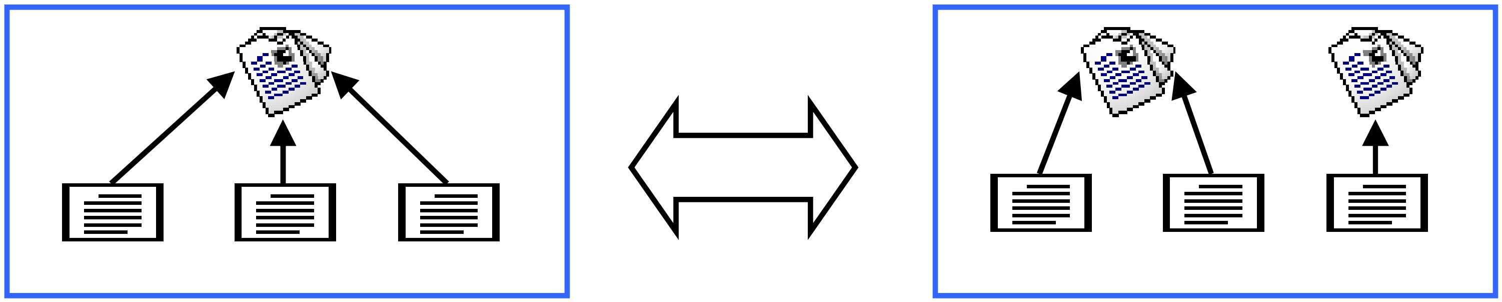
  - With probability α, let $\boldsymbol{s}_{n+1} = \boldsymbol{s}'$; else let $\boldsymbol{s}_{n+1} = \boldsymbol{s}_n$

Fraction of visited states in $\boldsymbol{Q}$ converges to $\boldsymbol{p}(\boldsymbol{Q}|\boldsymbol{E})$

Proposer for Citations

[Pasula *et al.*, NIPS 2002]

- Split-merge moves:



  – Propose titles and author names for affected publications based on citation strings

- Other moves change total number of publications

MCMC States

- Not complete instantiations!
  - No titles, author names for uncited publications
- States are partial instantiations of random variables

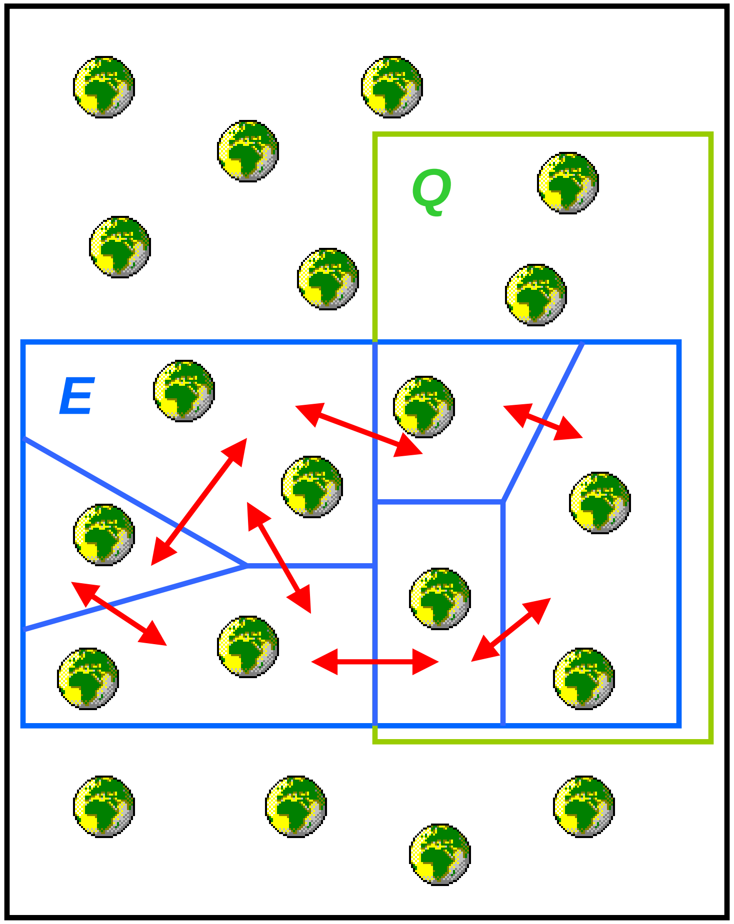
  #Pub = 100, PubCited(Cit1) = (Pub, 37), Title((Pub, 37)) = “Calculus”

  - Each state corresponds to an event: set of worlds satisfying description

MCMC over Events

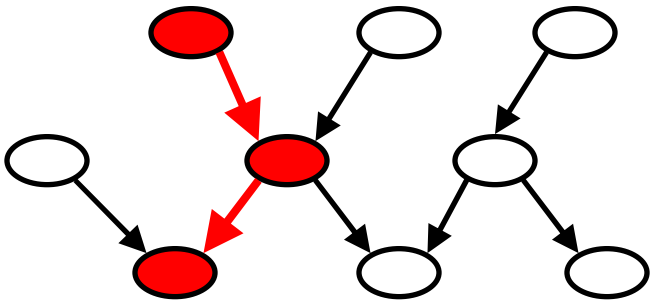
[Milch & Russell, UAI 2006]

- Markov chain over events σ, with stationary distrib. proportional to $\boldsymbol{p}(\sigma)$
- *Theorem:* Fraction of visited events in $\boldsymbol{Q}$ converges to $\boldsymbol{p}(\boldsymbol{Q}|\boldsymbol{E})$ if:
  – Each σ is either subset of $\boldsymbol{Q}$ or disjoint from $\boldsymbol{Q}$
  – Events form partition of $\boldsymbol{E}$



Computing Probabilities of Events

- Need to compute ***p***(σ′) / ***p***($\sigma_{\boldsymbol{n}}$) efficiently (without summations)
- Use instantiations that include all active parents of the variables they instantiate



- Then probability is product of CPDs:

$$p(\sigma) = \prod_{X \in \mathrm{vars}(\sigma)} p_X\left(\sigma(X) \mid \sigma(\mathrm{Pa}_\sigma(X))\right)$$

Learning

- Parameters:
  - Easy to estimate CPDs from complete data
  - With incomplete data, use EM algorithm
- Structure:
  - Choose parents [e.g., Friedman et al. 1999, Popescul et al. 2003, Landwehr et al. 2005, Kok & Domingos 2005]
  - Choose aggregation functions
  - Learn conditions under which CPDs apply

Predicate/Function Invention

- Predicate invention has long history in ILP
  - But typically new predicates are defined deterministically in terms of existing predicates
- In probabilistic case: Invent random functions
  - With existing functions as parents, as in [Revoredo *et al.*, this conference]
  - Without parents, e.g., relation Colleagues(a, b) to explain co-authorship patterns
- Inventing family of latent variables in BN

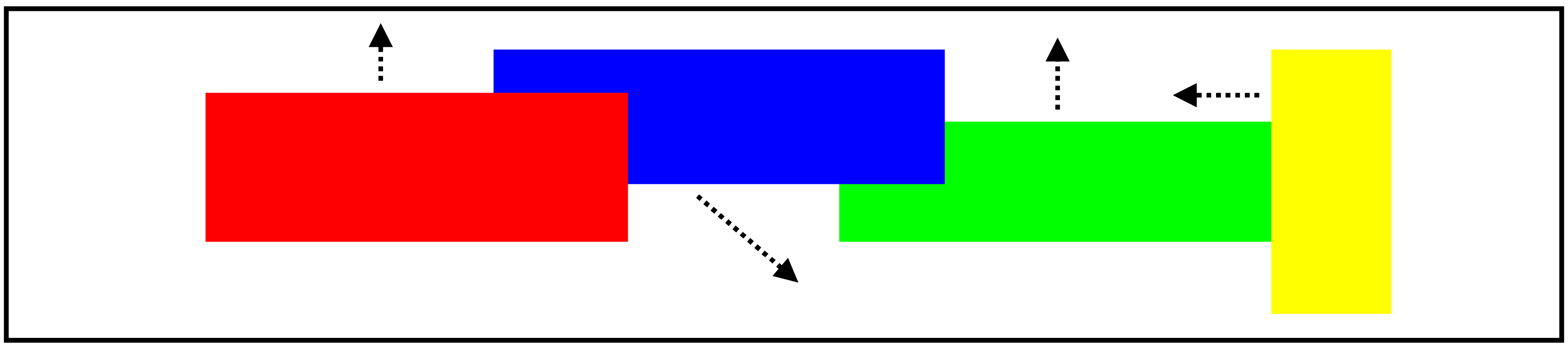
Entity Invention

- Invent new types of objects, such as:
  - Atoms (as in John McCarthy's talk)
  - Conferences, to explain recurring substrings of citation strings
- Requires representation that allows unknown objects
  - Objects of invented types will not be known to modeler in advance

Challenge Problem

[Courtesy of Prof. Josh Tenenbaum, MIT]

- Cognitive science question: could children *learn* concept of an object, or must it be innate?



- Given sequence of frames (pixel arrays), learn model that includes colored blocks
  – Initially, only functor is Color(x, y, t)

Summary

- There is method to the madness of FOPLs
- Bayesian logic (BLOG)
  - Defines full distribution over relational structures
  - Allows unknown objects, unknown mapping from symbols to objects
  - Makes contingent dependencies explicit
- Inference can be possible even when model yields infinite BN
- Exciting challenges in predicate/entity invention

http://www.cs.berkeley.edu/~milch/blog