BLOG: Probabilistic Models with Unknown Objects

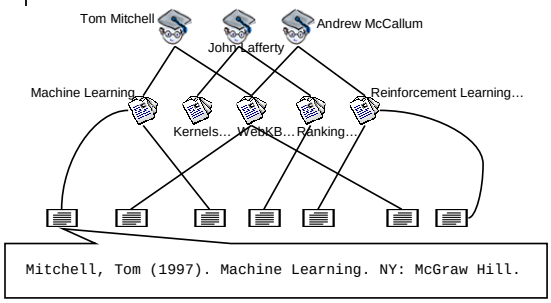
Brian Milch
CS 289
12/6/04

Joint work with Bhaskara Marthi, David Sontag, Daniel Ong, Andrey Kolobov, and Stuart Russell

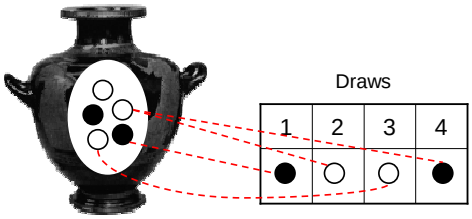
Task for Intelligent Agents

- Given observations, make inferences about underlying real-world objects
- But no list of objects is given in advance

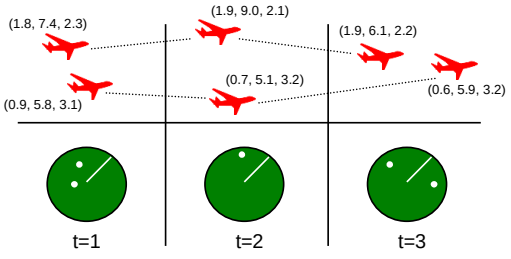
Example 1: Bibliographies



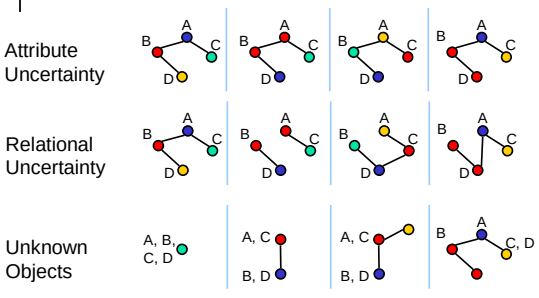
Example 2: Drawing Balls from an Urn (in the Dark)



Example 3: Tracking Aircraft



Levels of Uncertainty



Today's Lecture

- BLOG: language for representing scenarios with unknown objects
- Evidence about unknown objects
- Sampling-based inference algorithm

Why Not PRMs?

- Unknown objects handled only by various extensions:
  - number uncertainty [Koller & Pfeffer, 1998]
  - existence uncertainty [Getoor et al., 2002]
  - identity uncertainty [Pasula et al., 2003]
- Attributes apply only to single objects
  - can't have Position(***a***, ***t***)

BLOG Approach

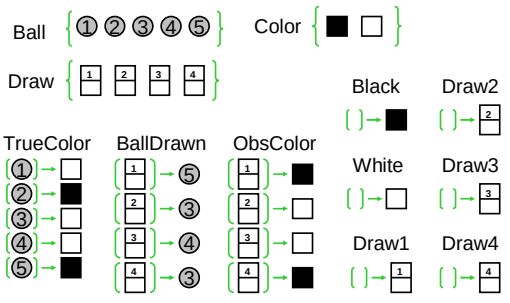
- BLOG model defines probability distribution over model structures of a typed first-order language
  [Gaifman 1964; Halpern 1990]
- Unique distribution, not just constraints on the distribution

Typed First-Order Language for Urn and Balls

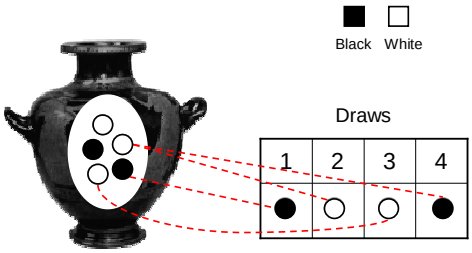
Types: Ball, Draw, Color

Symbol	Arg Types	Return Type
TrueColor(***b***)	(Ball)	Color
BallDrawn(***d***)	(Draw)	Ball
ObsColor(***d***)	(Draw)	Color
Black, White	()	Color
Draw1, …, Draw4	()	Draw

Model Structure



Generative Probability Model



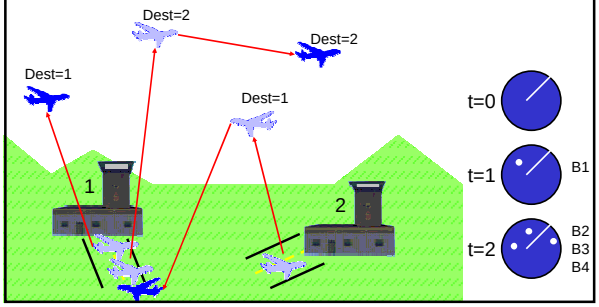
BLOG Model for Urn and Balls

```
type Color;  type Ball;  type Draw;

random Color TrueColor(Ball);
random Ball BallDrawn(Draw);
random Color ObsColor(Draw);

guaranteed Color Black, White;
guaranteed Draw Draw1, Draw2, Draw3, Draw4;

#Ball: () -> ()
    ~ Poisson[6]();

TrueColor(b)
    ~ TabularCPD[[0.5, 0.5]]();

BallDrawn(d)
    ~ UniformChoice[]({Ball b});

ObsColor(d)
    if !(BallDrawn(d) = null) then
        ~ TabularCPD[[0.8, 0.2], [0.2, 0.8]](TrueColor(BallDrawn(d)));
```

Number statement

Dependency statements

Generative Model for Aircraft Tracking



BLOG Model for Aircraft Tracking: Header

```
type AirBase; type Aircraft; type RadarBlip;

random R2Vector Location(AirBase);
random Boolean TakesOff(Aircraft, Integer);
random Boolean Lands(Aircraft, Integer);
random AirBase CurBase(Aircraft, Integer);
random Boolean InFlight(Aircraft, Integer);
random R6Vector State(Aircraft, Integer);
random AirBase Dest(Aircraft, Integer);
random R3Vector ApparentPos(RadarBlip);

generating AirBase HomeBase(Aircraft);
generating Aircraft BlipSource(RadarBlip);
generating Integer BlipTime(RadarBlip);

nonrandom Integer Pred(Integer) = PredFunction;
nonrandom Boolean Greater(Integer, Integer) = GreaterThanPredicate;
```

values determined when object is generated

Tracking Aircraft: Dependency Statements

```
Location(b)
    ~ UniformOnRectangle();

TakesOff(a, t)
    if Greater(t, 0) & !InFlight(a, t) then ~ TakeoffDistrib();

Lands(a, t)
    if Greater(t, 0) & InFlight(a, t) then ~ LandDistrib(State(a, t), Location(Dest(a, t)));

CurBase(a, t)
    if (t = 0) then = HomeBase(a)
    elseif TakesOff(a, t) then = null
    elseif Lands(a, t) then = Dest(a, Pred(t))
    elseif Greater(t, 0) then = CurBase(a, Pred(t));

InFlight(a, t)
    if Greater(t, 0) then = (CurBase(a, t) = null);

State(a, t)
    if TakesOff(a, t) then ~ InitialStateDistrib(Location(CurBase(a, Pred(t))))
    elseif InFlight(a, t) then ~ StateTransition(State(a, Pred(t)), Location(Dest(a, t)));

Dest(a, t)
    if TakesOff(a, t) then ~ UniformChoice({AirBase b})
    elseif InFlight(a, t) then = Dest(a, Pred(t));
```

Closeup of Dependency Statement

child variable

clauses

```
State(a, t)
    if TakesOff(a, t) then
        ~ InitialStateDistrib(Location(CurBase(a, Pred(t))))
    elseif InFlight(a, t) then
        ~ StateTransition(State(a, Pred(t)),
                          Location(Dest(a, t)));
```

- For a given assignment of objects to ***a***, ***t***:
  - Find first clause whose condition is satisfied
  - Evaluate CPD arguments (terms, formulas, or sets)
  - Pass them to CPD, get distribution over child variable

Aircraft Tracking: Number Statements

```
#AirBase: () -> ()
    ~ NumBasesDistrib();

#Aircraft: (HomeBase) -> (b)
    ~ NumAircraftDistrib();

#RadarBlip: (BlipSource, BlipTime) -> (a, t)
    if InFlight(a, t) then ~ NumDetectionsDistrib();

#RadarBlip: (BlipTime) -> (t)
    ~ NumFalseAlarmsDistrib();
```

Potential object patterns (POPs)

Summary of BLOG Basics

- Defining distribution over model structures of typed first-order language
- Generative process with two kinds of steps:
  - Generate objects, possibly from existing objects (described by number statement)
  - Set value of function on tuple of objects (described by dependency statement)

Advanced Topics in BLOG

- Semantic issues
  - What exactly are the objects?
  - When is a BLOG model well-defined?
- Asserting evidence
- Approximate inference

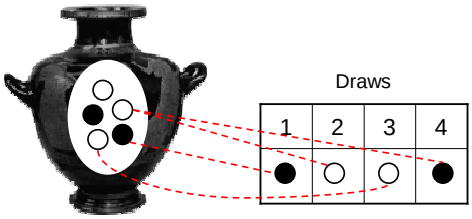
What Exactly Are the Objects?

Ball {  } Color { Black White }

Draw { Draw1 Draw2 Draw3 Draw4 }

- Substituting other objects yields isomorphic world – same formulas satisfied
- Must define distribution over specific set of possible worlds containing particular objects

Can We Avoid Representing Unobserved Objects?

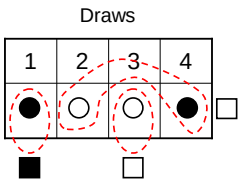


Can We Avoid Representing Unobserved Objects?

Multiset

Yes, but it's not obvious how to:
- Define distributions over multisets, partitions
- Handle relations among unobserved objects

Labeled partition



Letting Unobserved Objects Be Natural Numbers

Ball { 4 18 29 36 75 } Color { Black White }

Draw { Draw1 Draw2 Draw3 Draw4 }

- Problem: Too many possible worlds
  - Infinitely many possible worlds isomorphic to any given world
  - Can't define uniform distribution over them

Letting Unobserved Objects Be Consecutive Natural Numbers

Ball { 0 1 2 3 4 } Color { Black White }

Draw { Draw1 Draw2 Draw3 Draw4 }

- Still have isomorphic worlds obtained by permuting {0, 1, 2, 3, 4}
- But only finitely many for each given world
- Is this always true?

Representations for Radar Blips

RadarBlip { 0, 1, 2, … }

BlipTime
0 → 417
1 → 312
2 → 920
⋮ ⋮

- Infinitely many isomorphic worlds that differ in mapping from blip numbers to time steps
- Need more structured representation…

Objects as Tuples

(type, (genfunc$_1$, genobj$_1$), …, (genfunc$_k$, genobj$_k$), n)

AirBase { (AirBase, 1), (AirBase, 2), ... }

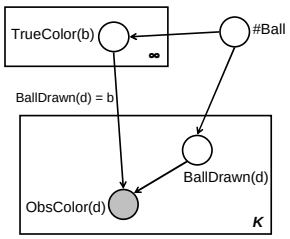
Aircraft { (Aircraft, (HomeBase, (AirBase, 1)), 1), ...
(Aircraft, (HomeBase, (AirBase, 2)), 1), ...
⋮ }

RadarBlip { ⋮
(RadarBlip, (BlipSource, (Aircraft, (HomeBase, (AirBase, 2)), 1)), (BlipTime, 8), 1)
⋮ }

Advantage of Tuple-Based Semantics

- Possible world is uniquely identified by instantiation of basic random variables:
  - Variable for each function and tuple of arguments – yields value
  - Variable for each POP and tuple of generating objects – yields number of objects generated
- So BLOG model just needs to define joint distribution over these variables

Graphical Representation of BLOG Model



- Like a BN, but:
  - Edges are only active in certain contexts
  - Ignoring contexts, ObsColor(d) has infinitely many parents
  - In other models, graph may be cyclic if you ignore contexts

Well-Defined BLOG Models

- BLOG model defines not ordinary BN, but contingent Bayesian network (CBN) [Milch et al., AI/Stats 2005]
- If this CBN satisfies certain context-specific finiteness and acyclicity conditions, then BLOG model defines unique distribution – it is well-defined

Checking Well-Definedness

- CBN for BLOG model is infinite
- Can we check well-definedness just by inspecting the (finite) BLOG model?
- Yes, by drawing abstract graph where nodes correspond to functions/POPs rather than individual variables
  - sound, but not complete
  - kind of like proving program termination

Evidence and Unknown Objects

- Evidence for urn and balls:

```
ObsColor(Draw1) = Black;
```

  - Can use constant symbols for draws because they are guaranteed objects
- Evidence for aircraft tracking:
  - Want to assert number of radar blips at time ***t***, ApparentPos value for each blip
  - But no symbols for radar blips!

Existential Evidence

- To say there are exactly 2 blips at time 8, with certain apparent positions:

$$\exists \text{RadarBlip}\, b_1, b_2 \begin{pmatrix} (\text{BlipTime}(b_1) = 8) \wedge (\text{BlipTime}(b_2) = 8) \\ \wedge (b_1 \neq b_2) \\ \wedge \neg \exists b_3 ((\text{BlipTime}(b_3) = 8) \wedge (b_3 \neq b_1) \wedge (b_3 \neq b_2)) \\ \wedge \text{ApparentPos}(b_1) = (9.6 \quad 1.2 \quad 32.8) \\ \wedge \text{ApparentPos}(b_2) = (2.9 \quad 1.6 \quad 3.3) \end{pmatrix}$$

- But this is awkward, doesn't allow queries about, say, State(BlipSource(***b***$_1$), 8)

Skolemization

- Introduce Skolem constants B1, B2

$$(\text{BlipTime}(\text{B1}) = 8) \wedge (\text{BlipTime}(\text{B2}) = 8)$$
$$(\text{B1} \neq \text{B2})$$
$$\neg \exists b_3 ((\text{BlipTime}(b_3) = 8) \wedge (b_3 \neq \text{B1}) \wedge (b_3 \neq \text{B2}))$$
$$\text{ApparentPos}(\text{B1}) = (9.6 \quad 1.2 \quad 32.8)$$
$$\text{ApparentPos}(\text{B2}) = (2.9 \quad 1.6 \quad 3.3)$$

- Now can query State(BlipSource(B1), 8)
- But what is distribution over interpretations of B1, B2?

Exhaustive Observations

- Our evidence asserts that B1, B2 exhaust {RadarBlip b : BlipTime(b) = 8}
- Theorem: If evidence asserts B1, …, BK exhaust S, then conditioning on evidence is equivalent to:
  - assuming values of B1, …, BK are sampled uniformly without replacement from S
  - conditioning on atomic sentences with B1, …, BK (e.g., ApparentPos(B1) = (9.6, 1.2, 32.8))

Existential Evidence versus Sampling in General

- Suppose you're in a wine shop, want to know whether it's fancy or not
- Existential evidence (not exhaustive!):
  $\exists \text{WineBottle}\, b (\text{In}(b, \text{ThisShop}) \wedge \text{Price}(b) = 40)$
- Evidence from sampling:

```
B() ~ UniformChoice[]({WineBottle b : In(b, ThisShop)});
Price(B) = 40;
```

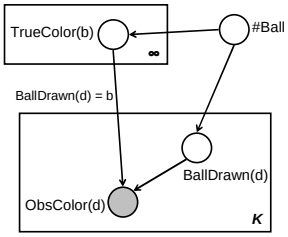
- Sampling $40 bottle is strong evidence that shop is fancy; knowing there is a $40 bottle tells you almost nothing

Skolemization in BLOG

- Only allow Skolemization for exhaustive observations
- Skolem constant introduction syntax:

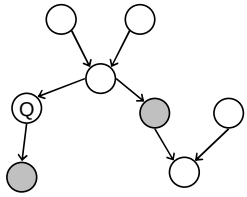
```
{RadarBlip b : BlipTime(b) = 8} = {B1, B2};
```

Sampling-Based Approximate Inference in BLOG



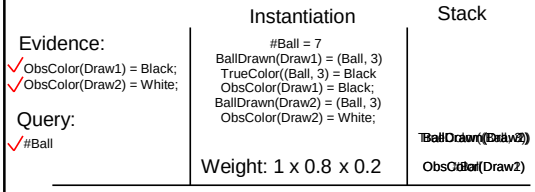
- Infinite CBN
- For inference, only need ancestors of query and evidence nodes
- But until we condition on BallDrawn(d), ObsColor(d) has infinitely many parents
- Solution: interleave sampling and relevance determination

Likelihood Weighting (LW)



- Sample non-evidence nodes top-down
- Weight each sample by product of probabilities of evidence nodes given their parents
- Provably converges to correct posterior

Likelihood Weighting for BLOG

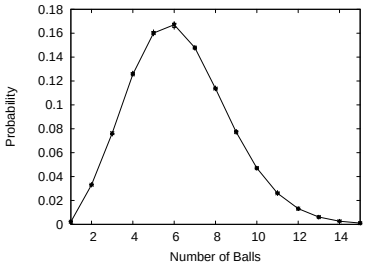



```
#Ball: () -> () ~ Poisson();
TrueColor(b) ~ TabularCPD();
BallDrawn(d) ~ UniformChoice({Ball b});
ObsColor(d)
    if !(BallDrawn(d) = null) then
        ~ TabularCPD(TrueColor(BallDrawn(d)));
```

Algorithm Correctness

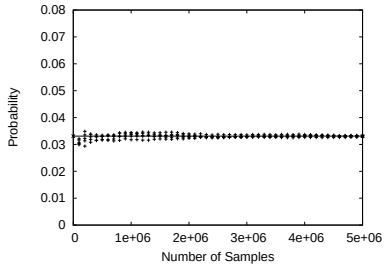
- Thm: If the BLOG model satisfies the finiteness and acyclicity conditions that guarantee it's well-defined, then:
  - LW algorithm generates each sample in finite time
  - Algorithm output converges to posterior defined by model as num samples $\rightarrow \infty$
- Holds even if infinitely many variables

Experiment: Approximating Posterior over #Ball



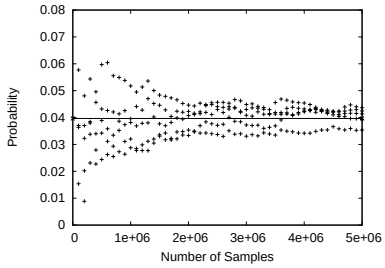
Given 10 draws, half black, half white
Results from 5 runs of 5,000,000 samples

Convergence Rate



P(#Ball = 2 | observations)

Convergence with Deterministic Observations



P(#Ball = 2 | observations)

Current Work

- Application: Building bibliographic database with human-level accuracy
  - Represent authors, topics, venues
- Inference algorithm that operates on objects, not just variables
  - Based on MCMC
  - Provide framework for hand-designed proposal distributions [Pasula et al. 2003]