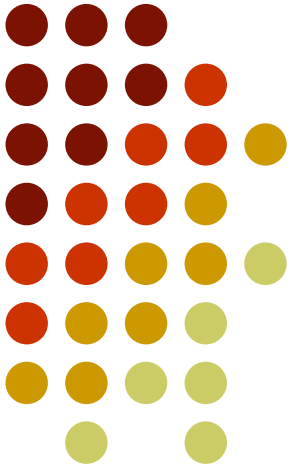
Unifying Logical and Statistical AI

Pedro Domingos
Dept. of Computer Science & Eng.
University of Washington

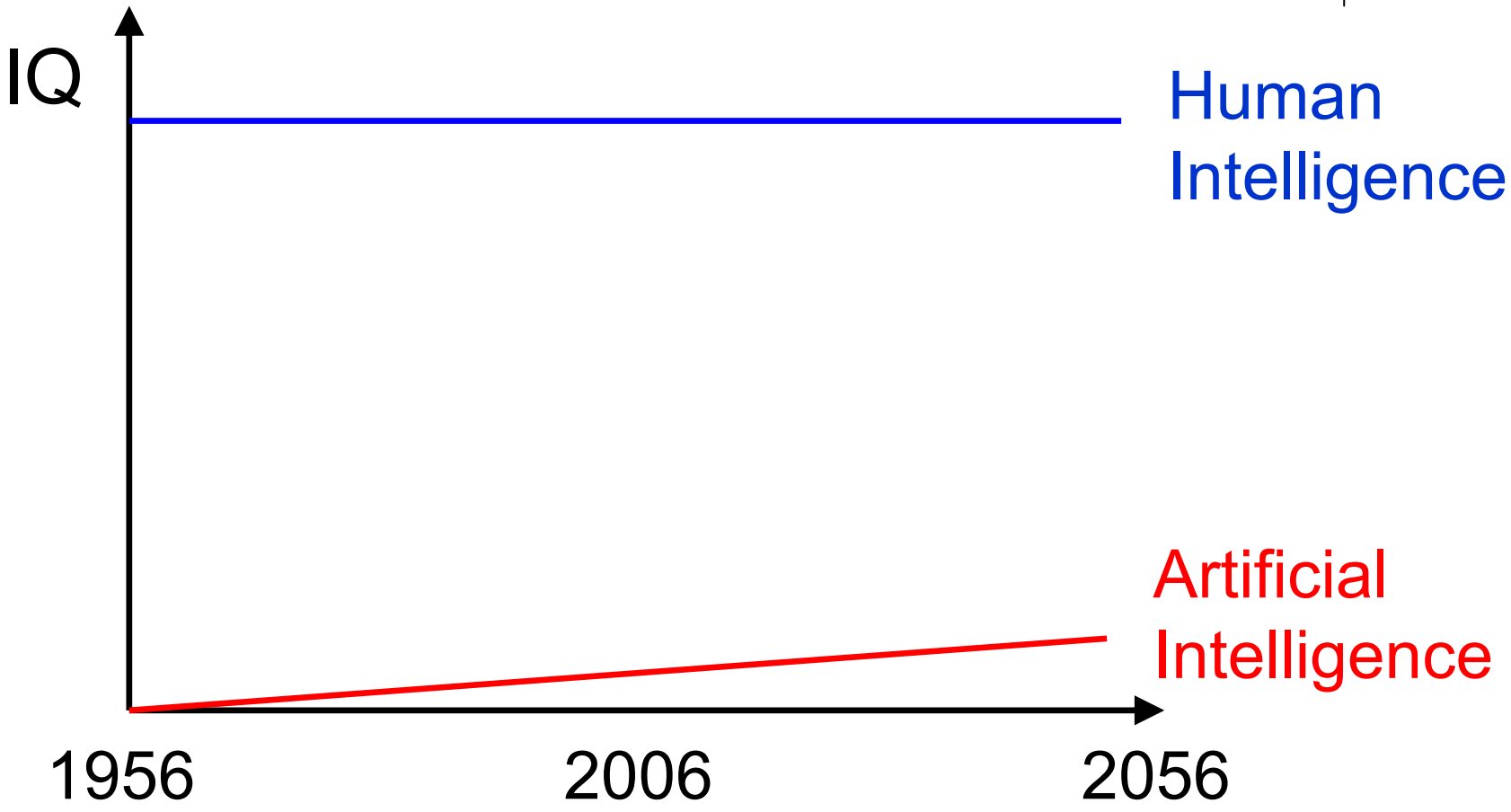


Joint work with Jesse Davis, Stanley Kok, Daniel Lowd, Aniruddh Nath, Hoifung Poon, Matt Richardson, Parag Singla, Marc Sumner, and Jue Wang

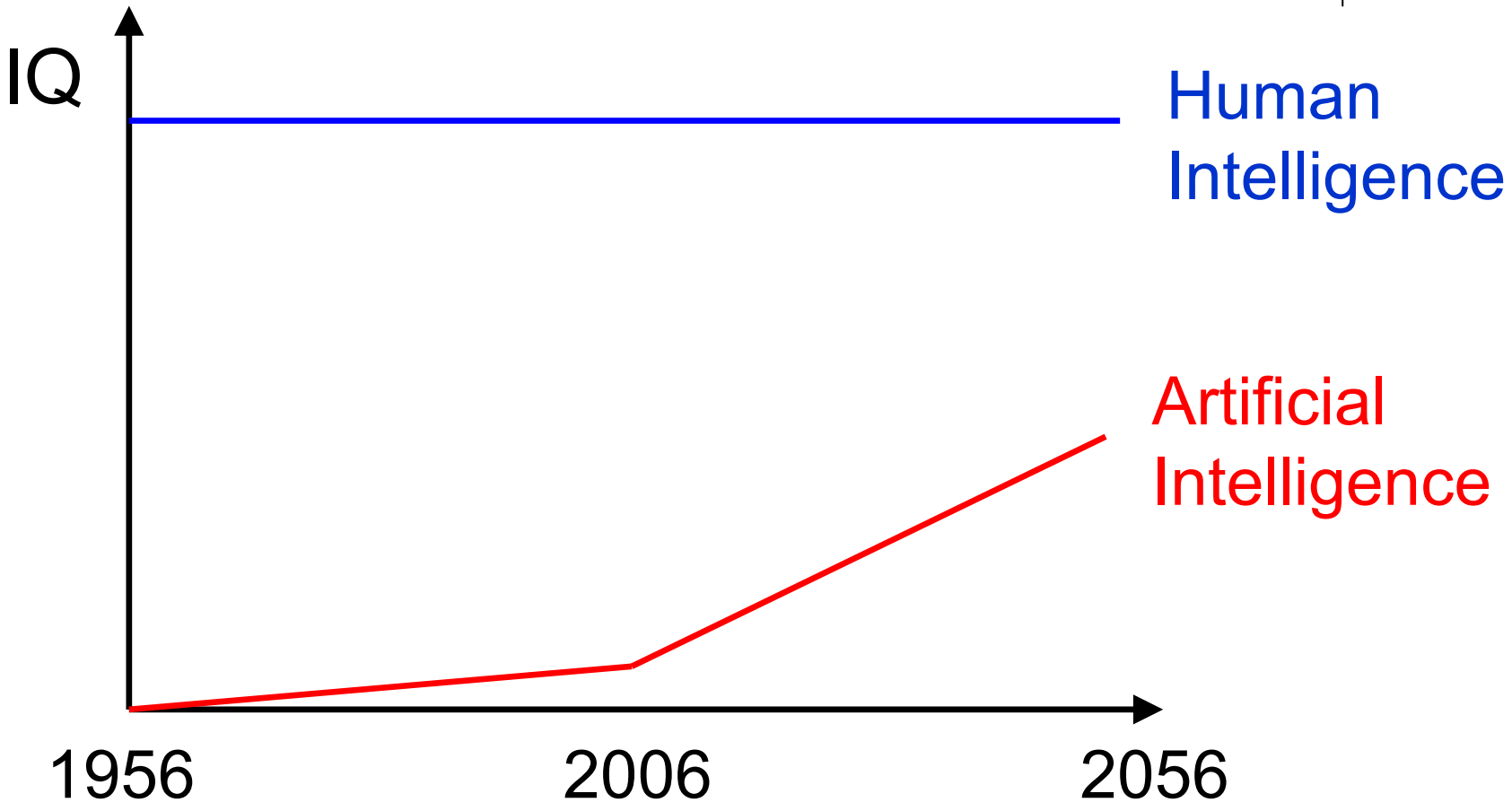
Overview

- **Motivation**
- Background
- Markov logic
- Inference
- Learning
- Software
- Applications
- Discussion

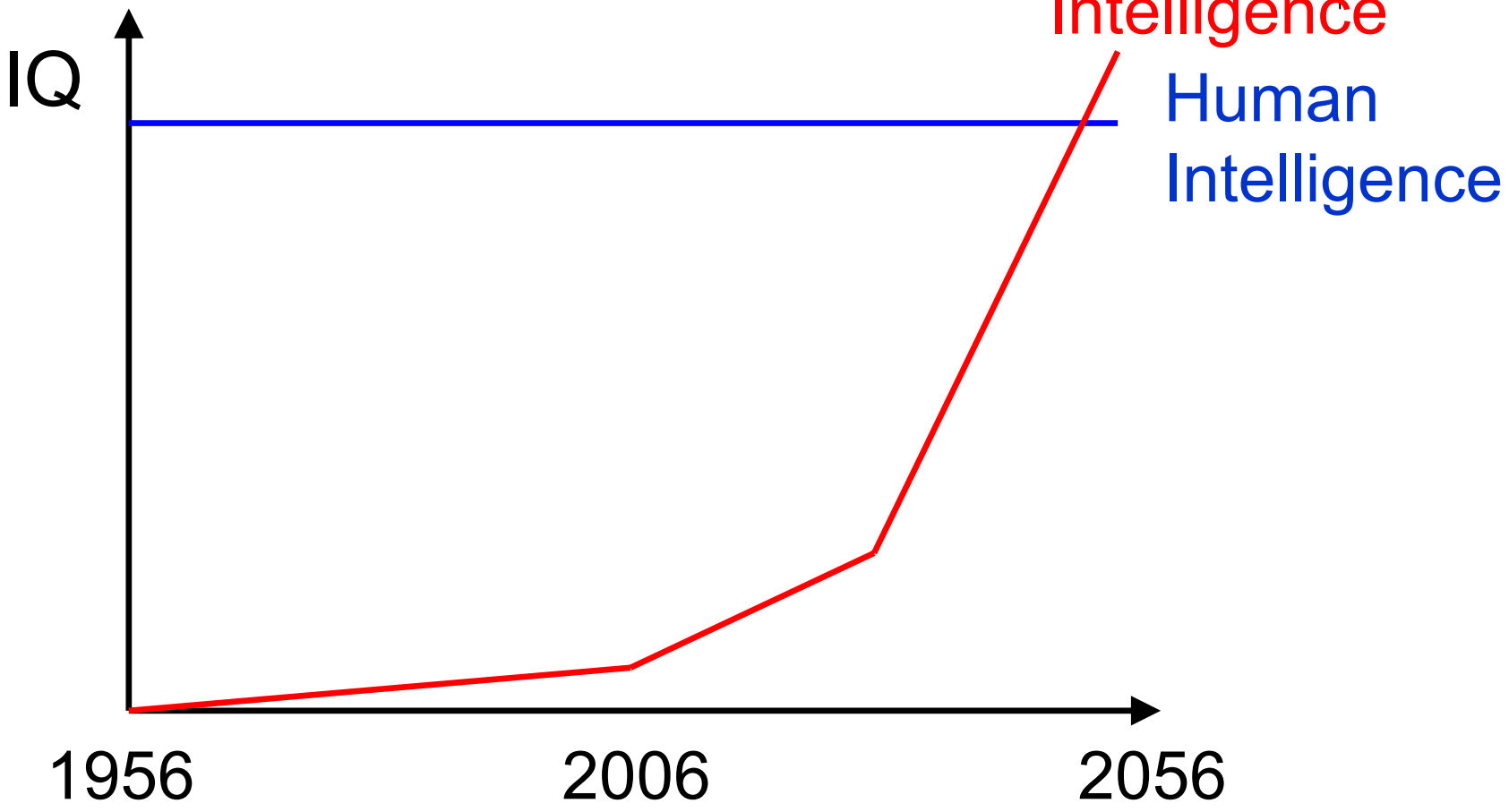
AI: The First 100 Years



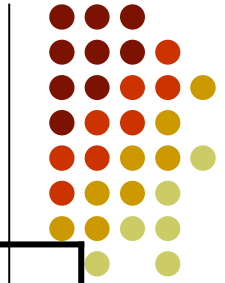
AI: The First 100 Years



AI: The First 100 Years



The Great AI Schism



Field	Logical approach	Statistical approach
Knowledge representation	First-order logic	Graphical models
Automated reasoning	Satisfiability testing	Markov chain Monte Carlo
Machine learning	Inductive logic programming	Neural networks
Planning	Classical planning	Markov decision processes
Natural language processing	Definite clause grammars	Prob. context-free grammars

We Need to Unify the Two

- The real world is complex and uncertain
- Logic handles complexity
- Probability handles uncertainty

Progress to Date

- Probabilistic logic [Nilsson, 1986]
- Statistics and beliefs [Halpern, 1990]
- Knowledge-based model construction [Wellman et al., 1992]
- Stochastic logic programs [Muggleton, 1996]
- Probabilistic relational models [Friedman et al., 1999]
- Relational Markov networks [Taskar et al., 2002]
- Etc.
- **This talk: Markov logic** [Richardson & Domingos, 2004]

Markov Logic

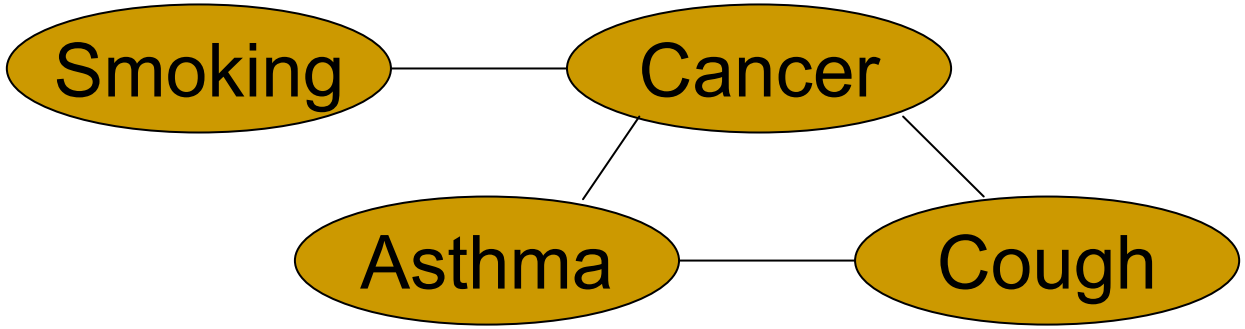
- **Syntax:** Weighted first-order formulas
- **Semantics:** Templates for Markov nets
- **Inference:** Lifted belief propagation, etc.
- **Learning:** Voted perceptron, pseudo-likelihood, inductive logic programming
- **Software:** Alchemy
- **Applications:** Information extraction, NLP, social networks, comp bio, etc.

Overview

- Motivation
- **Background**
- Markov logic
- Inference
- Learning
- Software
- Applications
- Discussion

Markov Networks

- **Undirected** graphical models



- Potential functions defined over cliques

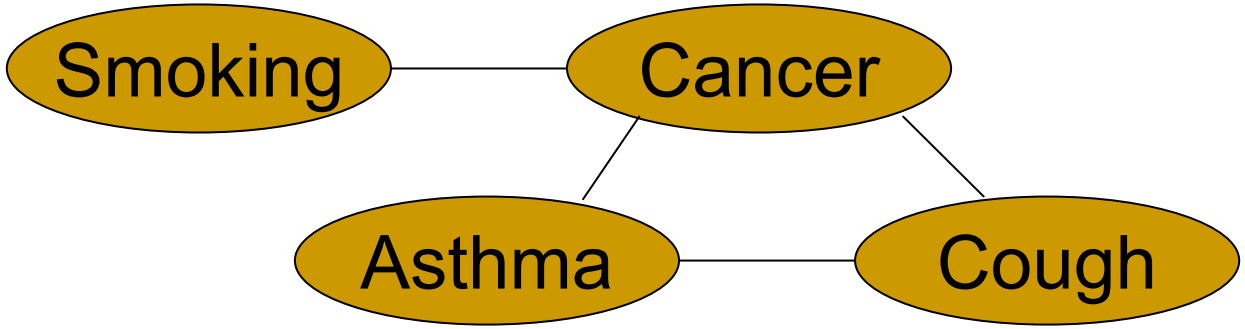
$$P(x) = \frac{1}{Z}\prod_{c}\Phi_c(x_c)$$

$$Z = \sum_{x}\prod_{c}\Phi_c(x_c)$$

Smoking	Cancer	Φ(S,C)
False	False	4.5
False	True	4.5
True	False	2.7
True	True	4.5

Markov Networks

- **Undirected** graphical models



- Log-linear model:

$$P(x) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(x) \right)$$

Weight of Feature *i*

Feature *i*

$$f_1(\text{Smoking}, \text{Cancer}) = \begin{cases} 1 & \text{if } \neg \text{Smoking} \vee \text{Cancer} \\ 0 & \text{otherwise} \end{cases}$$

$$w_1 = 0.51$$

First-Order Logic

- Constants, variables, functions, predicates
  E.g.: Anna, x, MotherOf(x), Friends(x,y)
- Grounding: Replace all variables by constants
  E.g.: Friends (Anna, Bob)
- **World** (model, interpretation):
  Assignment of truth values to all ground predicates

Overview

- Motivation
- Background
- **Markov logic**
- Inference
- Learning
- Software
- Applications
- Discussion

Markov Logic

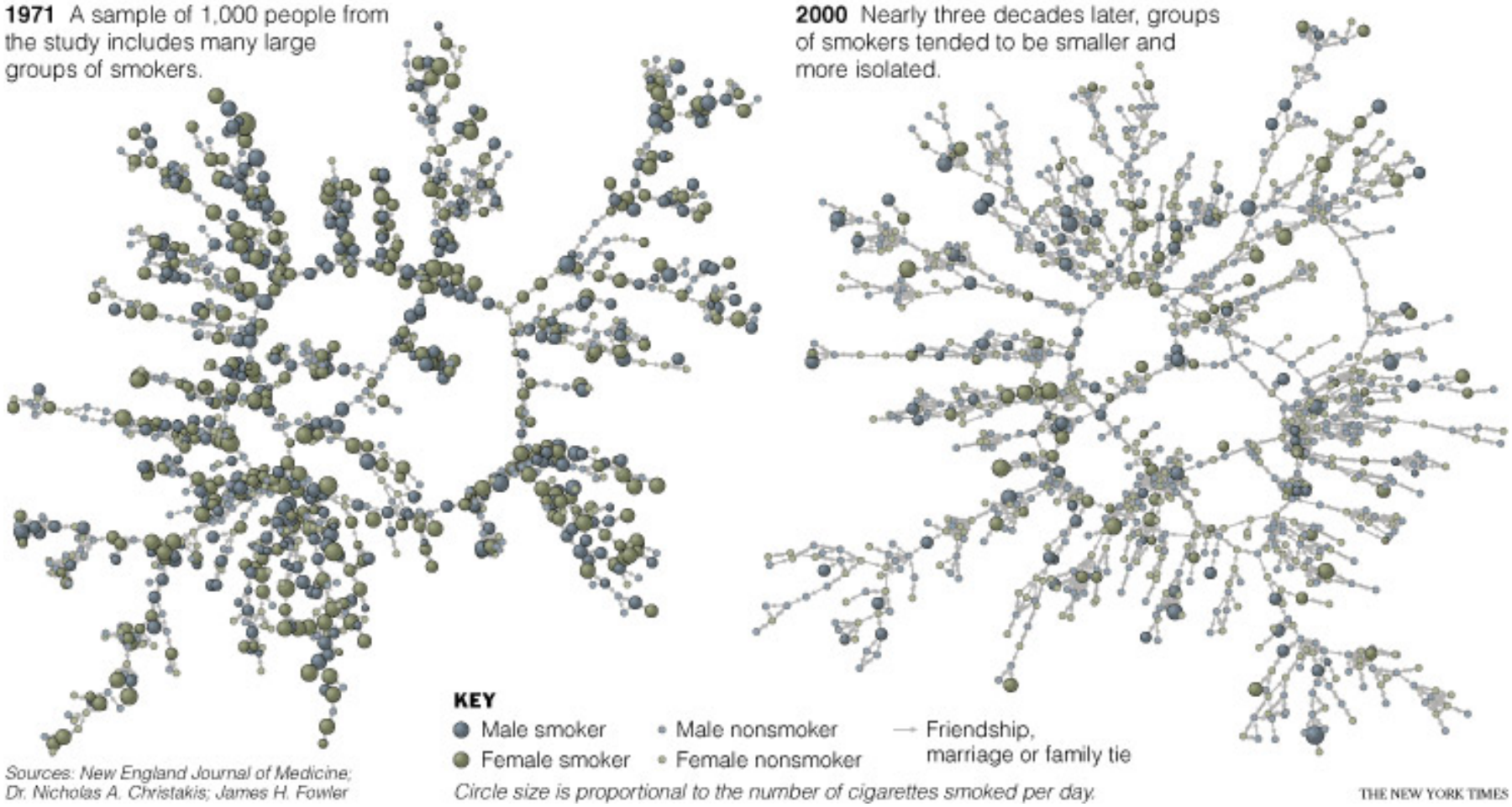
- A logical KB is a set of **hard constraints** on the set of possible worlds
- Let's make them **soft constraints**:
  When a world violates a formula,
  It becomes less probable, not impossible
- Give each formula a **weight**
  (Higher weight $\Rightarrow$ Stronger constraint)

$$P(\text{world}) \propto \exp\left(\sum \text{weights of formulas it satisfies}\right)$$

Definition

- A Markov Logic Network (MLN) is a set of pairs (F, w) where
  - F is a formula in first-order logic
  - w is a real number
- Together with a set of constants, it defines a Markov network with
  - One node for each grounding of each predicate in the MLN
  - One feature for each grounding of each formula F in the MLN, with the corresponding weight w

Example: Friends & Smokers



Example: Friends & Smokers

Smoking causes cancer.

Friends have similar smoking habits.

Example: Friends & Smokers

$$\forall x\ Smokes(x) \Rightarrow Cancer(x)$$

$$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$$

Example: Friends & Smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

Example: Friends & Smokers

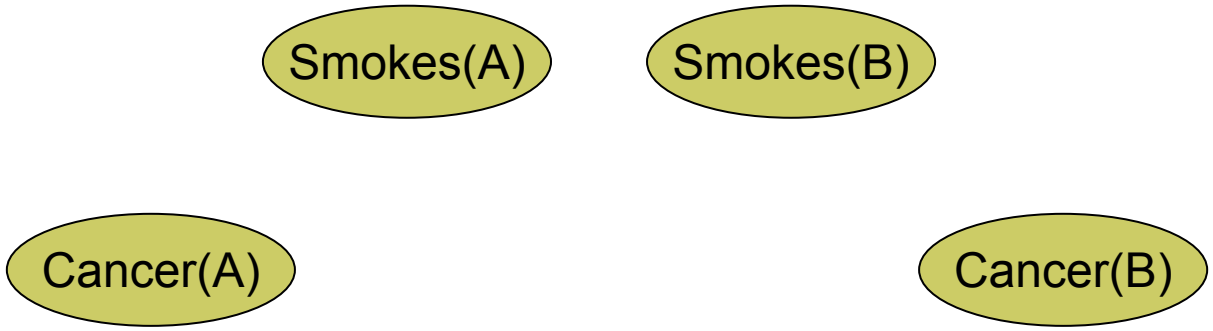
1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

Two constants: **Anna** (A) and **Bob** (B)

Example: Friends & Smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

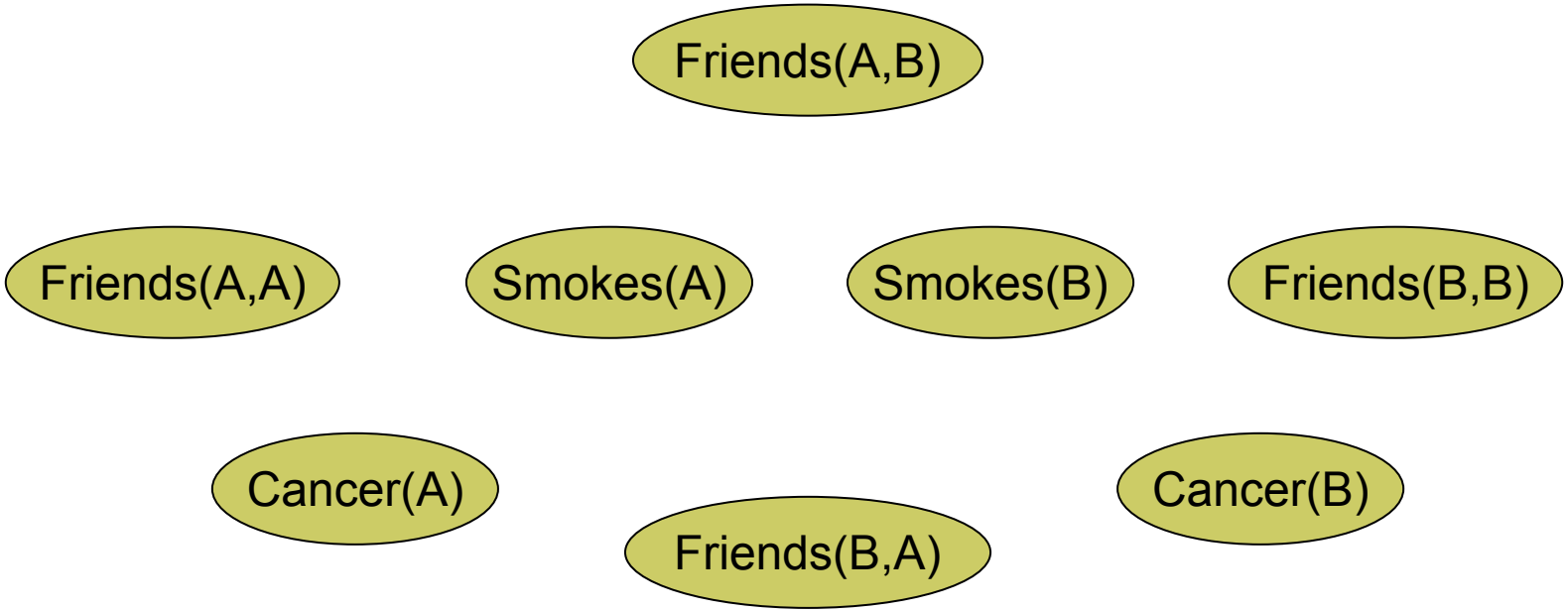
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

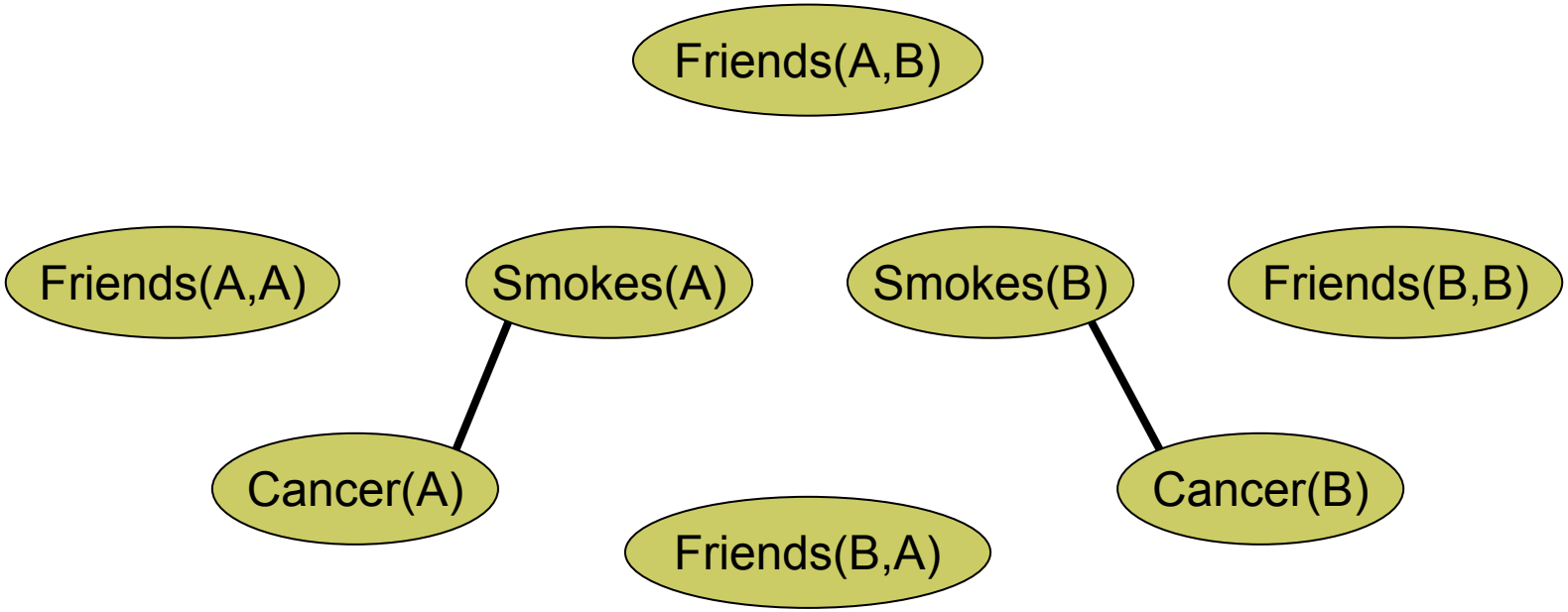
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

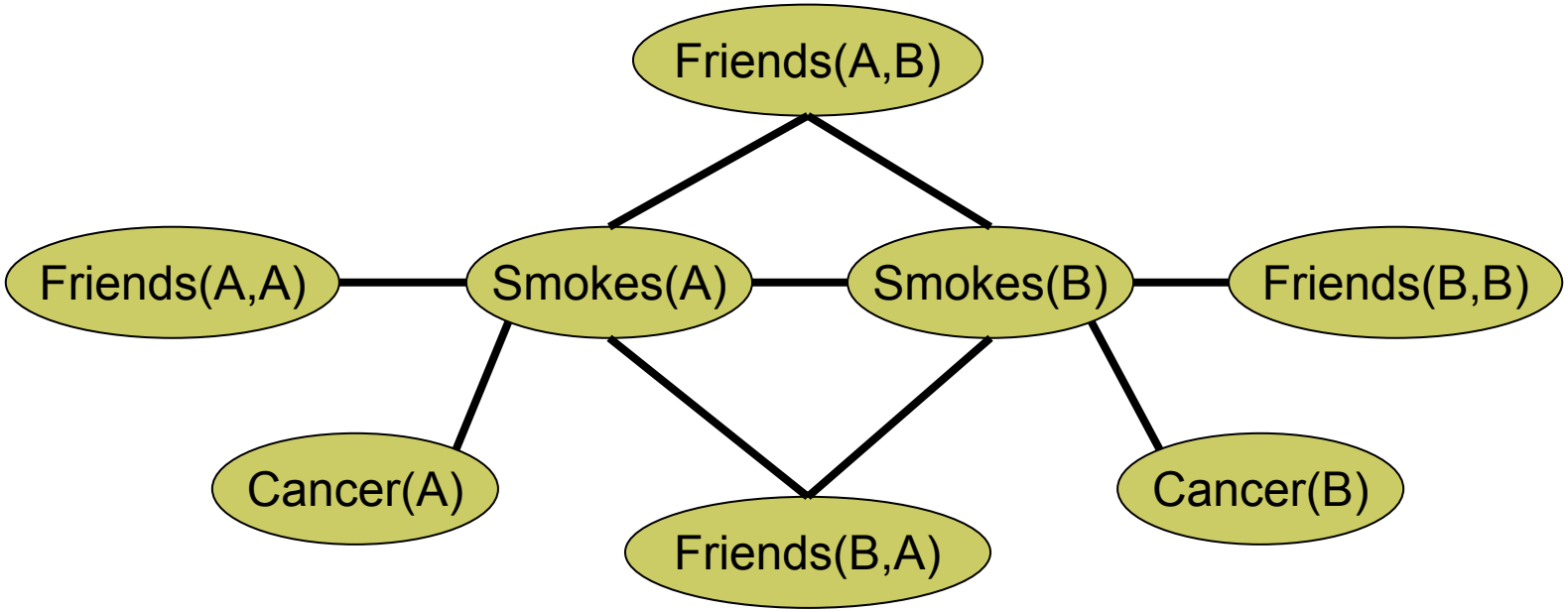
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

Two constants: **Anna** (A) and **Bob** (B)



Markov Logic Networks

- MLN is **template** for ground Markov nets
- Probability of a world x:

$$P(x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x) \right)$$

Weight of formula i

No. of true groundings of formula i in x

- **Typed** variables and constants greatly reduce size of ground Markov net
- Functions, existential quantifiers, etc.
- Infinite and continuous domains

Relation to Statistical Models

- Special cases:
  - Markov networks
  - Markov random fields
  - Bayesian networks
  - Log-linear models
  - Exponential models
  - Max. entropy models
  - Gibbs distributions
  - Boltzmann machines
  - Logistic regression
  - Hidden Markov models
  - Conditional random fields
- Obtained by making all predicates zero-arity
- Markov logic allows objects to be interdependent (non-i.i.d.)

Relation to First-Order Logic

- Infinite weights $\Rightarrow$ First-order logic
- Satisfiable KB, positive weights $\Rightarrow$ Satisfying assignments = Modes of distribution
- Markov logic allows contradictions between formulas

Overview

- Motivation
- Background
- Markov logic
- **Inference**
- Learning
- Software
- Applications
- Discussion

Inference

- MAP/MPE state
  - MaxWalkSAT
  - LazySAT
- Marginal and conditional probabilities
  - MCMC: Gibbs, MC-SAT, etc.
  - Knowledge-based model construction
  - Lifted belief propagation

Inference

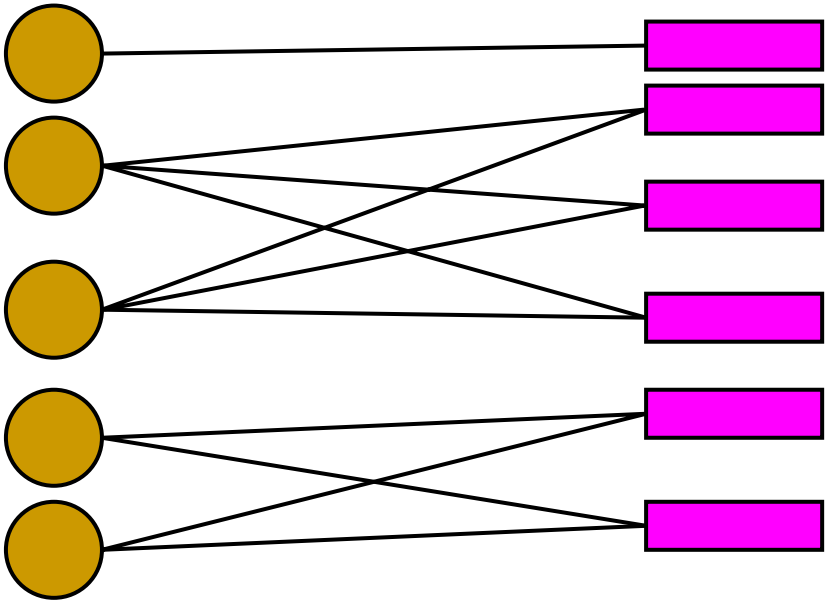
- MAP/MPE state
  - MaxWalkSAT
  - LazySAT
- Marginal and conditional probabilities
  - MCMC: Gibbs, MC-SAT, etc.
  - Knowledge-based model construction
  - **Lifted belief propagation**

Lifted Inference

- We can do inference in first-order logic without grounding the KB (e.g.: resolution)
- Let's do the same for inference in MLNs
- Group atoms and clauses into "indistinguishable" sets
- Do inference over those
- First approach: Lifted variable elimination (not practical)
- Here: Lifted belief propagation

Belief Propagation

$$\mu_{x \to f}(x) = \prod_{h \in n(x) \setminus \{f\}} \mu_{h \to x}(x)$$



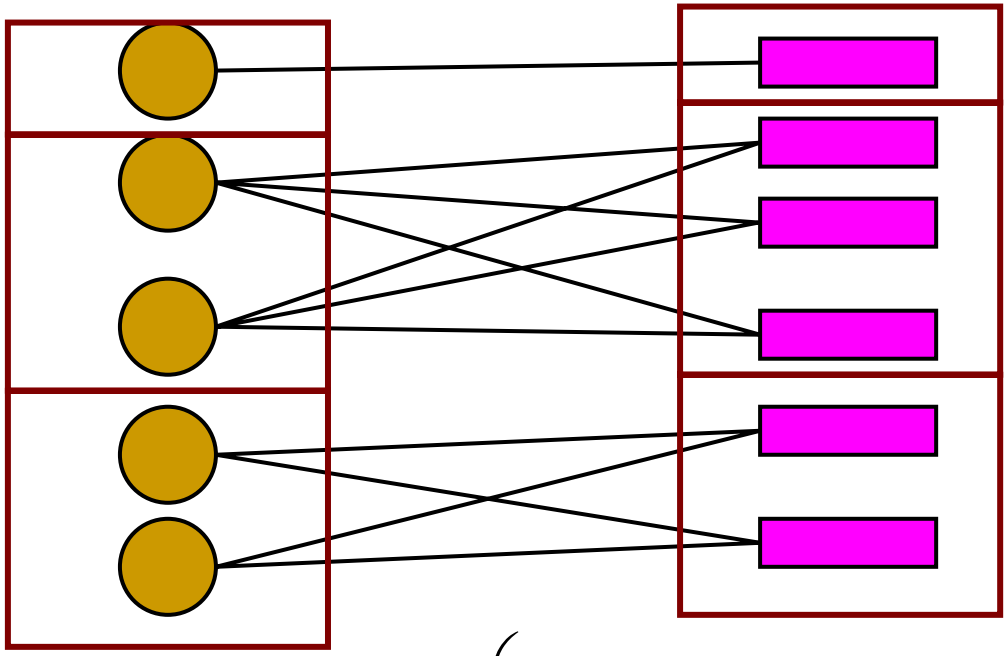
Nodes (x)

Features (f)

$$\mu_{f \to x}(x) = \sum_{\sim\{x\}} \left(e^{wf(x)} \prod_{y \in n(f) \setminus \{x\}} \mu_{y \to f}(y) \right)$$

Lifted Belief Propagation

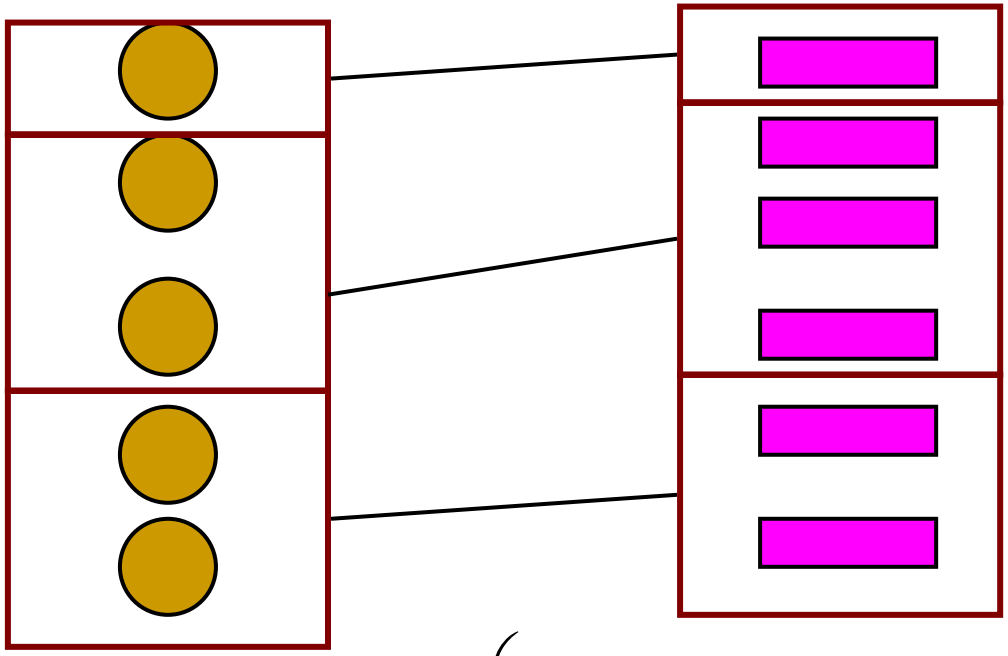
$$\mu_{x \to f}(x) = \prod_{h \in n(x) \setminus \{f\}} \mu_{h \to x}(x)$$



$$\mu_{f \to x}(x) = \sum_{\sim\{x\}} \left(e^{wf(x)} \prod_{y \in n(f) \setminus \{x\}} \mu_{y \to f}(y) \right)$$

Lifted Belief Propagation

$$\mu_{x \to f}(x) = \prod_{h \in n(x) \setminus \{f\}} \mu_{h \to x}(x)$$



$$\mu_{f \to x}(x) = \sum_{\sim\{x\}} \left(e^{wf(x)} \prod_{y \in n(f) \setminus \{x\}} \mu_{y \to f}(y) \right)$$

Lifted Belief Propagation

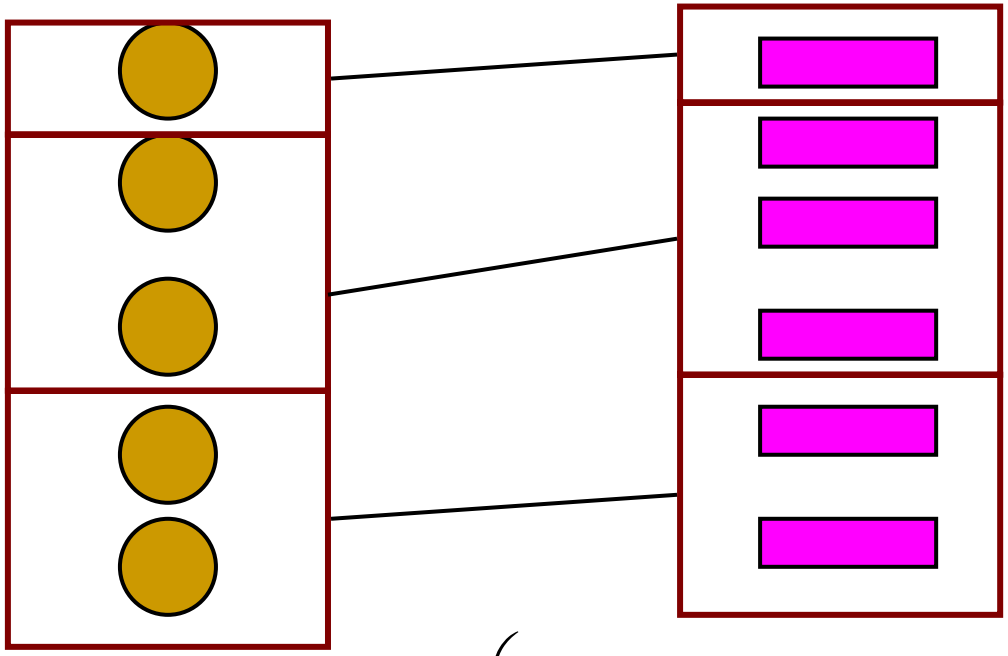
α,β :

Functions of edge counts

α

$$\mu_{x\to f}(x) = \beta \prod_{h\in n(x)\setminus\{f\}} \mu_{h\to x}(x)$$

Nodes (x)



Features (f)

$$\mu_{f\to x}(x) = \sum_{\sim\{x\}} \left(e^{wf(x)} \prod_{y\in n(f)\setminus\{x\}} \mu_{y\to f}(y) \right)$$

Lifted Belief Propagation

- Form **lifted network** composed of supernodes and superfeatures
  - **Supernode:** Set of ground atoms that all send and receive same messages throughout BP
  - **Superfeature:** Set of ground clauses that all send and receive same messages throughout BP
- Run belief propagation on lifted network
- Guaranteed to produce same results as ground BP
- Time and memory savings can be huge

Forming the Lifted Network

1. **Form initial supernodes**
   One per predicate and truth value
   (true, false, unknown)
2. **Form superfeatures by doing joins of their supernodes**
3. **Form supernodes by projecting superfeatures down to their predicates**
   Supernode = Groundings of a predicate with same number of projections from each superfeature
4. **Repeat until convergence**

Theorem

- There exists a unique minimal lifted network
- The lifted network construction algo. finds it
- BP on lifted network gives same result as on ground network

Representing Supernodes And Superfeatures

- List of tuples: Simple but inefficient
- Resolution-like: Use equality and inequality
- Form clusters (in progress)

Open Questions

- Can we do approximate KBMC/lazy/lifting?
- Can KBMC, lazy and lifted inference be combined?
- Can we have lifted inference over both probabilistic and deterministic dependencies? (Lifted MC-SAT?)
- Can we unify resolution and lifted BP?
- Can other inference algorithms be lifted?

Overview

- Motivation
- Background
- Markov logic
- Inference
- **Learning**
- Software
- Applications
- Discussion

Learning

- Data is a relational database
- Closed world assumption (if not: EM)
- Learning parameters (weights)
  - Generatively
  - Discriminatively
- Learning structure (formulas)

Generative Weight Learning

- Maximize likelihood
- Use gradient ascent or L-BFGS
- No local maxima

$$\frac{\partial}{\partial w_i} \log P_w(x) = n_i(x) - E_w[n_i(x)]$$

No. of true groundings of clause *i* in data

Expected no. true groundings according to model

- Requires inference at each step (slow!)

Pseudo-Likelihood

$$PL(x) \equiv \prod_i P(x_i \mid neighbors(x_i))$$

- Likelihood of each variable given its neighbors in the data [Besag, 1975]
- Does not require inference at each step
- Consistent estimator
- Widely used in vision, spatial statistics, etc.
- But PL parameters may not work well for long inference chains

Discriminative Weight Learning

- Maximize conditional likelihood of query (y) given evidence (x)

$$\frac{\partial}{\partial w_i} \log P_w(y \mid x) = n_i(x, y) - E_w\left[n_i(x, y)\right]$$

No. of true groundings of clause *i* in data

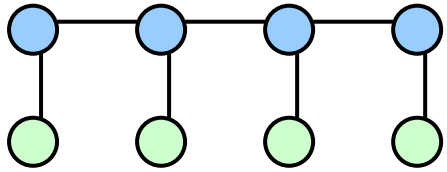
Expected no. true groundings according to model

- Approximate expected counts by counts in MAP state of y given x

Voted Perceptron

- Originally proposed for training HMMs discriminatively [Collins, 2002]
- Assumes network is linear chain

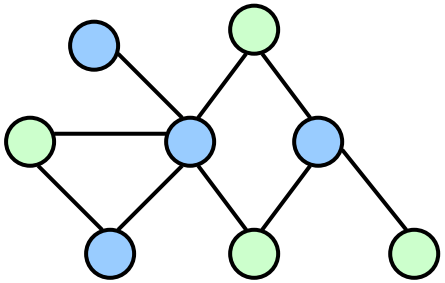
$w_i \leftarrow 0$
for $t \leftarrow 1$ **to** T **do**
    $y_{MAP} \leftarrow \text{Viterbi}(x)$
    $w_i \leftarrow w_i + \eta\,[\text{count}_i(y_{Data}) - \text{count}_i(y_{MAP})]$
return $\sum_t w_i / \text{T}$



Voted Perceptron for MLNs

- HMMs are special case of MLNs
- Replace Viterbi by MaxWalkSAT
- Network can now be arbitrary graph

$w_i \leftarrow 0$
for $t \leftarrow 1$ **to** T **do**
  $y_{MAP} \leftarrow$ MaxWalkSAT(x)
  $w_i \leftarrow w_i + \eta\,[\text{count}_i(y_{Data}) - \text{count}_i(y_{MAP})]$
return $\sum_t w_i / \text{T}$



Structure Learning

- Generalizes feature induction in Markov nets
- Any inductive logic programming approach can be used, but . . .
- Goal is to induce any clauses, not just Horn
- Evaluation function should be likelihood
- Requires learning weights for each candidate
- Turns out not to be bottleneck
- Bottleneck is counting clause groundings
- Solution: Subsampling

Structure Learning

- **Initial state:** Unit clauses or hand-coded KB
- **Operators:** Add/remove literal, flip sign
- **Evaluation function:**
  Pseudo-likelihood + Structure prior
- **Search:**
  - Beam [Kok & Domingos, 2005]
  - Shortest-first [Kok & Domingos, 2005]
  - Bottom-up [Mihalkova & Mooney, 2007]

Overview

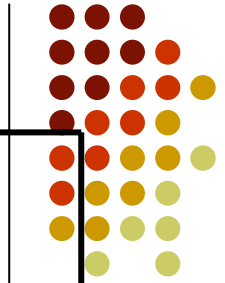
- Motivation
- Background
- Markov logic
- Inference
- Learning
- **Software**
- Applications
- Discussion

Alchemy

Open-source software including:

- Full first-order logic syntax
- MAP and marginal/conditional inference
- Generative & discriminative weight learning
- Structure learning
- Programming language features

alchemy.cs.washington.edu



	Alchemy	**Prolog**	**BUGS**
Represent-ation	F.O. Logic + Markov nets	Horn clauses	Bayes nets
Inference	Lifted BP, etc.	Theorem proving	Gibbs sampling
Learning	Parameters & structure	No	Params.
Uncertainty	Yes	No	Yes
Relational	Yes	Yes	No

Overview

- Motivation
- Background
- Markov logic
- Inference
- Learning
- Software
- **Applications**
- Discussion

Applications

- Information extraction
- Entity resolution
- Link prediction
- Collective classification
- Web mining
- Natural language processing
- Computational biology
- Social network analysis
- Robot mapping
- Activity recognition
- Probabilistic Cyc
- CALO
- Etc.

Information Extraction

Parag Singla and Pedro Domingos, “Memory-Efficient Inference in Relational Domains” (AAAI-06).

Singla, P., & Domingos, P. (2006). Memory-efficent inference in relatonal domains. In Proceedings of the Twenty-First National Conference on Artificial Intelligence (pp. 500-505). Boston, MA: AAAI Press.

H. Poon & P. Domingos, Sound and Efficient Inference with Probabilistic and Deterministic Dependencies”, in Proc. AAAI-06, Boston, MA, 2006.

P. Hoifung (2006). Efficent inference. In Proceedings of the Twenty-First National Conference on Artificial Intelligence.

Segmentation

Author
Title
Venue

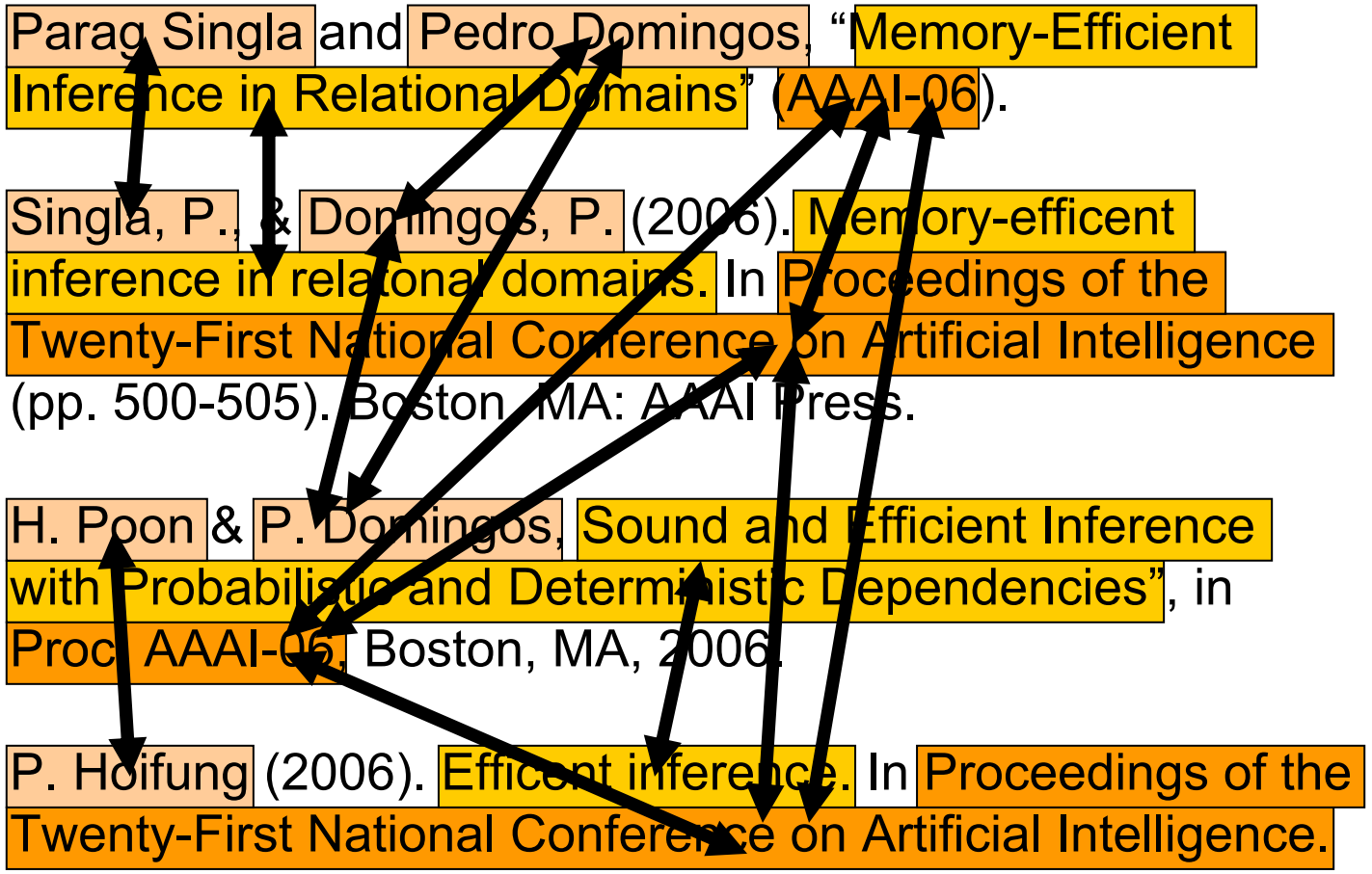
Parag Singla and Pedro Domingos, "Memory-Efficient Inference in Relational Domains" (AAAI-06).

Singla, P., & Domingos, P. (2006). Memory-efficent inference in relatonal domains. In Proceedings of the Twenty-First National Conference on Artificial Intelligence (pp. 500-505). Boston, MA: AAAI Press.

H. Poon & P. Domingos, Sound and Efficient Inference with Probabilistic and Deterministic Dependencies", in Proc. AAAI-06, Boston, MA, 2006.

P. Hoifung (2006). Efficent inference. In Proceedings of the Twenty-First National Conference on Artificial Intelligence.

Entity Resolution



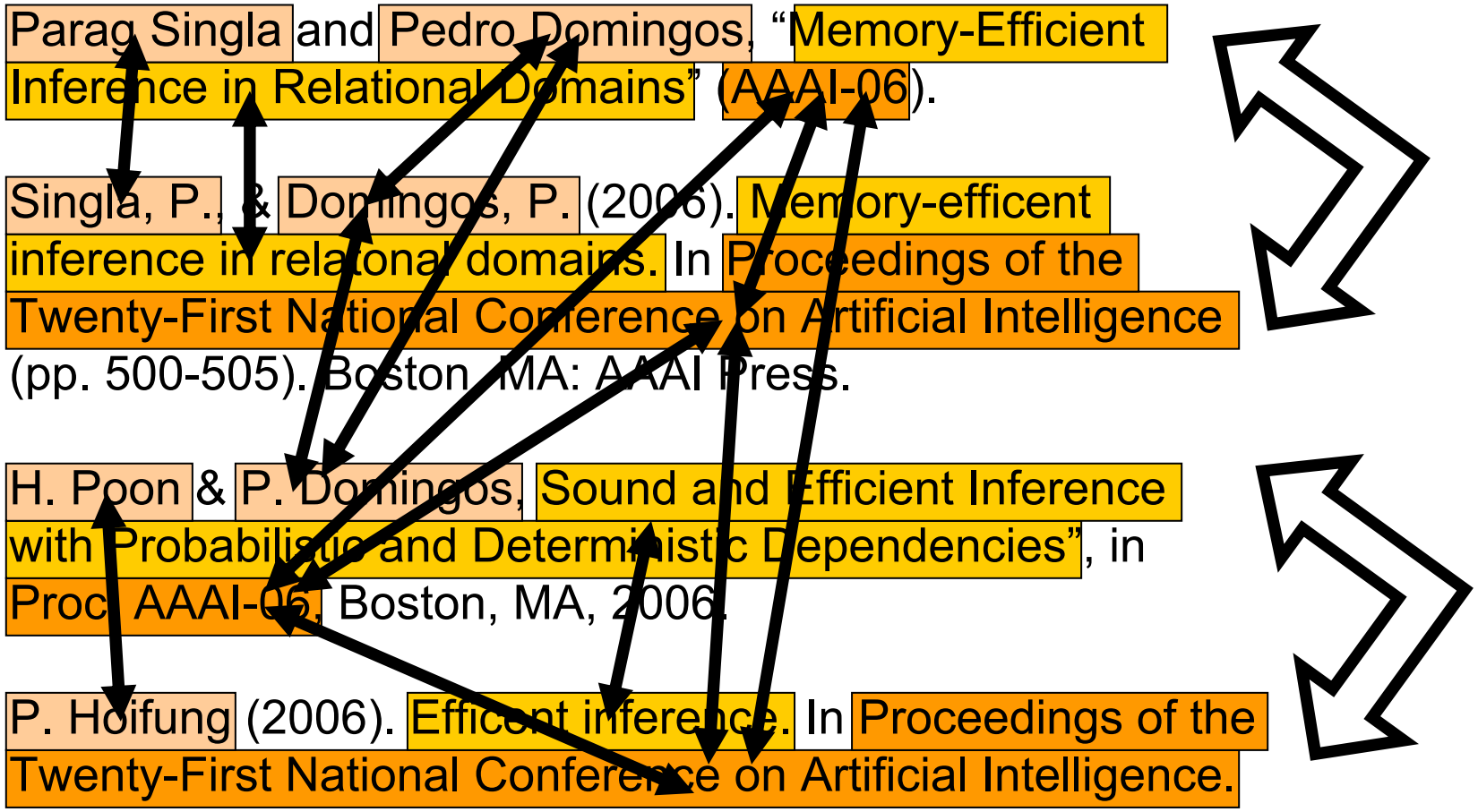
Parag Singla and Pedro Domingos, "Memory-Efficient Inference in Relational Domains" (AAAI-06).

Singla, P., & Domingos, P. (2006). Memory-efficent inference in relational domains. In Proceedings of the Twenty-First National Conference on Artificial Intelligence (pp. 500-505). Boston, MA: AAAI Press.

H. Poon & P. Domingos, Sound and Efficient Inference with Probabilistic and Deterministic Dependencies", in Proc. AAAI-06, Boston, MA, 2006.

P. Hoifung (2006). Efficent inference. In Proceedings of the Twenty-First National Conference on Artificial Intelligence.

Entity Resolution



State of the Art

- Segmentation
  - HMM (or CRF) to assign each token to a field
- Entity resolution
  - Logistic regression to predict same field/citation
  - Transitive closure
- Alchemy implementation: Seven formulas

Types and Predicates

```
token = {Parag, Singla, and, Pedro, ...}
field = {Author, Title, Venue}
citation = {C1, C2, ...}
position = {0, 1, 2, ...}

Token(token, position, citation)
InField(position, field, citation)
SameField(field, citation, citation)
SameCit(citation, citation)
```

Types and Predicates

```
token = {Parag, Singla, and, Pedro, ...}
field = {Author, Title, Venue, ...}
citation = {C1, C2, ...}
position = {0, 1, 2, ...}
```

Optional

```
Token(token, position, citation)
InField(position, field, citation)
SameField(field, citation, citation)
SameCit(citation, citation)
```

Types and Predicates

```
token = {Parag, Singla, and, Pedro, ...}
field = {Author, Title, Venue}
citation = {C1, C2, ...}
position = {0, 1, 2, ...}

Token(token, position, citation)
InField(position, field, citation)
SameField(field, citation, citation)
SameCit(citation, citation)
```

← Evidence

Types and Predicates

```
token = {Parag, Singla, and, Pedro, ...}
field = {Author, Title, Venue}
citation = {C1, C2, ...}
position = {0, 1, 2, ...}

Token(token, position, citation)
InField(position, field, citation)
SameField(field, citation, citation)
SameCit(citation, citation)
```

← Query

Formulas

```
Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f != f' => (!InField(i,+f,c) v !InField(i,+f',c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i',c')
   ^ InField(i',+f,c') => SameField(+f,c,c')
SameField(+f,c,c') <=> SameCit(c,c')
SameField(f,c,c') ^ SameField(f,c',c")
   => SameField(f,c,c")
SameCit(c,c') ^ SameCit(c',c") => SameCit(c,c")
```

Formulas

```
Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f != f' => (!InField(i,+f,c) v !InField(i,+f',c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i',c')
   ^ InField(i',+f,c') => SameField(+f,c,c')
SameField(+f,c,c') <=> SameCit(c,c')
SameField(f,c,c') ^ SameField(f,c',c")
   => SameField(f,c,c")
SameCit(c,c') ^ SameCit(c',c") => SameCit(c,c")
```

Formulas

```
Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f != f' => (!InField(i,+f,c) v !InField(i,+f',c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i',c')
   ^ InField(i',+f,c') => SameField(+f,c,c')
SameField(+f,c,c') <=> SameCit(c,c')
SameField(f,c,c') ^ SameField(f,c',c")
   => SameField(f,c,c")
SameCit(c,c') ^ SameCit(c',c") => SameCit(c,c")
```

Formulas

```
Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f != f' => (!InField(i,+f,c) v !InField(i,+f',c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i',c')
   ^ InField(i',+f,c') => SameField(+f,c,c')
SameField(+f,c,c') <=> SameCit(c,c')
SameField(f,c,c') ^ SameField(f,c',c")
   => SameField(f,c,c")
SameCit(c,c') ^ SameCit(c',c") => SameCit(c,c")
```

Formulas

```
Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f != f' => (!InField(i,+f,c) v !InField(i,+f',c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i',c')
   ^ InField(i',+f,c') => SameField(+f,c,c')
SameField(+f,c,c') <=> SameCit(c,c')
SameField(f,c,c') ^ SameField(f,c',c")
   => SameField(f,c,c")
SameCit(c,c') ^ SameCit(c',c") => SameCit(c,c")
```

Formulas

```
Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f != f' => (!InField(i,+f,c) v !InField(i,+f',c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i',c')
   ^ InField(i',+f,c') => SameField(+f,c,c')
SameField(+f,c,c') <=> SameCit(c,c')
SameField(f,c,c') ^ SameField(f,c',c")
   => SameField(f,c,c")
SameCit(c,c') ^ SameCit(c',c") => SameCit(c,c")
```

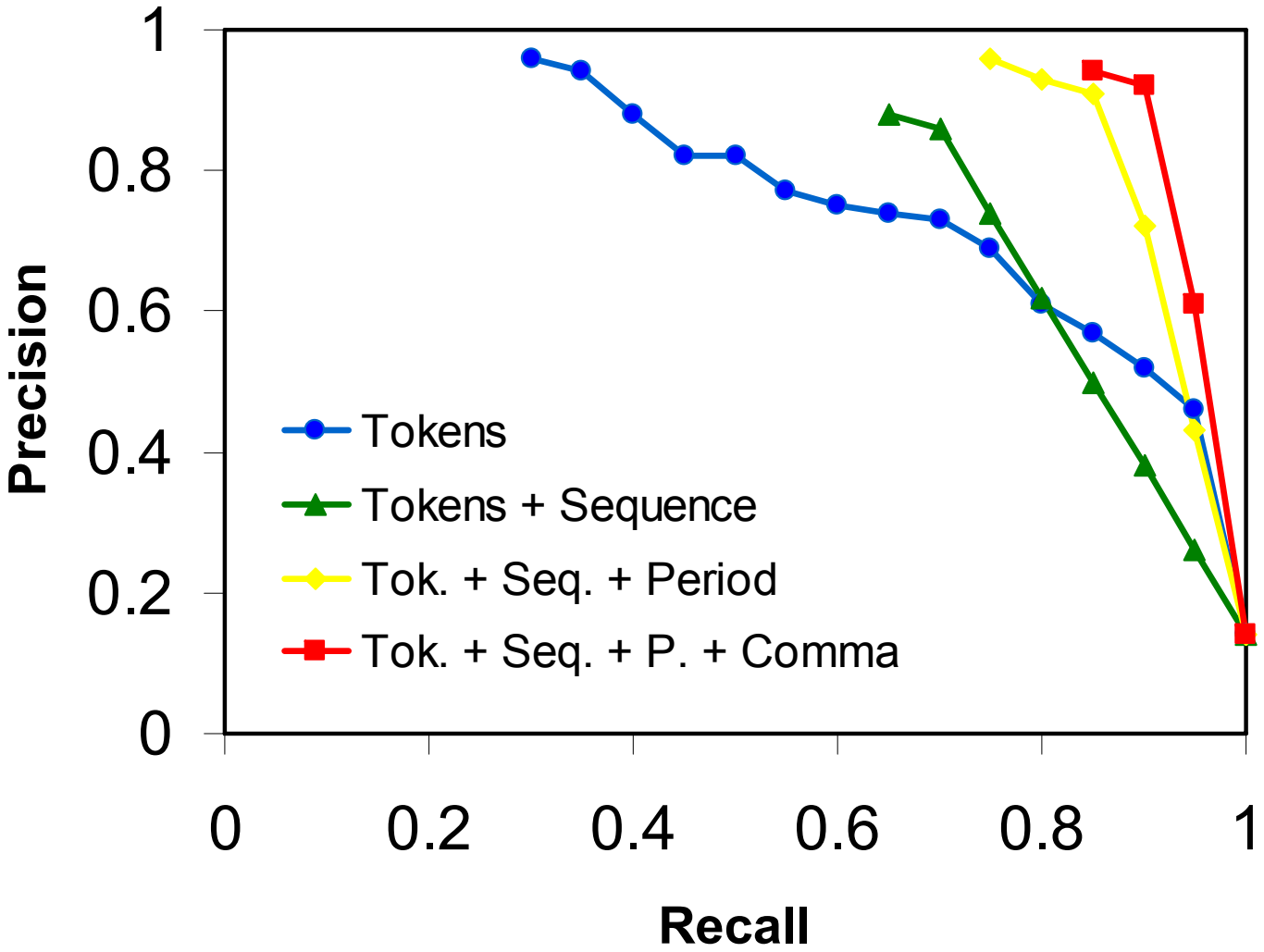
Formulas

```
Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f != f' => (!InField(i,+f,c) v !InField(i,+f',c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i',c')
   ^ InField(i',+f,c') => SameField(+f,c,c')
SameField(+f,c,c') <=> SameCit(c,c')
SameField(f,c,c') ^ SameField(f,c',c")
   => SameField(f,c,c")
SameCit(c,c') ^ SameCit(c',c") => SameCit(c,c")
```

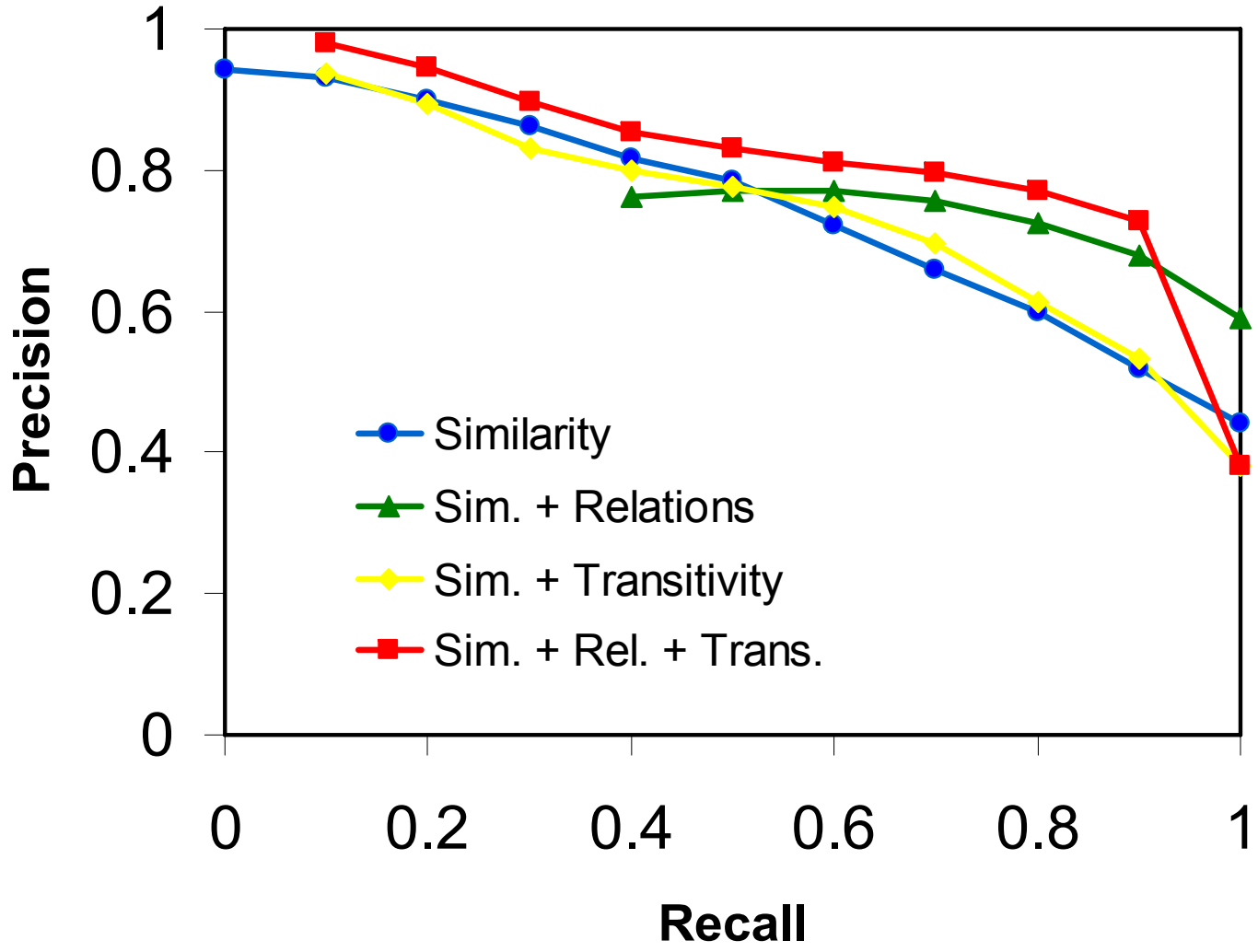
Formulas

```
Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) ^ !Token(“.”,i,c) <=> InField(i+1,+f,c)
f != f’ => (!InField(i,+f,c) v !InField(i,+f’,c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i’,c’)
   ^ InField(i’,+f,c’) => SameField(+f,c,c’)
SameField(+f,c,c’) <=> SameCit(c,c’)
SameField(f,c,c’) ^ SameField(f,c’,c”)
   => SameField(f,c,c”)
SameCit(c,c’) ^ SameCit(c’,c”) => SameCit(c,c”)
```

Results: Segmentation on Cora



Results: Matching Venues on Cora



Overview

- Motivation
- Background
- Markov logic
- Inference
- Learning
- Software
- Applications
- **Discussion**

The Interface Layer

Applications

Interface Layer

Infrastructure

Networking

WWW

Applications

Email

Interface Layer

Internet

Protocols

Infrastructure

Routers

Databases

ERP

Applications

CRM

OLTP

Interface Layer

Relational Model

Transaction Management

Infrastructure

Query Optimization

Programming Systems

Applications

Programming

Interface Layer

High-Level Languages

Infrastructure

Compilers

Code Optimizers

Artificial Intelligence

Planning

Robotics

Applications

NLP

Multi-Agent Systems

Vision

Interface Layer

Representation

Infrastructure

Inference

Learning

Artificial Intelligence

Planning

Robotics

Applications

NLP

Multi-Agent Systems

Vision

Interface Layer

First-Order Logic?

Representation

Infrastructure

Inference

Learning

Artificial Intelligence

Planning

Robotics

Applications

NLP

Multi-Agent Systems

Vision

Interface Layer

Graphical Models?

Representation

Infrastructure

Inference

Learning

Artificial Intelligence

Planning

Robotics

Applications

NLP

Multi-Agent Systems

Vision

Interface Layer

Markov Logic

Representation

Infrastructure

Inference

Learning

Artificial Intelligence

Planning

Robotics

Applications

NLP

Multi-Agent Systems

Vision

Alchemy: alchemy.cs.washington.edu

Representation

Infrastructure

Inference

Learning