

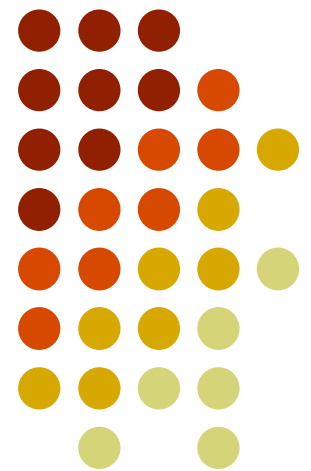
# Unifying Logical and Statistical AI

---

**Pedro Domingos**

Dept. of Computer Science & Eng.  
University of Washington

*Joint work with Stanley Kok, Hoifung Poon,  
Matt Richardson and Parag Singla*

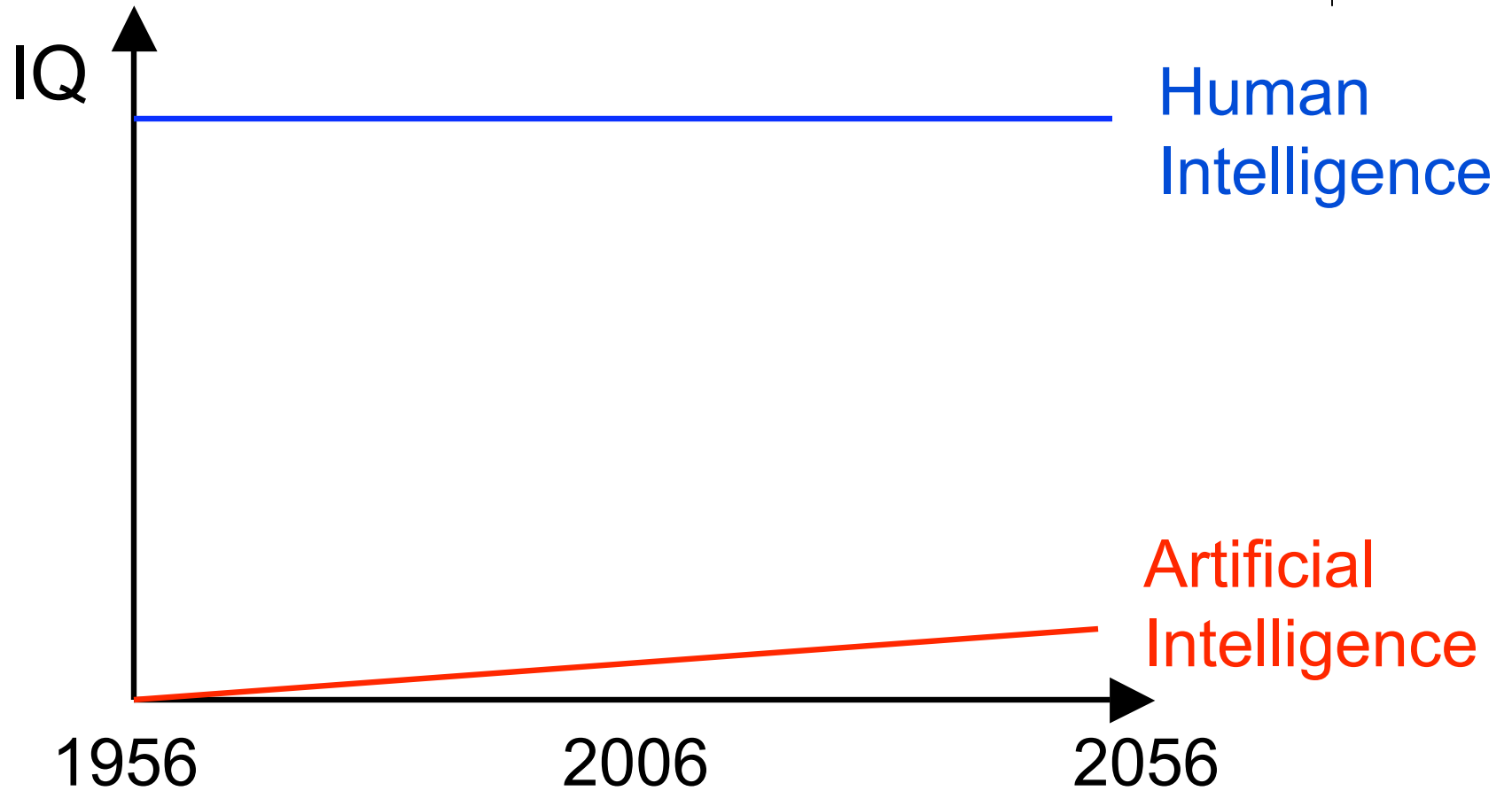
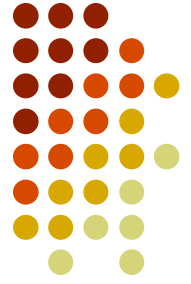




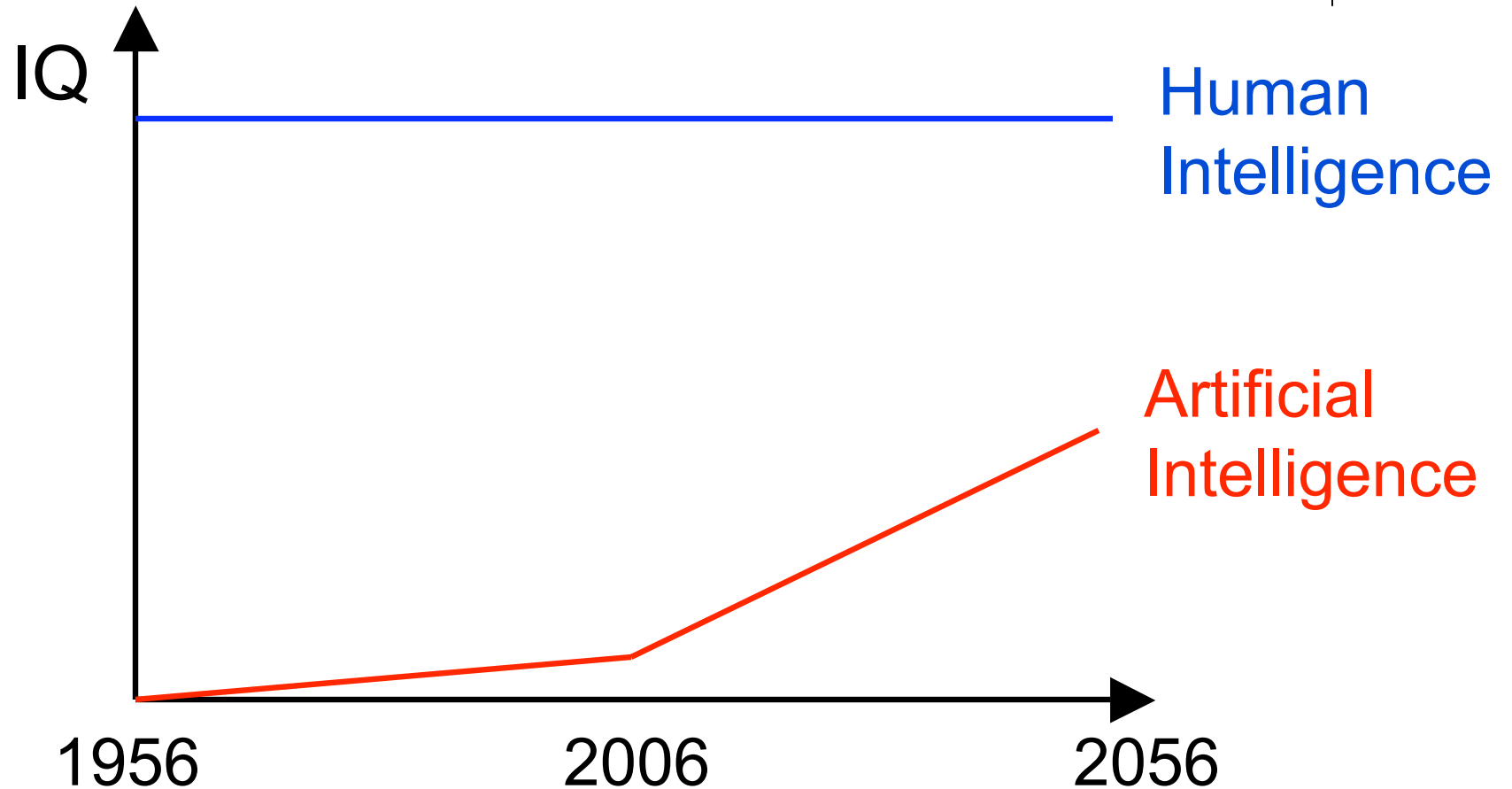
# Overview

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

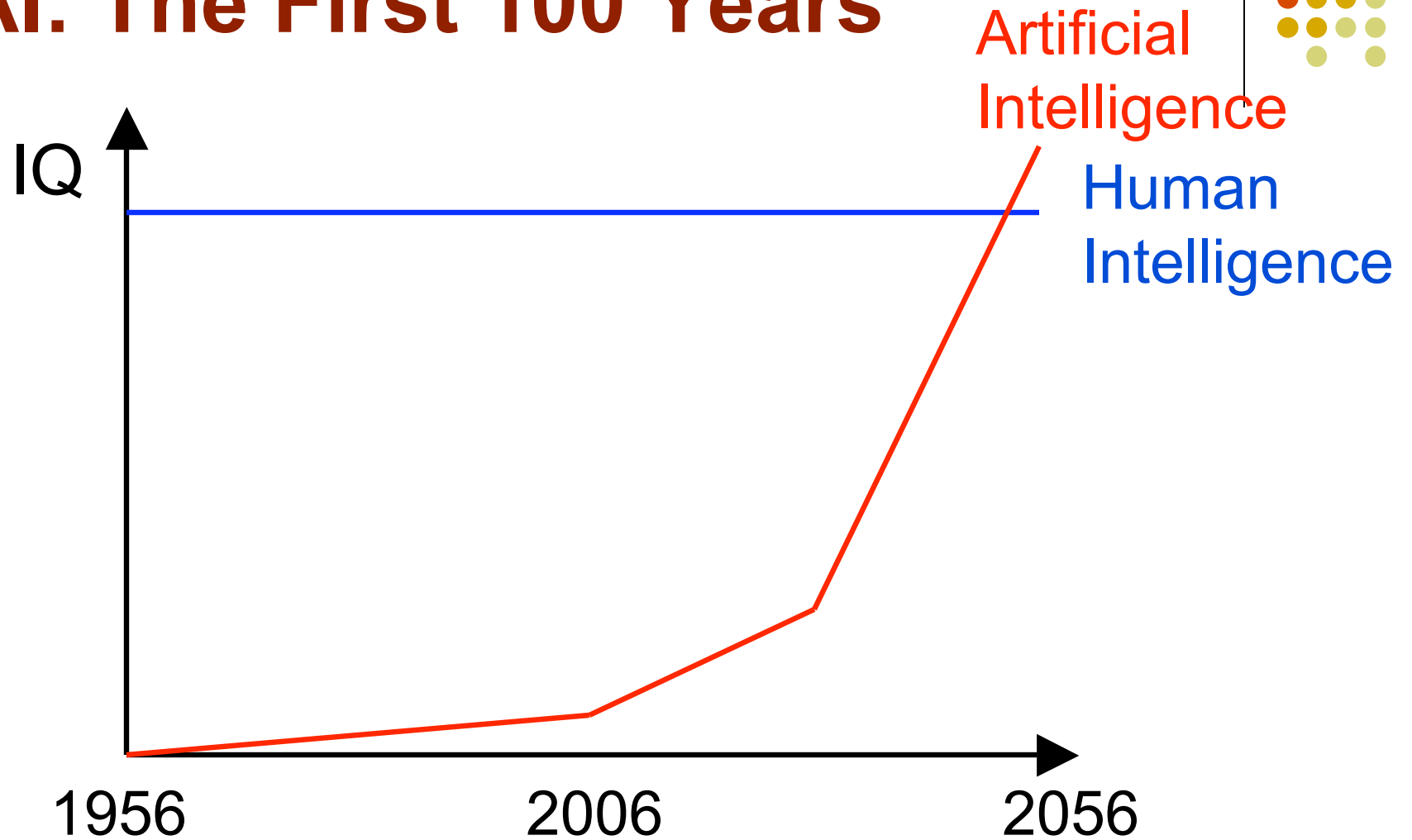
# AI: The First 100 Years



# AI: The First 100 Years



# AI: The First 100 Years



# Logical and Statistical AI



<b>Field</b>	<b>Logical approach</b>	<b>Statistical approach</b>
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:** WalkSAT, MCMC, KBMC
- **Learning:** Voted perceptron, pseudo-likelihood, inductive logic programming
- **Software:** Alchemy
- **Applications:** Information extraction, link prediction, etc.



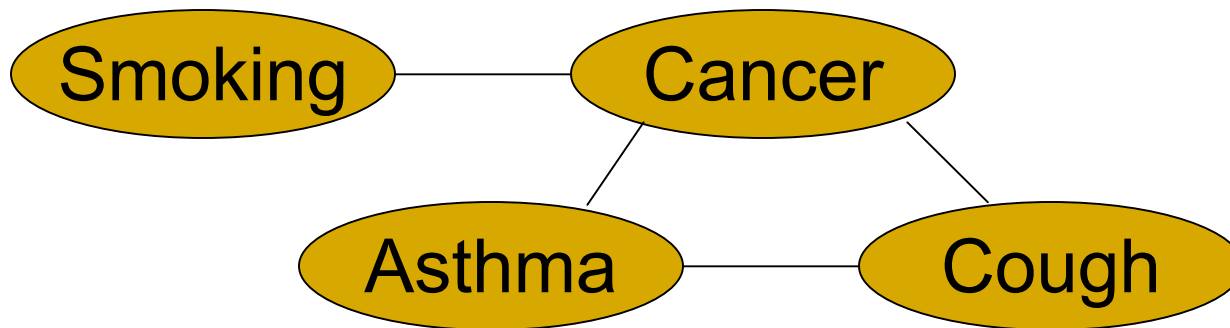
# Overview

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

# Markov Networks



- **Undirected** graphical models



- Potential functions defined over cliques

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

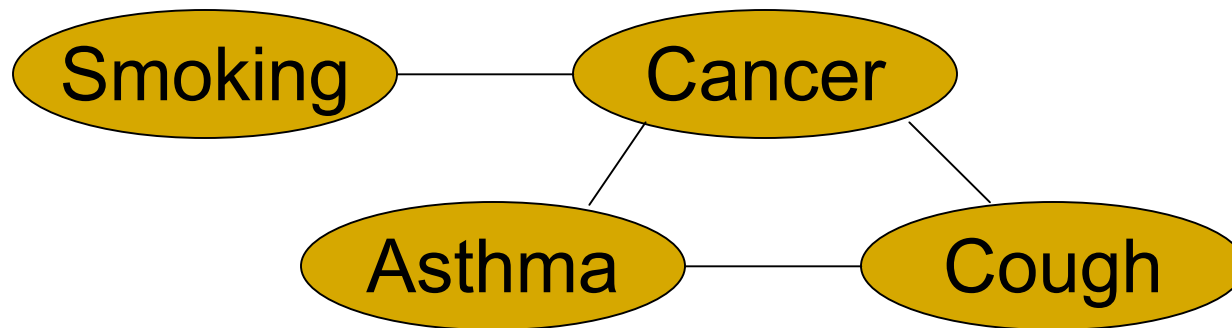
$$Z = \sum_x \prod_c \Phi_c(x_c)$$

Smoking	Cancer	$\Phi(S,C)$
False	False	4.5
False	True	4.5
True	False	2.7
True	True	4.5

# Markov Networks



- **Undirected** graphical models

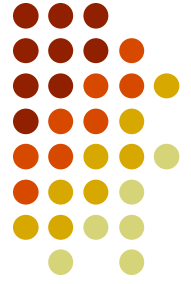


- Log-linear model:

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

Weight of Feature  $i$       Feature  $i$

$$f_1(\text{Smoking}, \text{Cancer}) = \begin{cases} 1 & \text{if } \neg \text{Smoking} \vee \text{Cancer} \\ 0 & \text{otherwise} \end{cases}$$
$$w_1 = 1.5$$



# First-Order Logic

- Constants, variables, functions, predicates  
E.g.: Anna, X, mother\_of(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



Smoking causes cancer.

Friends have similar smoking habits.

# Example: Friends & Smokers



$\forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x)$

$\forall x, y \text{ Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y))$

# Example: Friends & Smokers



1.5  $\forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x)$

1.1  $\forall x, y \text{ Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y))$

# Example: Friends & Smokers



1.5	$\forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x)$
1.1	$\forall x, y \text{ Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y))$

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

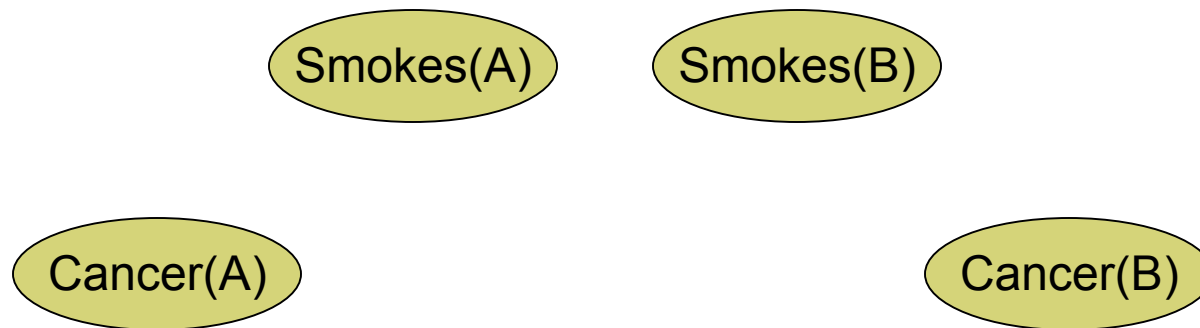
# Example: Friends & Smokers



1.5  $\forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x)$

1.1  $\forall x, y \text{ Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y))$

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



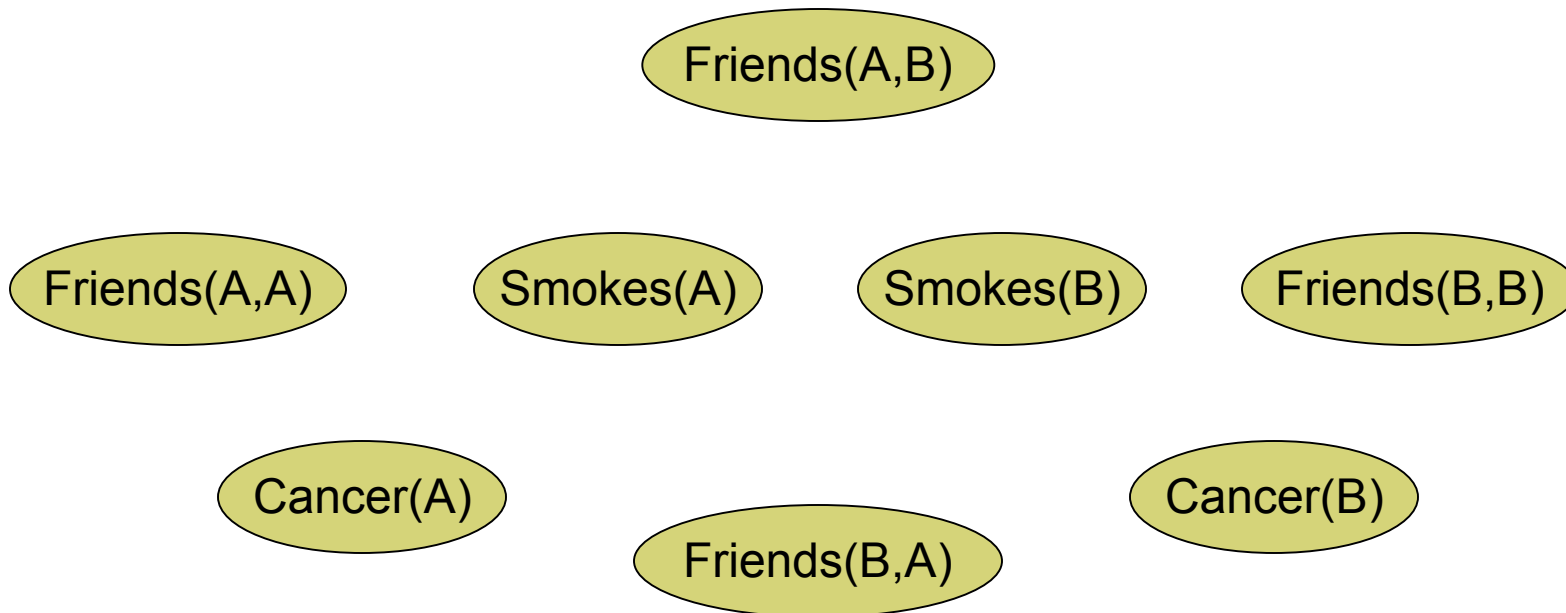
# Example: Friends & Smokers



1.5  $\forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x)$

1.1  $\forall x, y \text{ Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y))$

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

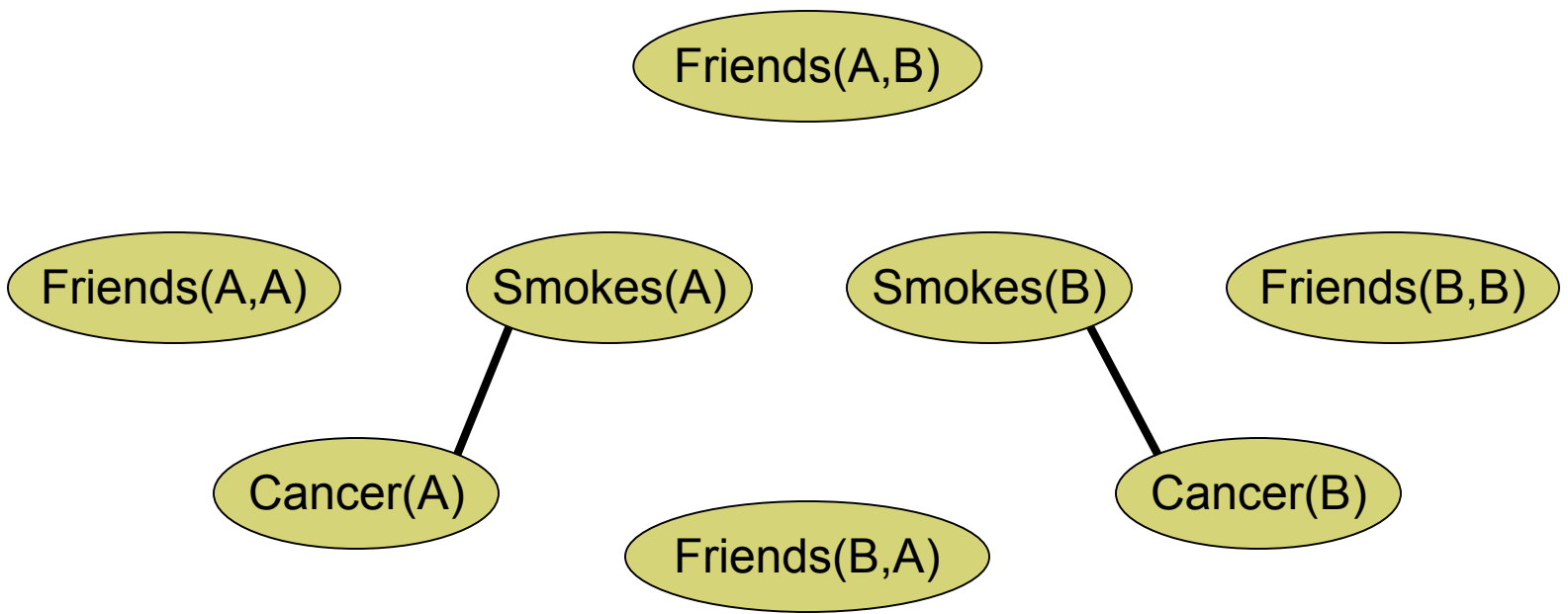




# Example: Friends & Smokers

- 1.5  $\forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x)$
- 1.1  $\forall x, y \text{ Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y))$

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

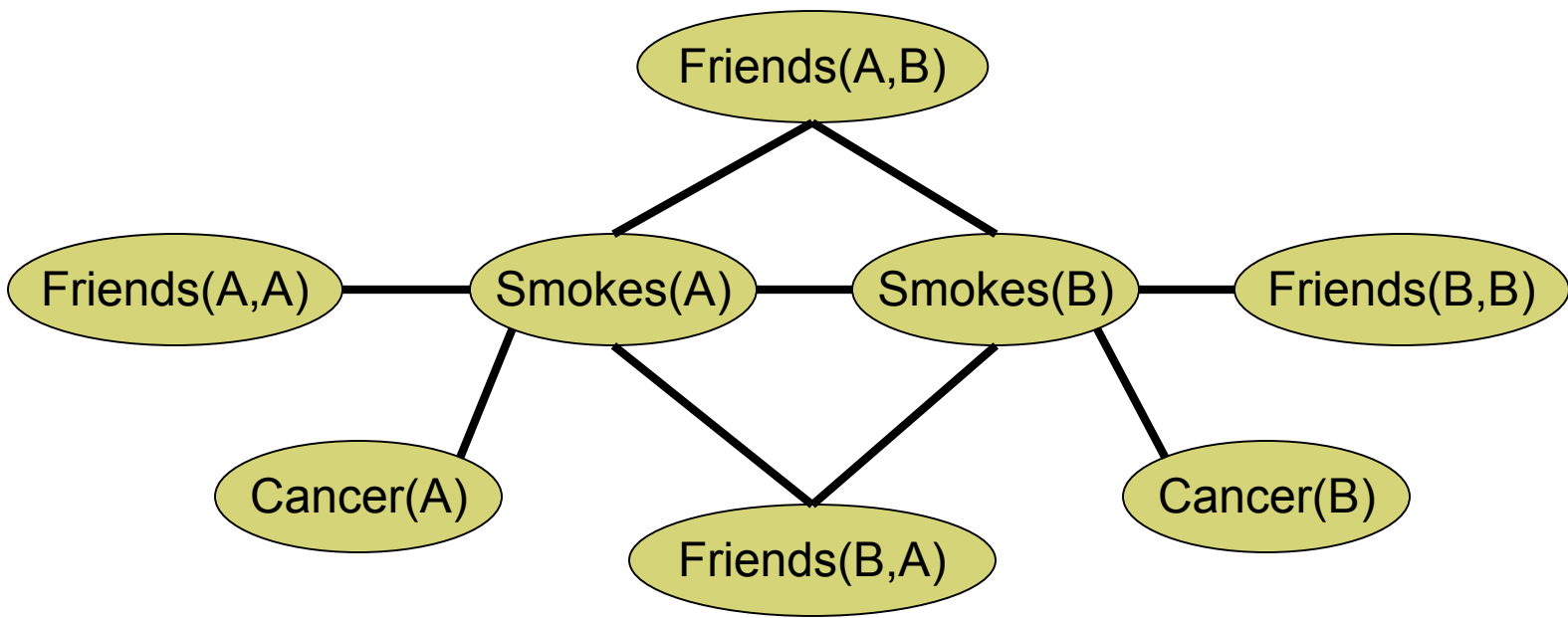




# Example: Friends & Smokers

- 1.5  $\forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x)$
- 1.1  $\forall x, y \text{ Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y))$

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.
- Open question: Infinite domains

# Relation to Statistical Models

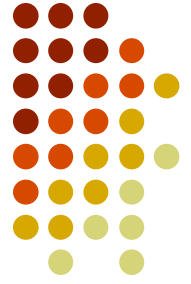


- Special cases:
  - Markov networks
  - Markov random fields
  - Bayesian networks
  - Log-linear models
  - Exponential models
  - Max. entropy models
  - Gibbs distributions
  - Boltzmann machines
  - Logistic regression
  - Hidden Markov models
  - Conditional random fields
- Obtained by making all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

# 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



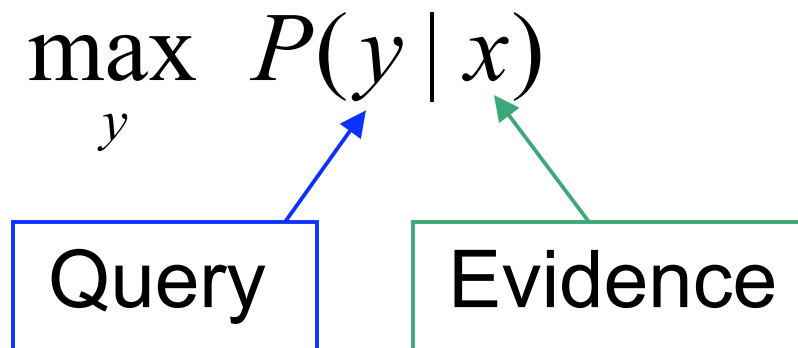
# Overview

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



# MAP/MPE Inference

- **Problem:** Find most likely state of world given evidence





# MAP/MPE Inference

- **Problem:** Find most likely state of world given evidence

$$\max_y \frac{1}{Z_x} \exp \left( \sum_i w_i n_i(x, y) \right)$$



# MAP/MPE Inference

- **Problem:** Find most likely state of world given evidence

$$\max_y \sum_i w_i n_i(x, y)$$



# MAP/MPE Inference

- **Problem:** Find most likely state of world given evidence

$$\max_y \sum_i w_i n_i(x, y)$$

- This is just the weighted MaxSAT problem
- Use weighted SAT solver  
(e.g., MaxWalkSAT [Kautz et al., 1997] )
- Potentially faster than logical inference (!)



# The WalkSAT Algorithm

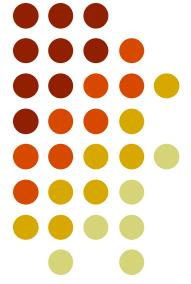


```
for  $i \leftarrow 1$  to max-tries do  
  solution = random truth assignment  
  for  $j \leftarrow 1$  to max-flips do  
    if all clauses satisfied then  
      return solution  
     $c \leftarrow$  random unsatisfied clause  
    with probability  $p$   
      flip a random variable in  $c$   
    else  
      flip variable in  $c$  that maximizes  
        number of satisfied clauses  
return failure
```

# The MaxWalkSAT Algorithm



```
for  $i \leftarrow 1$  to max-tries do  
    solution = random truth assignment  
    for  $j \leftarrow 1$  to max-flips do  
        if  $\sum \text{weights}(\text{sat. clauses}) > \text{threshold}$  then  
            return solution  
         $c \leftarrow$  random unsatisfied clause  
        with probability  $p$   
            flip a random variable in  $c$   
        else  
            flip variable in  $c$  that maximizes  
                 $\sum \text{weights}(\text{sat. clauses})$   
    return failure, best solution found
```



# But ... Memory Explosion

- **Problem:**

If there are  $n$  constants  
and the highest clause arity is  $c$ ,  
the ground network requires  $O(n^c)$  memory

- **Solution:**

Exploit sparseness; ground clauses lazily  
→ LazySAT algorithm [Singla & Domingos, 2006]



# Computing Probabilities

- $P(\text{Formula} | \text{MLN}, \text{C}) = ?$
- MCMC: Sample worlds, check formula holds
- $P(\text{Formula1} | \text{Formula2}, \text{MLN}, \text{C}) = ?$
- If **Formula2** = Conjunction of ground atoms
  - First construct min subset of network necessary to answer query (generalization of KBMC)
  - Then apply MCMC (or other)
- Can also do lifted inference [Braz et al, 2005]

# Ground Network Construction



```
network ← ∅  
queue ← query nodes  
repeat  
  node ← front(queue)  
  remove node from queue  
  add node to network  
  if node not in evidence then  
    add neighbors(node) to queue  
until queue = ∅
```

# MCMC: Gibbs Sampling



```
state ← random truth assignment
for i ← 1 to num-samples do
  for each variable x
    sample x according to  $P(x|neighbors(x))$ 
    state ← state with new value of x
P(F) ← fraction of states in which F is true
```



## But ... Insufficient for Logic

- **Problem:**  
Deterministic dependencies break MCMC  
Near-deterministic ones make it **very** slow
- **Solution:**  
Combine MCMC and WalkSAT  
→ MC-SAT algorithm [Poon & Domingos, 2006]



# 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



# Generative Weight Learning

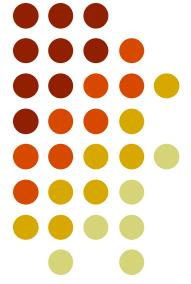
- Maximize likelihood
- Use gradient ascent or L-BFGS
- No local maxima

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

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

Expected no. true groundings according to model

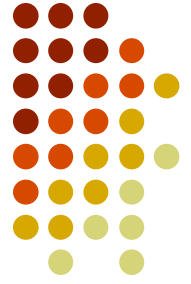
- Requires inference at each step (slow!)



# Pseudo-Likelihood

$$PL(x) \equiv \prod_i P(x_i | \text{neighbors}(x_i))$$

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



# Discriminative Weight Learning

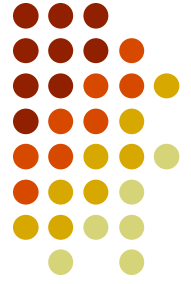
- Maximize conditional likelihood of query ( $y$ ) given evidence ( $x$ )

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

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

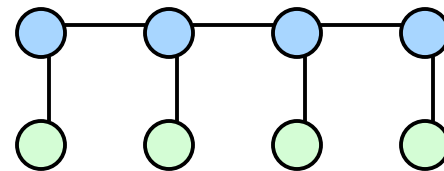
$w_i \leftarrow 0$

**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**  $\sum_t w_i / T$





# Voted Perceptron for MLNs

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

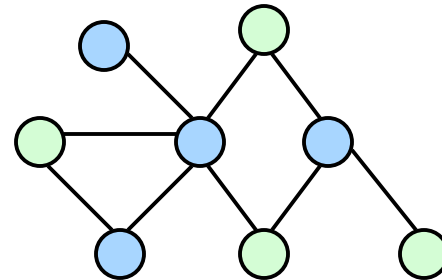
$w_i \leftarrow 0$

**for**  $t \leftarrow 1$  **to**  $T$  **do**

$y_{MAP} \leftarrow \text{MaxWalkSAT}(x)$

$w_i \leftarrow w_i + \eta [\text{count}_i(y_{Data}) - \text{count}_i(y_{MAP})]$

**return**  $\sum_t w_i / T$





# 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 search, shortest-first search





# Overview

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

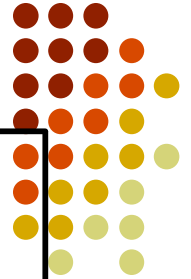


# Alchemy

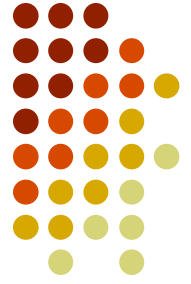
Open-source software including:

- Full first-order logic syntax
- Generative & discriminative weight learning
- Structure learning
- Weighted satisfiability and MCMC
- Programming language features

[www.cs.washington.edu/ai/alchemy](http://www.cs.washington.edu/ai/alchemy)



	<b>Alchemy</b>	<b>Prolog</b>	<b>BUGS</b>
<b>Represent- ation</b>	F.O. Logic + Markov nets	Horn clauses	Bayes nets
<b>Inference</b>	Model check- ing, MCMC	Theorem proving	MCMC
<b>Learning</b>	Parameters & structure	No	Params.
<b>Uncertainty</b>	Yes	No	Yes
<b>Relational</b>	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
- Online games
- Probabilistic Cyc
- Etc.

\* Markov logic approach won LLL-2005 information extraction competition [Riedel & Klein, 2005]



# Information Extraction

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

Singla, P., & Domingos, P. (2006). Memory-efficient 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). Efficient 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-efficient 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). Efficient inference. In Proceedings of the Twenty-First National Conference on Artificial Intelligence.



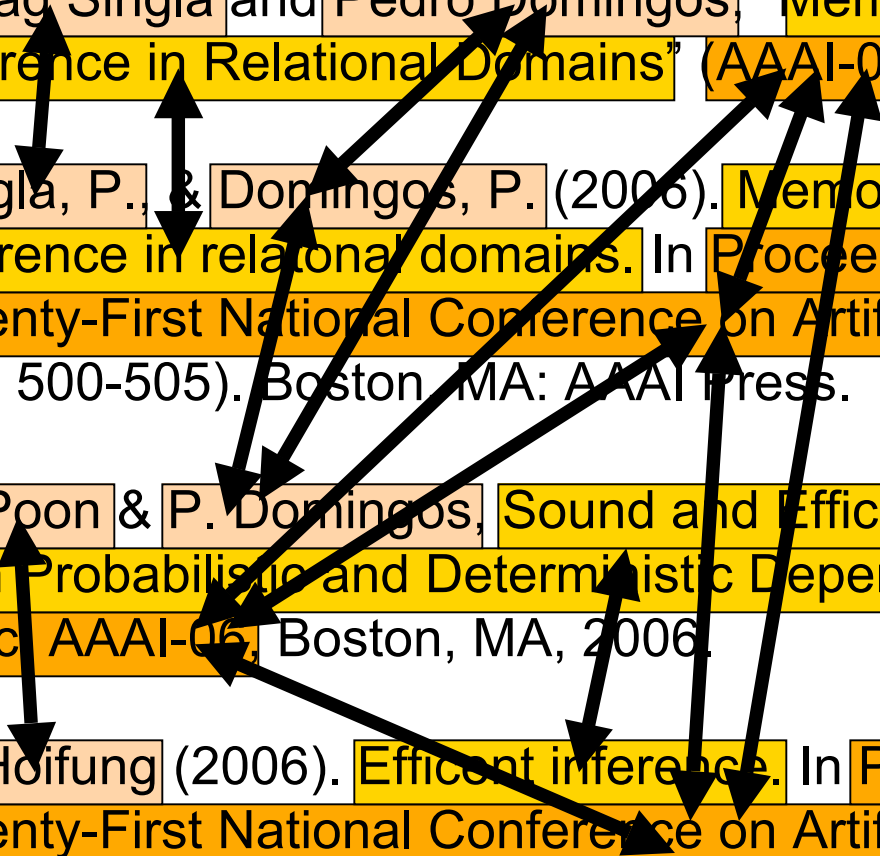
# Entity Resolution

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

Singla, P., & Domingos, P. (2006). Memory-efficient 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. Hofnung (2006). Efficient inference. In Proceedings of the Twenty-First National Conference on Artificial Intelligence.







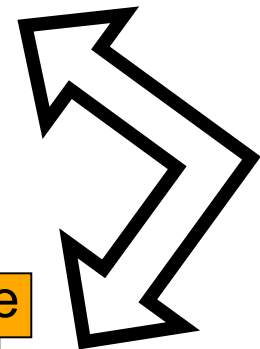
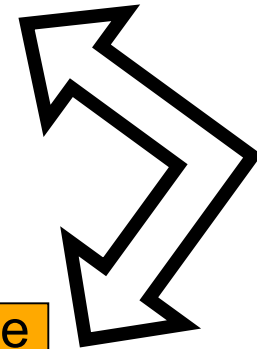
# Entity Resolution

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

Singla, P., & Domingos, P. (2006). Memory-efficient 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.

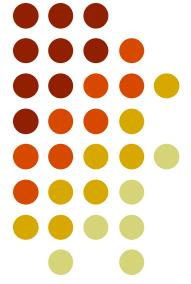
P. Hofnung (2006). Efficient inference. In Proceedings of the Twenty-First National Conference on Artificial Intelligence.





# 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: Eight 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)  
HasToken(field, citation, token)  
SameField(field, citation, citation)  
SameCit(citation, citation)
```

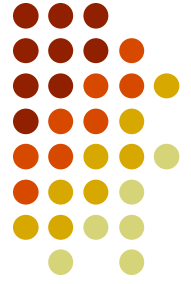
# Types and Predicates



```
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)  
HasToken(field, citation, token)  
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) ← Evidence  
InField(position, field, citation)  
HasToken(field, citation, token)  
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)
```

```
HasToken(field, citation, token)
```

```
SameField(field, citation, citation)
```

```
SameCit(citation, citation)
```

Query

Two arrows originate from the word 'Query' and point to the 'InField' and 'HasToken' predicates, indicating that these are queryable predicates.

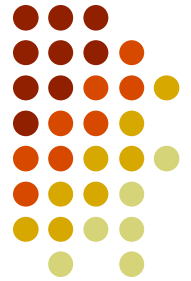


# Types and Predicates

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

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

# Formulas



$\text{Token}(+t, i, c) \Rightarrow \text{InField}(i, +f, c)$   
 $\text{InField}(i, +f, c) \Leftrightarrow \text{InField}(i+1, +f, c)$   
 $f \neq f' \Rightarrow (\neg \text{InField}(i, +f, c) \vee \neg \text{InField}(i, +f', c))$

$\text{Token}(t, i, c) \wedge \text{InField}(i, f, c) \Rightarrow \text{HasToken}(f, c, t)$   
 $\text{HasToken}(+f, c, +t) \wedge \text{HasToken}(+f, c', +t)$   
 $\quad \Rightarrow \text{SameField}(+f, c, c')$   
 $\text{SameField}(+f, c, c') \Leftrightarrow \text{SameCit}(c, c')$   
 $\text{SameField}(f, f') \wedge \text{SameField}(f', f'') \Rightarrow \text{SameField}(f, f'')$   
 $\text{SameCit}(c, c') \wedge \text{SameCit}(c', c'') \Rightarrow \text{SameCit}(c, c'')$



# 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) => HasToken(f,c,t)`

`HasToken(+f,c,+t) ^ HasToken(+f,c',+t)`

`=> SameField(+f,c,c')`

`SameField(+f,c,c') <=> SameCit(c,c')`

`SameField(f,f') ^ SameField(f',f'') => SameField(f,f'')`

`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) => HasToken(f,c,t)`

`HasToken(+f,c,+t) ^ HasToken(+f,c',+t)`

`=> SameField(+f,c,c')`

`SameField(+f,c,c') <=> SameCit(c,c')`

`SameField(f,f') ^ SameField(f',f'') => SameField(f,f'')`

`SameCit(c,c') ^ SameCit(c',c'') => SameCit(c,c'')`

# Formulas



$\text{Token}(+t, i, c) \Rightarrow \text{InField}(i, +f, c)$

$\text{InField}(i, +f, c) \Leftrightarrow \text{InField}(i+1, +f, c)$

$f \neq f' \Rightarrow (!\text{InField}(i, +f, c) \vee !\text{InField}(i, +f', c))$

$\text{Token}(t, i, c) \wedge \text{InField}(i, f, c) \Rightarrow \text{HasToken}(f, c, t)$

$\text{HasToken}(+f, c, +t) \wedge \text{HasToken}(+f, c', +t)$

$\Rightarrow \text{SameField}(+f, c, c')$

$\text{SameField}(+f, c, c') \Leftrightarrow \text{SameCit}(c, c')$

$\text{SameField}(f, f') \wedge \text{SameField}(f', f'') \Rightarrow \text{SameField}(f, f'')$

$\text{SameCit}(c, c') \wedge \text{SameCit}(c', c'') \Rightarrow \text{SameCit}(c, c'')$

# Formulas



$\text{Token}(+t, i, c) \Rightarrow \text{InField}(i, +f, c)$   
 $\text{InField}(i, +f, c) \Leftrightarrow \text{InField}(i+1, +f, c)$   
 $f \neq f' \Rightarrow (\neg \text{InField}(i, +f, c) \vee \neg \text{InField}(i, +f', c))$

$\text{Token}(t, i, c) \wedge \text{InField}(i, f, c) \Rightarrow \text{HasToken}(f, c, t)$

~~$\text{HasToken}(+f, c, +t) \wedge \text{HasToken}(+f, c', +t)$~~

$\Rightarrow \text{SameField}(+f, c, c')$

$\text{SameField}(+f, c, c') \Leftrightarrow \text{SameCit}(c, c')$

$\text{SameField}(f, f') \wedge \text{SameField}(f', f'') \Rightarrow \text{SameField}(f, f'')$

$\text{SameCit}(c, c') \wedge \text{SameCit}(c', c'') \Rightarrow \text{SameCit}(c, c'')$

# Formulas



$\text{Token}(+t, i, c) \Rightarrow \text{InField}(i, +f, c)$   
 $\text{InField}(i, +f, c) \Leftrightarrow \text{InField}(i+1, +f, c)$   
 $f \neq f' \Rightarrow (\neg \text{InField}(i, +f, c) \vee \neg \text{InField}(i, +f', c))$

$\text{Token}(t, i, c) \wedge \text{InField}(i, f, c) \Rightarrow \text{HasToken}(f, c, t)$

$\text{HasToken}(+f, c, +t) \wedge \text{HasToken}(+f, c', +t)$   
 $\Rightarrow \text{SameField}(+f, c, c')$

~~$\text{SameField}(+f, c, c') \Leftrightarrow \text{SameCit}(c, c')$~~

$\text{SameField}(f, f') \wedge \text{SameField}(f', f'') \Rightarrow \text{SameField}(f, f'')$

$\text{SameCit}(c, c') \wedge \text{SameCit}(c', c'') \Rightarrow \text{SameCit}(c, c'')$

# Formulas



$\text{Token}(+t, i, c) \Rightarrow \text{InField}(i, +f, c)$   
 $\text{InField}(i, +f, c) \Leftrightarrow \text{InField}(i+1, +f, c)$   
 $f \neq f' \Rightarrow (\neg \text{InField}(i, +f, c) \vee \neg \text{InField}(i, +f', c))$

$\text{Token}(t, i, c) \wedge \text{InField}(i, f, c) \Rightarrow \text{HasToken}(f, c, t)$   
 $\text{HasToken}(+f, c, +t) \wedge \text{HasToken}(+f, c', +t)$   
 $\Rightarrow \text{SameField}(+f, c, c')$

$\text{SameField}(+f, c, c') \Leftrightarrow \text{SameCit}(c, c')$

~~$\text{SameField}(f, f') \wedge \text{SameField}(f', f'') \Rightarrow \text{SameField}(f, f'')$~~

$\text{SameCit}(c, c') \wedge \text{SameCit}(c', c'') \Rightarrow \text{SameCit}(c, c'')$

# Formulas



$\text{Token}(+t, i, c) \Rightarrow \text{InField}(i, +f, c)$   
 $\text{InField}(i, +f, c) \Leftrightarrow \text{InField}(i+1, +f, c)$   
 $f \neq f' \Rightarrow (!\text{InField}(i, +f, c) \vee !\text{InField}(i, +f', c))$

$\text{Token}(t, i, c) \wedge \text{InField}(i, f, c) \Rightarrow \text{HasToken}(f, c, t)$   
 $\text{HasToken}(+f, c, +t) \wedge \text{HasToken}(+f, c', +t)$   
 $\Rightarrow \text{SameField}(+f, c, c')$

$\text{SameField}(+f, c, c') \Leftrightarrow \text{SameCit}(c, c')$

$\text{SameField}(f, f') \wedge \text{SameField}(f', f'') \Rightarrow \text{SameField}(f, f'')$

$\text{SameCit}(c, c') \wedge \text{SameCit}(c', c'') \Rightarrow \text{SameCit}(c, c'')$

# 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) => HasToken(f,c,t)`

`HasToken(+f,c,+t) ^ HasToken(+f,c',+t)`

`=> SameField(+f,c,c')`

`SameField(+f,c,c') <=> SameCit(c,c')`

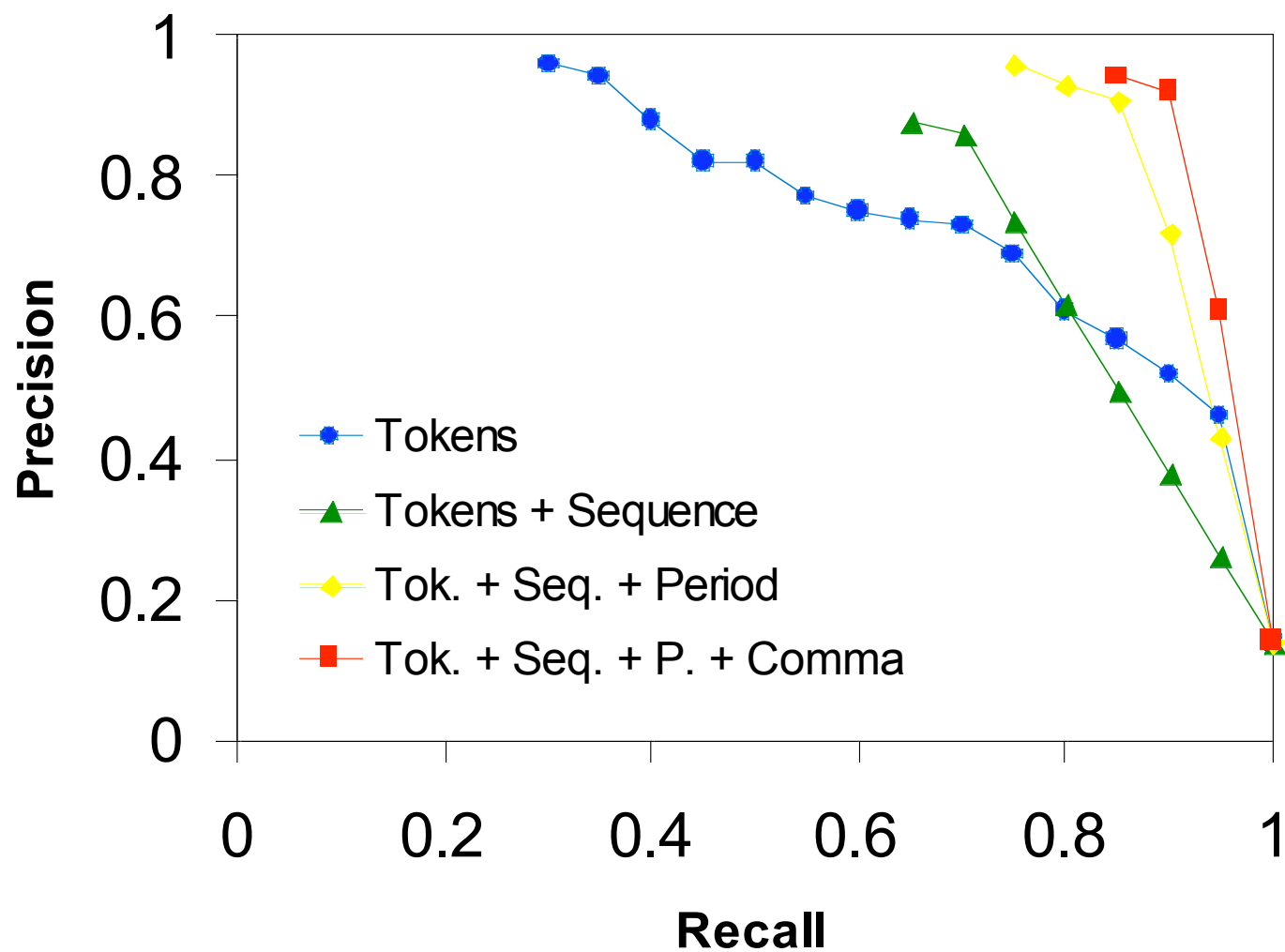
`SameField(f,f') ^ SameField(f',f'') => SameField(f,f'')`

`SameCit(c,c') ^ SameCit(c',c'') => SameCit(c,c'')`

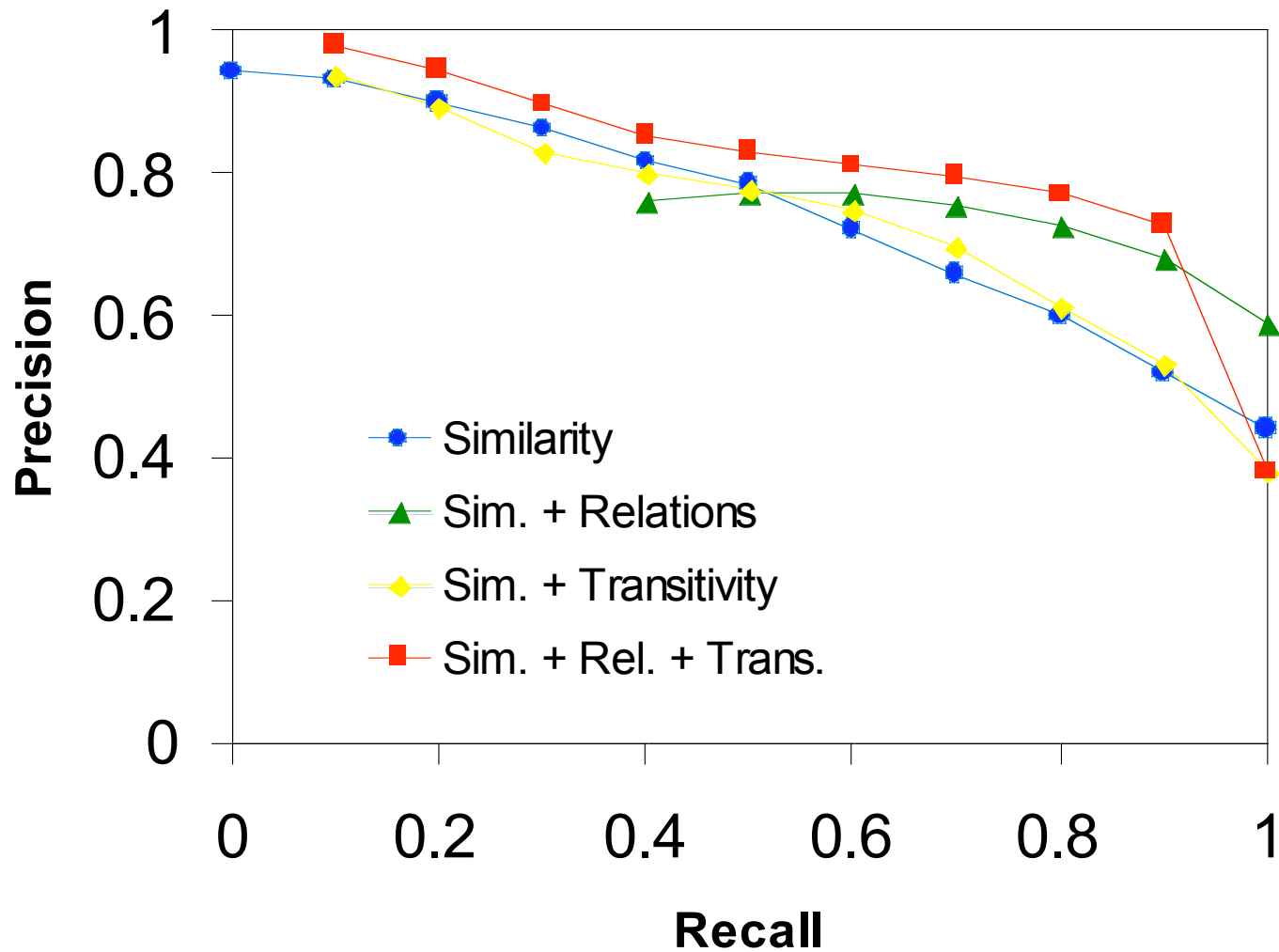


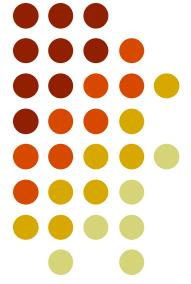


# Results: Segmentation on Cora



# Results: Matching Venues on Cora





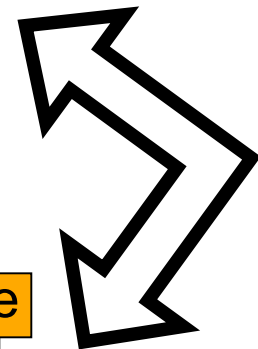
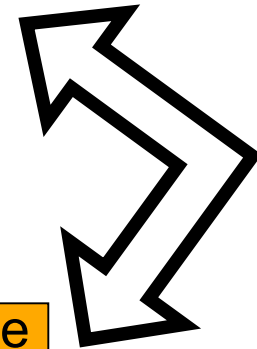
# Why It Works

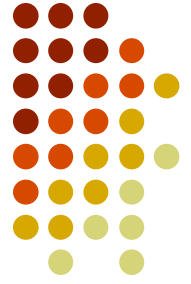
Parag Singla and Pedro Domingos, "Memory-Efficient Inference in Relational Domains" (AAAI-06).

Singla, P., & Domingos, P. (2006). Memory-efficient 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.

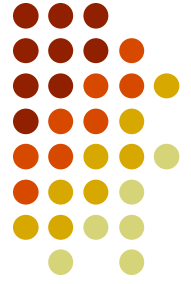
P. Hoifung (2006). Efficient inference. In Proceedings of the Twenty-First National Conference on Artificial Intelligence.





# Next Steps

- Broad-spectrum information extraction
- Instead of segmentation, parsing
- Resolve relations, paraphrases, etc.
- Add more knowledge
- Mine knowledge from extracted DB and add to MLN
- Do inference, use feedback
- Add physical sensors and effectors
- Bootstrap way to higher-level AI



# Overview

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



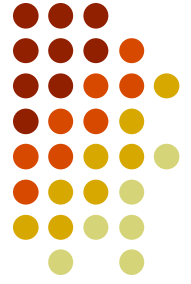
# The Interface Layer

**Applications**

**Interface Layer**

**Infrastructure**

# Networking



**Applications**

**WWW**

**Email**

**Interface Layer**

**Internet**

**Infrastructure**

**Protocols**

**Routers**



# Databases

Applications

ERP

CRM

OLTP

Interface Layer

Relational Model

Infrastructure

Query  
Optimization

Transaction  
Management



# Programming Systems



**Applications**

**Programming**

**Interface Layer**

**High-Level Languages**

**Infrastructure**

**Compilers**

**Code  
Optimizers**

# Artificial Intelligence



**Robotics**

**Planning**

**Applications**

**NLP**

**Multi-Agent  
Systems**

**Vision**

**Interface Layer**

**Representation**

**Infrastructure**

**Inference**

**Learning**

# Artificial Intelligence



**Robotics**

**Planning**

**Applications**

**NLP**

**Multi-Agent  
Systems**

**Vision**

**Interface Layer**

**First-Order Logic?**

**Representation**

**Infrastructure**

**Inference**

**Learning**

# Artificial Intelligence



**Robotics**

**Planning**

**Applications**

**NLP**

**Multi-Agent  
Systems**

**Vision**

**Interface Layer**

**Graphical Models?**

**Representation**

**Infrastructure**

**Inference**

**Learning**

# Artificial Intelligence



**Robotics**

**Planning**

**Applications**

**NLP**

**Multi-Agent  
Systems**

**Vision**

**Interface Layer**

**Markov Logic**

**Representation**

**Infrastructure**

**Inference**

**Learning**

# Artificial Intelligence



**Robotics**

**Planning**

**Applications**

**NLP**

**Multi-Agent  
Systems**

**Vision**

**Alchemy:** [www.cs.washington.edu/ai/alchemy](http://www.cs.washington.edu/ai/alchemy)

**Representation**

**Infrastructure**

**Inference**

**Learning**