CSE 573 Knowledge Representation: Propositional, FO & Markov Logic

Dan Weld

(With some slides from Mausam, Stuart Russell, Dieter Fox, Henry Kautz, Pedro Domingos, Min-Yen Kan…)

Irrationally held truths may be more harmful than reasoned errors.

- Thomas Huxley (1825-1895)

Project Presentations

- Friday 12/7
- Length = 4, 6, 7 or 8 min (includes questions) – practice!
- Default = your laptop; else mail me slides (.ppt or .pdf) by 9am Fri
  - Bring slides on a backup USB memory.
- Every team member should talk for some part of the presentation
- Subtopics to cover:
  - Aspirations & reality of what you built
  - Demo?
  - Suprises (What was harder or easier than expected?)
  - What did you learn?
  - Experiments & validation
  - Plans for remaining week
  - Who did what

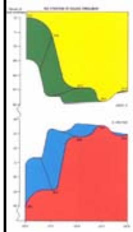
Final Reports (see web page)

- Goals for the project
- System design and algorithmic choices
- Sample screens of typical usage scenarios (if applicable)
- Experiments and results
- Anything you considered surprising or that you learned.
  - What would you do differently if you could?
- Conclusions and ideas for future work
- Appendices
- No limit on length, but **we appreciate good organization and tight, precise writing.** Points off for rambling and repetition.

Experiments

- Clearly state question being asked
- Kinds of experiments
  - Informal user study
  - Formal user study
  - System (or module) performance comparison
    - Baselines
    - Ablation experiments

Presenting Results

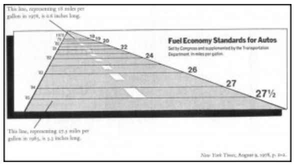


Graphs *vs* tables

Chartjunk

Data / ink ratio

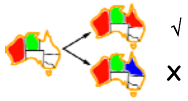
Visualization integrity



Previously

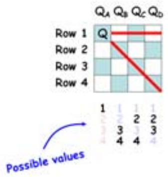
- CSPs are a special (factored) kind of search problem:
  - States defined by values (domains) of a fixed set of variables
  - Goal test defined by constraints on variable values
- Backtracking = DFS - one legal variable assigned per node
- Heuristics
  - Variable ordering: min remaining values

  - Value ordering: least contraining value



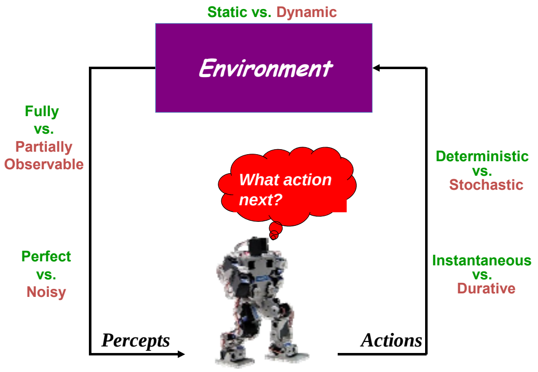
Previously

- CSPs are a special (factored) kind of search problem:
  - States defined by values (domains) of a fixed set of variables
  - Goal test defined by constraints on variable values
- Backtracking = DFS - one legal variable assigned per node
- Variable ordering and value selection heuristics help
- Forward checking prevents assignments that fail later



Previously

- CSPs are a special (factored) kind of search problem:
  - States defined by values (domains) of a fixed set of variables
  - Goal test defined by constraints on variable values
- Backtracking = DFS - one legal variable assigned per node
- Variable ordering and value selection heuristics help
- Forward checking prevents assignments that fail later
- Constraint propagation (e.g., arc consistency)
  - does additional work to constrain values and detect inconsistencies
- Constraint graph representation
  - Allows analysis of problem structure
- Tree-structured CSPs can be solved in linear time
- Local (stochastic) search often effective in practice
  - Iterative min-conflicts



Static vs. Dynamic

Environment

Fully vs. Partially Observable

Perfect vs. Noisy

What action next?

Deterministic vs. Stochastic

Instantaneous vs. Durative

Percepts

Actions

Algorithms

Blind search

Heuristic search

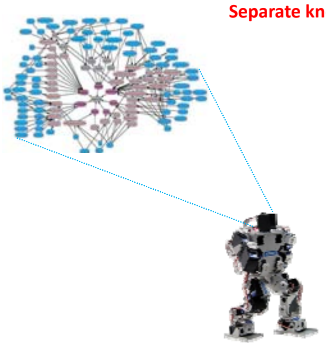
Mini-max & Expectima

MDPs (& POMDPS)

Reinforcement learnin

State estimation

Knowledge Representation



Separate knowledge from algorithms

HMMs

Bayesian networks

Propositional logic

First-order logic

Description logic

Constraint networks

Markov logic networks

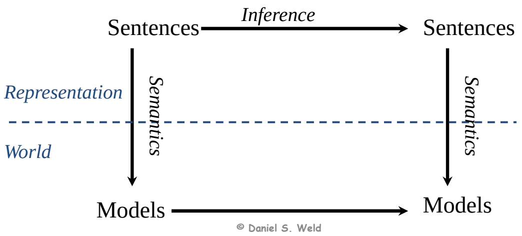
…

Overview

- Knowledge Representation & Reasoning
- Propositional Logic
  - Foundations: Syntax, semantics & inference
  - Algorithms: DPLL, Resolution, WalkSAT
  - Tractable subsets
- First-Order Logic
- Markov Logic

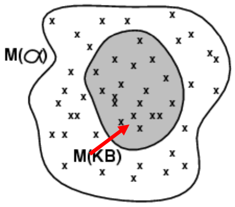
Semantics

- ***Syntax***: which arrangements of symbols are *legal*
  - (Def "sentences")
- ***Semantics***: what the symbols *mean* in the world
  - (Mapping between symbols and worlds)



Models

- Logicians often think in terms of models, which are formally structured worlds with respect to which truth can be evaluated
  - In propositional case, each model = truth assignment
  - Set of models can be enumerated in a truth table
- We say *m* is a model ***of*** a sentence α if α is true in *m*
- *M(α)* is the set of all models ***of*** α
- Then KB ╞ α iff *M(KB)* ⊆ *M*(α)
-
  - E.g. *KB* = (P∨Q) ∧(¬P ∨ R)
    α = (P∨R)
- How to check?
  - One way is to enumerate all elements in the truth table – slow ☹



Satisfiability, Validity, & Entailment

- S is **satisfiable** if it is true in ***some*** model (aka *world*, *interpretation*)
- S is **unsatisfiable** if it is false ***all*** models
- S is **valid** if it is true in ***all*** models
- S1 **entails** S2 if ***wherever*** S1 is true S2 is also true

---

Propositional Logic

- Syntax
  - Atomic sentences: P, Q, …
  - Connectives: ∧ , ∨, ¬, ⇒
- Semantics
  - Model = an assignment of T/F values to every atomic sentence
  - Truth Tables

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Satisfiability, Validity, & Entailment

- S is **satisfiable** if it is true in ***some*** model (aka *world*, *interpretation*)
- S is **unsatisfiable** if it is false ***all*** models
- S is **valid** if it is true in ***all*** models
- S1 **entails** S2 if ***wherever*** S1 is true S2 is also true

  P ∨ (Q ∧ ¬S ∧ ¬P) |=

Types of Reasoning (Inference)

- Deduction (showing **entailment**, |=)

  S = question

  Prove that KB |= S

  Two approaches:

  - Rules to derive new formulas from old (inference)
  - Show (KB ∧ ¬ S) is unsatisfiable

- Model Finding (showing **satisfiability**)

  S = description of problem

  Show S is satisfiable

  A kind of constraint satisfaction

Propositional Logic:
Inference Algorithms

1. Backward & Forward Chaining
2. Resolution (Proof by Contradiction)

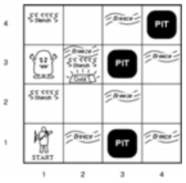
} **Deduction**

3. Exhaustive Enumeration
4. DPLL (Davis, Putnam Loveland & Logemann)
5. GSAT

} **Model Finding**

Wumpus World

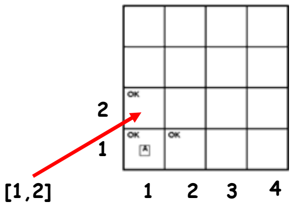
- Performance measure
  - Gold: +1000, death: -1000
  - -1 per step, -10 for using the arrow
- Environment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square



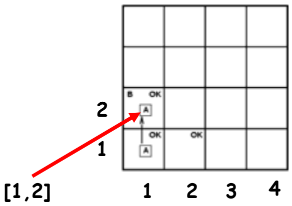
- Sensors: Stench, Breeze, Glitter, Bump, Scream
- Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

---

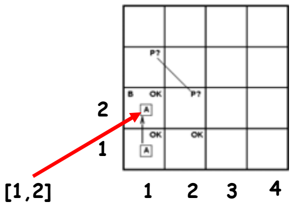
Exploring a wumpus world



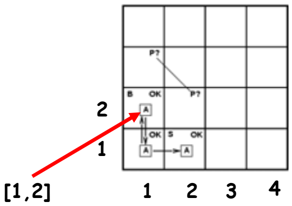
Exploring a wumpus world



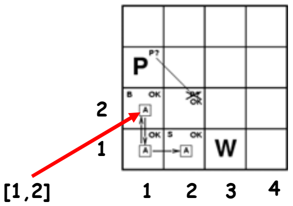
Exploring a wumpus world



Exploring a wumpus world



Exploring a wumpus world



Wumpus world sentences: KB

Let $P_{i,j}$ be true if there is a pit in [i, j].
Let $B_{i,j}$ be true if there is a breeze in [i, j].

KB:

$\neg P_{1,1}$

$\neg B_{1,1}$

"Pits cause breezes in adjacent squares"

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

Propositional Logic:
Inference Algorithms

1. Backward & Forward Chaining
2. Resolution (Proof by Contradiction)

} **Deduction**

3. Exhaustive Enumeration
4. DPLL (Davis, Putnam Loveland & Logemann)
5. GSAT

} **Model Finding**

Representing Formulae

- CNF = Conjunctive Normal Form
  - Conjunction (∧) of Disjunctions (∨)
- Represent as set of sets
  - ((A, B), (¬A, C), (¬C))
  - ((¬A), (A))
  - (())
  - ((A))
  - ()

Inference 4: DPLL
(Enumeration of ***Partial*** Models)

[Davis, Putnam, Loveland & Logemann 1962]

Version 1

```
dpll_1(pa){
  if (pa makes F false) return false;
  if (pa makes F true) return true;
  choose P in F;
  if (dpll_1(pa ∪ {P=0})) return true;
  return dpll_1(pa ∪ {P=1});
}
```

Returns true if F is satisfiable, false otherwise

DPLL Version 1

$(a \vee b \vee c)$

$(a \vee \neg b)$

$(a \vee \neg c)$

$(\neg a \vee c)$

DPLL Version 1

$(a \vee b \vee c)$

$(a \vee \neg b)$

$(a \vee \neg c)$

$(\neg a \vee c)$

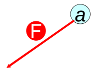
DPLL Version 1

$(F \vee b \vee c)$

$(F \vee \neg b)$

$(F \vee \neg c)$

$(T \vee c)$



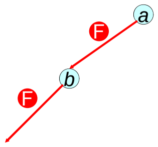
DPLL Version 1

$(F \vee F \vee c)$

$(F \vee T)$

$(F \vee \neg c)$

$(T \vee c)$



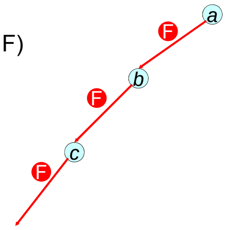
DPLL Version 1

$(F \vee F \vee F)$

$(F \vee T)$

$(F \vee T)$

$(T \vee F)$



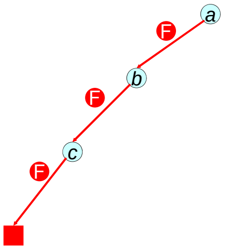
DPLL Version 1

F

T

T

T



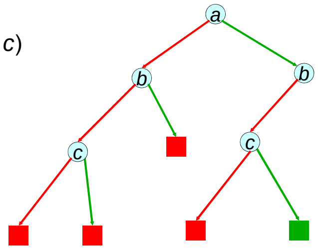
DPLL Version 1

$(a \vee b \vee c)$

$(a \vee \neg b)$

$(a \vee \neg c)$

$(\neg a \vee c)$



Improving DPLL

If literal L_1 is true, then clause $(L_1 \vee L_2 \vee ...)$ is true

If clause C_1 is true, then $C_1 \wedge C_2 \wedge C_3 \wedge ...$ has the same value as $C_2 \wedge C_3 \wedge ...$

Therefore: Okay to delete clauses containing true literals!

Improving DPLL

If literal L_1 is true, then clause $(L_1 \vee L_2 \vee ...)$ is true

If clause C_1 is true, then $C_1 \wedge C_2 \wedge C_3 \wedge ...$ has the same value as $C_2 \wedge C_3 \wedge ...$

Therefore: Okay to delete clauses containing true literals!

If literal L_1 is false, then clause $(L_1 \vee L_2 \vee L_3 \vee ...)$ has the same value as $(L_2 \vee L_3 \vee ...)$

Therefore: Okay to shorten clauses containing false literals

Improving DPLL

If literal L_1 is true, then clause $(L_1 \vee L_2 \vee ...)$ is true

If clause C_1 is true, then $C_1 \wedge C_2 \wedge C_3 \wedge ...$ has the same value as $C_2 \wedge C_3 \wedge ...$

Therefore: Okay to delete clauses containing true literals!

If literal L_1 is false, then clause $(L_1 \vee L_2 \vee L_3 \vee ...)$ has the same value as $(L_2 \vee L_3 \vee ...)$

Therefore: Okay to delete shorten containing false literals!

If literal L_1 is false, then clause (L_1) is false

Therefore: the empty clause means false!

DPLL version 2

```
dpll_2(F, literal){
  remove clauses containing literal
  if (F contains no clauses)return true;
  shorten clauses containing ¬literal
  if (F contains empty clause)
     return false;
  choose V in F;
  if (dpll_2(F, ¬V))return true;
  return dpll_2(F, V);
}
```

Partial assignment corresponding to a node is the set of chosen literals on the path from the root to the node

---

Benefit

- Like forward checking
- Can backtrack before getting to leaf

Structure in Clauses

- Unit Literals

  A literal that appears in a singleton clause

  {{¬b c}**{¬c}**{a ¬b e}{d b}{e a ¬c}}

  Might as well set it true! And simplify

  {{¬b} {a ¬b e}{d b}}

  {{d}}

- Pure Literals
  - A symbol that always appears with same sign
  - {{a ¬b c} {¬c **d** ¬e} {¬a ¬b e}{**d** b} {e a ¬c}}

  Might as well set it true! And simplify

  {{a ¬b c} {¬a ¬b e} {e a ¬c}}

DPLL (for real!)

Davis – Putnam – Loveland – Logemann

```
dpll(F, literal){
  remove clauses containing literal
  if (F contains no clauses) return true;
  shorten clauses containing ¬literal
  if (F contains empty clause)
     return false;
  if (F contains a unit or pure L)
     return dpll(F, L);
  choose V in F;
  if (dpll(F, ¬V))return true;
  return dpll(F, V);
}
```

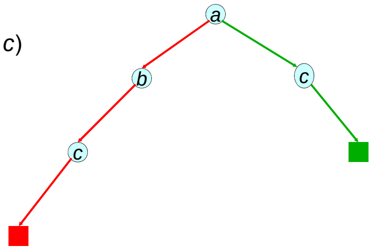
DPLL (for real)

$(a \vee b \vee c)$

$(a \vee \neg b)$

$(a \vee \neg c)$

$(\neg a \vee c)$



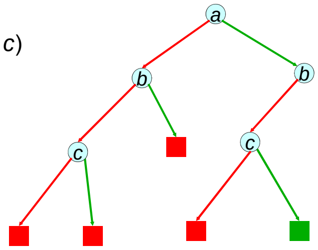
Compare with DPLL Version 1

$(a \vee b \vee c)$

$(a \vee \neg b)$

$(a \vee \neg c)$

$(\neg a \vee c)$



Heuristic Search in DPLL

- Heuristics are used in DPLL to select a (non-unit, non-pure) proposition for branching
- Idea: identify a most constrained variable
  - Likely to create many unit clauses
- MOM's heuristic:
  - **M**ost **o**ccurrences in clauses of **m**inimum length

---

Success of DPLL

- 1962 – DPLL invented
- 1992 – 300 propositions
- 1997 – 600 propositions (satz)
- Additional techniques:
  - Learning conflict clauses at backtrack points
  - Randomized restarts
  - 2002 (zChaff) 1,000,000 propositions – encodings of hardware verification problems

---

Other Ideas?

- How else could we solve SAT problems?

WalkSat (Take 1)

- ***Local*** search (Hill Climbing + Random Walk) over space of ***complete*** truth assignments
  - With prob p: flip any variable in any unsatisfied clause
  - With prob (1-p): flip best variable in any unsat clause
    - best = one which minimizes #unsatisfied clauses

Refining Greedy Random Walk

- Each flip
  - makes some false clauses become true
  - breaks some true clauses, that become false
- Suppose s1→s2 by flipping x. Then:

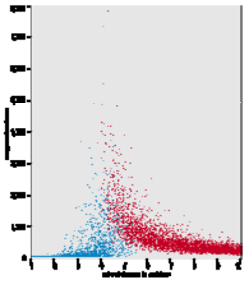
  #unsat(s2) = #unsat(s1) – make(s1,x) + break(s1,x)
- Idea 1: if a choice breaks nothing, it's likely good!
- Idea 2: near the solution, only the break count matters
  - the make count is usually 1

Walksat (Take 2)

```
state = random truth assignment;
while ! GoalTest(state) do
    clause := random member { C | C is false in state };
    for each x in clause do compute break[x];
    if exists x with break[x]=0 then var := x;
    else
        with probability p do
            var := random member { x | x is in clause };
        else
            var := arg x min { break[x] | x is in clause };
    endif
    state[var] := 1 – state[var];
end
return state;
```

**Put everything inside of a restart loop.
Parameters: p, max_flips, max_runs**

Random 3-SAT



- Random 3-SAT
  - sample uniformly from space of all possible 3-clauses
  - *n* variables, *l* clauses
- Which are the hard instances?
  - around *l*/*n* = 4.3

Special Syntactic Forms

- General Form:

  ((q∧¬ r) → s)) ∧ ¬ (s ∧ t)
- Conjunction Normal Form (CNF)

  (¬ q ∨ r ∨ s) ∧ (¬ s ∨ ¬ t)

  Set notation: { (¬ q, r, s), (¬ s, ¬ t) }

  empty clause () = *false*
- Binary clauses: 1 or 2 literals per clause

  (¬ q ∨ r) (¬ s ∨ ¬ t)
- Horn clauses: 0 or 1 positive literal per clause

  (¬ q ∨ ¬ r ∨ s) (¬ s ∨ ¬ t)

  (q∧r) → s (s∧t) → *false*

Prop. Logic Themes

- Expressiveness

  Expressive but awkward

  No notion of objects, properties, or relations

  Number of propositions is fixed

  Brittle
- Tractability

  NP in general

  Completeness / speed tradeoff

  Horn clauses, binary clauses

Overview

- Knowledge Representation & Reasoning
- Propositional Logic
- First-Order Logic
  - Foundations: Syntax, semantics & inference
  - Algorithms: Chaining, Resolution, Compilation to SAT
  - Tractable subsets
- Markov Logic

Propositional. Logic *vs.* **First Order**

Ontology	Propositional Symbols	Objects, Properties, Relations
Syntax	Atomic sentences Connectives	Variables & quantification Sentences have structure: terms father-of(mother-of(X)))
Semantics	Truth Tables	Interpretations *(Much more complicated)*
Inference Algorithm	DPLL, WalkSAT Fast in practice	Unification Forward, Backward chaining Prolog, theorem proving
Complexity	NP-Complete	Semi-decidable

FOL Definitions

- ***Constants***: a,b, dog33.
  - Name a specific object.
- ***Variables***: X, Y.
  - Refer to an object without naming it.
- ***Functions***: dad-of
  - Mapping from objects to objects.
- ***Terms***: dad-of(dog33)
  - Refer to objects
- ***Atomic Sentences***: in(dad-of(dog33), food6)
  - Can be true or false
  - Correspond to propositional symbols P, Q

More Definitions

- **Quantifiers**:
  - ∀ Forall
  - ∃ There exists
- **Examples**
  - Dumbo is grey
    grey(dumbo)
  - Elephants are grey
    ∀ x elephant(x) ⇒ grey(x)
  - There is a grey elephant
    ∃ x elephant(x) ∧ grey(x)

Quantifier / Connective Interaction

E(x) == "x is an elephant"
G(x) == "x has the color grey"

1. ∀x E(x) ∧ G(x)
2. ∀x E(x) ⇒G(x)
3. ∃x E(x) ∧ G(x)
4. ∃x E(x) ⇒G(x)

Nested Quantifiers: Order matters!

$$\forall x \exists y \; P(x,y) \neq \exists y \forall x \; P(x,y)$$

- Examples
  - Every dog has a tail

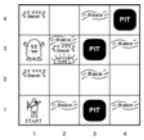
$\forall d \exists t \; has(d,t)$?

Every dog ***shares*** a tail!
$\exists t \forall d \; has(d,t)$

Someone is loved by everyone
$\exists x \forall y \; loves(y, x)$

Wumpus world in prop logic

Let $P_{i,j}$ be true if there is a pit in [i, j].
Let $B_{i,j}$ be true if there is a breeze in [i, j].



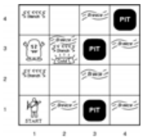
KB:

$\neg P_{1,1}$
$\neg B_{1,1}$

"Pits cause breezes in adjacent squares"
$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

Wumpus world in prop logic

Let pit(i,j) be true if there is a pit in [i, j].
Let breeze(i,j) be true if breezy in [i, j].



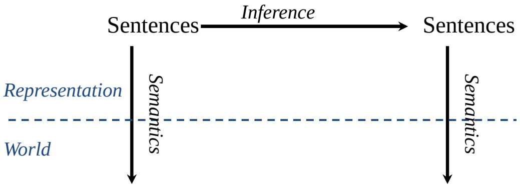
KB:

$\neg$ pit(1,1)
$\neg$ breeze(1,1)

"Pits cause breezes in adjacent squares"
$\forall$i,j breeze(i,j) $\Leftrightarrow$ pit(i, add(j,1)) $\vee$ pit(i, add(j, -1)) $\vee$ ...

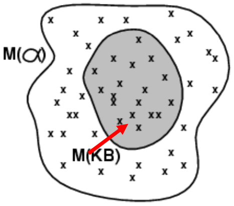
Semantics

- ***Syntax***: a description of the *legal* arrangements of symbols
  - (Def "sentences")
- ***Semantics***: what the arrangement of symbols *means* in the world



Models

- Logicians often think in terms of models, which are formally structured worlds with respect to which truth can be evaluated
  - In propositional case, each model = truth assignment
  - Set of models can be enumerated in a truth table
- We say *m* is a model ***of*** a sentence α if α is true in m
  - (Equivalently "m satisfies α")
- *M(α)* is the set of all models ***of*** α
- Then KB $\models$ α iff *M(KB)* $\subseteq$ *M*(α)
-
  - E.g. *KB* = (P∨Q) ∧(¬P ∨ R)
    α = (P∨R)



Satisfiability, Validity, & Entailment

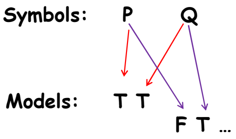
- S is **valid** if it is true in all models
- S is **satisfiable** if it is true in some model
- S is **unsatisfiable** if it is false all model

|=

- S1 **entails** S2 if
  - For all models where S1 is true,
  - S2 is also true

Propositional Logic: **SEMANTICS**

- Possible modela are TRUTH ASSIGNMENTS
  - Assignment to each variable either T or F
  - Assignment of T or F to each connective

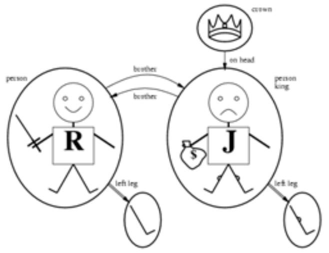


		Q	
		T	F
P	T		
	F		

P ∧ Q

First Order Logic: Worlds

- Depiction of one possible world

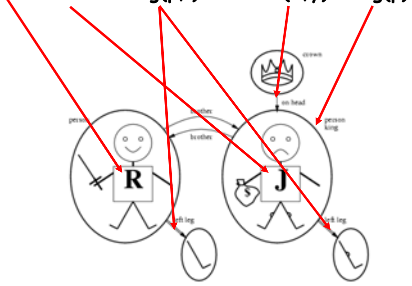


Models = Mappings
syntactic tokens → world elements

Depiction of one possible interpretation, assuming

- Constants: Functions: Relations:

Richard John Leg(p,l) On(x,y) King(p)

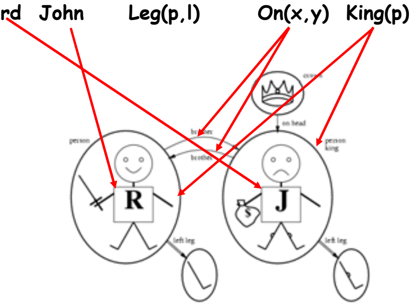


Models=Mappings
syntactic tokens → world elements

Another interpretation, same assumptions

- Constants: Functions: Relations:

Richard John Leg(p,l) On(x,y) King(p)



FOL Reasoning

- FO Forward & Backward Chaining
- FO Resolution
- Many other types of theorem proving
- Specialized provers for restricted representations
  - Description logics
  - Horn Clauses
- Compilation to SAT

Compilation to Prop. Logic I

- Typed Logic
  - $\forall_{city}\, a,b$ connected(a,b)
- Finite Universe
  - Cities: seattle, tacoma, enumclaw
- Equivalent propositional formula:

Compilation to Prop. Logic II

- Universe
  - Cities: Seattle, Chicago
  - Firms: Microsoft, Boeing
- First-Order formula
  - $\forall_{city}\, c\ \exists_{firm}\, f$ hasHQ(c, f)
- Equivalent propositional formula?

Hey!

- You said FO Inference is semi-decidable
- But you compiled it to SAT
  - Which is NP Complete
- So now we can always do the inference?!?
  - Tho it might take exponential time…
- Something seems wrong here….????

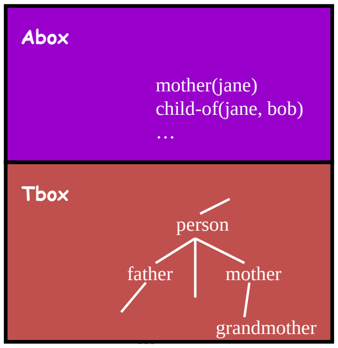
Restricted Forms of FO Logic

- Known, Finite Universes
  - Compile to SAT
- Description Logics (Frame Systems)
  - Ban certain types of expressions
- Horn Clauses
  - Aka Prolog
- Function-Free Horn Clauses
  - Aka Datalog

KR with Description Logics

Assertions

Term Defs



Logical and Statistical AI

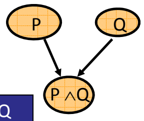
Field	Logical approach	Statistical approach
Knowledge representation	First-order logic	Graphical models
Automated reasoning	Satisfiability testing	Markov chain Monte Carlo
Machine learning	Inductive logic programming	Neural networks
Planning	Classical planning	Markov decision processes
Natural language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

---

Relationship

$P \wedge Q$

$P \vee Q$

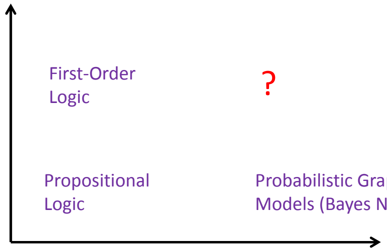


P	Q	$P \wedge Q$
+	+	1.0
+	–	0
–	+	0
–	–	0

Propositional Logic

Probabilistic Graphical Models (Bayes Nets)

Knowledge Representation



Complexity
Entities & Relations

First-Order Logic

?

Propositional Logic

Probabilistic Graphical Models (Bayes Nets)

Uncertainty

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.
- **Here at UW: MLNs** [Richardson & Domingos, 2004]

Markov Logic

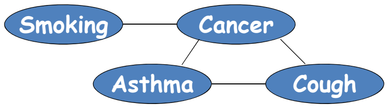
- **Syntax:** Weighted first-order formulas
- **Semantics:** Templates for Markov nets
- **Inference:** WalkSAT, MCMC, KBMC
- **Learning:** Voted perceptron, pseudo-likelihood, inductive logic programming
- **Software:** Alchemy
- **Applications:** Information extraction, link prediction, etc.

Overview

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

Markov Networks

- **Undirected** graphical models



- **Potential functions defined over cliques**

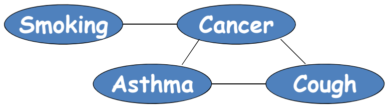
$$P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c)$$

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

Smoking	Cancer	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:**

$$P(x) = \frac{1}{Z} \exp\left(\sum_i \underbrace{w_i}_{\text{Weight of Feature } i} \underbrace{f_i(x)}_{\text{Feature } i} \right)$$

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

$$w_1 = 1.5$$

First-Order Logic

- Constants, variables, functions, predicates
  E.g.: Anna, x, 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)

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

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

$$\forall x \; Smokes(x) \Rightarrow Cancer(x)$$
$$\forall x, y \; Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$$

Example: Friends & Smokers

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

Example: Friends & Smokers

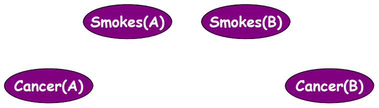
1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y))$

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

Example: Friends & Smokers

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

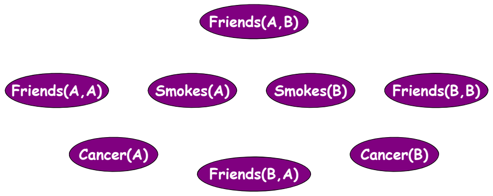
Two constants: Anna (A) and Bob (B)



Example: Friends & Smokers

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

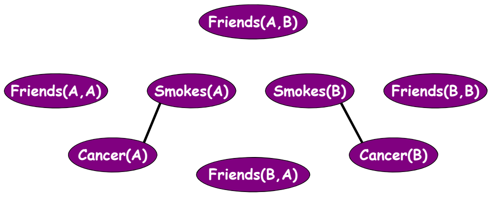
Two constants: Anna (A) and Bob (B)



Example: Friends & Smokers

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

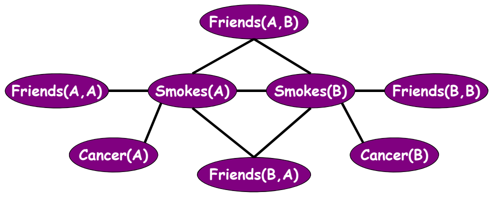
Two constants: Anna (A) and Bob (B)



Example: Friends & Smokers

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

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.
- 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

MAP/MPE Inference

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

$$\arg\max_y \; P(y \mid x)$$

Query

Evidence

MAP/MPE Inference

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

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

MAP/MPE Inference

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

$$\arg\max_{y} \sum_{i} w_i n_i(x, y)$$

MAP/MPE Inference

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

$$\arg\max_{y} \sum_{i} w_i n_i(x, y)$$

- 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 ← 1 to max-tries do
    solution = random truth assignment
    for j ← 1 to max-flips do
        if all clauses satisfied then
            return solution
        c ← 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 ← 1 to max-tries do
    solution = random truth assignment
    for j ← 1 to max-flips do
        if ∑ weights(sat. clauses) > threshold then
            return solution
        c ← random unsatisfied clause
        with probability p
            flip a random variable in c
        else
            flip variable in c that maximizes
                ∑ weights(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(Formula|MLN,C) = ?
- MCMC: Sample worlds, check formula holds
- P(Formula1|Formula2,MLN,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 ← Ø
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 = Ø
```

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 (formulas)

Generative Weight Learning

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

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

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

Expected no. true groundings according to mode

- 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)

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

No. of true groundings of clause i in data

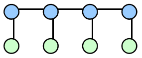
Expected no. true groundings according to model

- 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

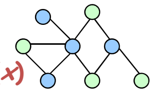
```
$w_i \leftarrow 0$
for $t \leftarrow 1$ to $T$ do
    $y_{MAP} \leftarrow$ Viterbi($x$)
    $w_i \leftarrow w_i + \eta$ [count$_i$($y_{Data}$) – count$_i$($y_{MAP}$)]
return $\sum_t w_i$ / T
```



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
    $y_{MAP} \leftarrow$ MaxWalkSAT($x$)
    $w_i \leftarrow w_i + \eta$ [count$_i$($y_{Data}$) – 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 [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
- Generative & discriminative weight learning
- Structure learning
- Weighted satisfiability and MCMC
- Programming language features

alchemy.cs.washington.edu

	Alchemy	**Prolog**	**BUGS**
Representation	F.O. Logic + Markov nets	Horn clauses	Bayes nets
Inference	Model checking, MC-SAT	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.

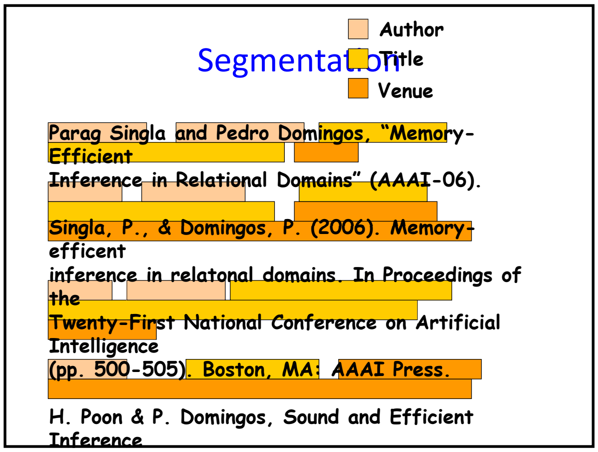
*** Markov logic approach won LLL-2005 information extraction competition [Riedel & Klein, 2005]**

Information Extraction

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

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

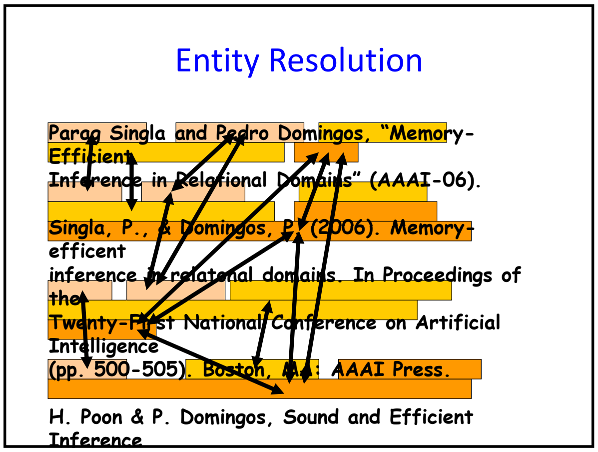


Segmentation

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

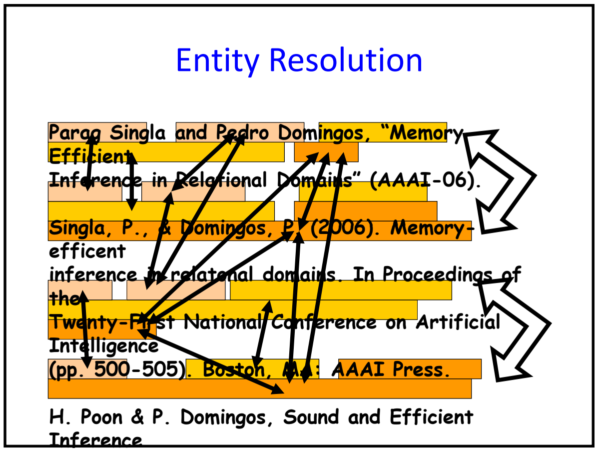


Entity Resolution

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



Entity Resolution

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

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, ...}Optional
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, ...}
                                         ← Evidence
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)  → Query
SameField(field, citation, citation)
SameCit(citation, citation)
```

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’,

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i’,
   ^ 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’,

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i’,
   ^ 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’,

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i’,
   ^ 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’,

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i’,
   ^ 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”)
```

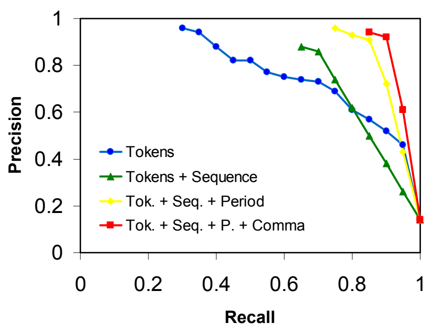
Formulas

```
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
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