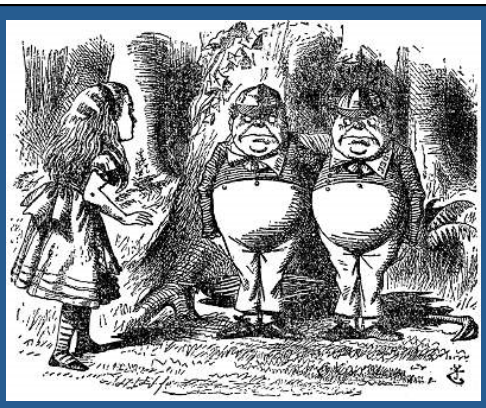


"I know what you're thinking about," said Tweedledum; "but it isn't so, nohow."

"Contrariwise," continued Tweedledee, "if it was so, it might be; and if it were so, it would be; but as it isn't, it ain't. That's logic."

Propositional Logic

EECS 492
February 8th, 2011

Administrative

- Practice exam has been posted
- Quiz review tonight, 7-9p right here!
  - Will go over practice exam.
  - BYOQ!
- Exam: 7-9p in CHRYS 220
  - Just you and a pencil-- closed book

Last time

- Propositional Logic
  - Connectives (logic gates)
  - Entailment (inference)
- We used human brains to deduce new facts from those that we already knew.
  - i.e., where it's safe to go in the Wumpus world.

Today

- A tiny, tiny review.
- How do we automate inference of propositional logic?
  - Simple method:
    - Model Checking
  - Constructing proofs
  - Resolution
  - Search strategies
    - Forward Chaining
    - PL-Resolution

Five Logical Connectives

P	Q	¬P (not)	P∧Q (and)	P∨Q (or)	P⇒Q (implies)	P⇔Q (if and only if)
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Odd Implications

- "5 is odd" *implies* "Tokyo is the capital of Japan"
- "5 is even" *implies* "Tokyo is the capital of Japan"
- "5 is even" *implies* "Tokyo is the capital of the United States"

Mini Quiz

Are the following sentences:

- Valid?
- Satisfiable?

Consider these sentences (with respect to the vacuum world)

- $(D_A \vee D_B) \wedge \neg D_B \wedge D_A$?
- $[(D_A \vee D_B) \wedge \neg D_B] \Rightarrow D_A$?
- $[(D_A \vee D_B) \wedge \neg D_B] \Leftrightarrow D_A$?

Inference

- Process by which some sentences are derived from others.
  - Aka *reasoning*
  - Record of an inference process called a proof
- Can derive α from KB using method *i*:

$$KB \vdash_i \alpha$$

Properties of inference methods

- Soundness

$$KB \vdash_i \alpha \quad \text{only when} \quad KB \models \alpha$$

i.e., no false conclusions

- Completeness

whenever $KB \models \alpha$ it is true that $KB \vdash_i \alpha$

i.e., no missing conclusions

Gottfried Wilhelm Leibniz (1646-1716)

… if we could find characters or signs appropriate for expressing all our thoughts as definitely and as exactly as arithmetic expresses numbers…, we could in all subjects in so far as they are amenable to reasoning accomplish what is done in Arithmetic… For all inquiries… would be performed by the transposition of characters and by a kind of calculus, which would immediately facilitate the discovery of beautiful results…

—*Dissertio de Arte Combinatoria, 1666*

Inference Methods

- Now, all we need is an inference method we can implement!
- We'll describe several!
  - Model Checking
  - Proof Trees
  - Forward Chaining
  - Backwards Chaining

Model Checking

$$KB \overset{?}{\vdash_i} \alpha$$

- A generic inference mechanism
- Enumerate all models (i.e., truth assignments) and check that α is valid in which KB is true
  - I.e., check that **α is valid** given KB
  - Time complexity: $O(2^n)$
  - Space complexity: $O(n)$
- Sound:
  - Yes: directly implements the definition of entailment
- Complete
  - Yes: given finite KB and α (because there are only finitely many models to examine)

Model Checking Example

- KB:
  (IsDog(Fido) ∨ IsCat(Fido)) ∧
  (IsCat(Fido) ⇔ Meows(Fido)) ∧
  (¬ Meows(Fido))

- Does $KB \models$ IsDog(Fido) ?

IsDog(Fido)	IsCat(Fido)	Meows(Fido)
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Model Checking Example

- KB:
  (IsDog(Fido) ∨ IsCat(Fido)) ∧
  (IsCat(Fido) ⇔ Meows(Fido)) ∧
  (¬ Meows(Fido))

- Does $KB \models$ IsDog(Fido) ?

IsDog(Fido)	IsCat(Fido)	Meows(Fido)
0	0	0
~~0~~	~~0~~	~~1~~
0	1	0
~~0~~	~~1~~	~~1~~
1	0	0
~~1~~	~~0~~	~~1~~
1	1	0
~~1~~	~~1~~	~~1~~

Model Checking Example

- KB:
  (IsDog(Fido) ∨ IsCat(Fido)) ∧
  (IsCat(Fido) ⇔ Meows(Fido)) ∧
  (¬ Meows(Fido))
- Does $KB \models IsDog(Fido)$?

IsDog(Fido)	IsCat(Fido)	Meows(Fido)
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Model Checking Example

- KB:
  (IsDog(Fido) ∨ IsCat(Fido)) ∧
  (IsCat(Fido) ⇔ Meows(Fido)) ∧
  (¬ Meows(Fido))
- Does $KB \models IsDog(Fido)$?

IsDog(Fido)	IsCat(Fido)	Meows(Fido)
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Model Checking Example

- KB:
  (IsDog(Fido) ∨ IsCat(Fido)) ∧
  (IsCat(Fido) ⇔ Meows(Fido)) ∧
  (¬ Meows(Fido))

- Does $KB \models$ IsDog(Fido) ?
  - For all world models in which KB is true, IsDog(Fido) is true, so YES!

IsDog(Fido)	IsCat(Fido)	Meows(Fido)
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Model Checking: Hindsight

- Our goal in PL was to be able to efficiently handle incomplete knowledge
  - Belief state becomes cumbersome!
  - But model checking has the same complexity…
- Model checking enumerates every state!
  - We haven't yet really "cashed in" on our new fancy representation of incomplete knowledge!

Proving without Model Checking

- Standard patterns of inference that can be applied to derive chains of conclusions.
- Pattern:

$$\frac{premise}{conclusion}$$

  - If KB contains *premise*, can add *conclusion*
- Provides a different (than enumeration) way of deciding entailment: by **proof**
  - A proof is a sequence of applications of sound inference rules.

Why Proof?

- Big win if the desired entailment can be derived without even caring about the truth values of most of the propositions!
- In our vacuum cleaner world, we might have 3 propositions: D_a, D_b, R_a.
  - If we know (D_a v D_b) and ¬ D_b then we should be able to conclude D_a without having to consider whether R is in a or not!
  - We can! Proof using resolution (you'll see!)
  - More pronounced savings with propositions about the door is open, the lights are on, the weather, etc.

Some Inference Rules

- Modus Ponens (MP) $\dfrac{\alpha \Rightarrow \beta, \alpha}{\beta}$
- And-elimination (AE) $\dfrac{\alpha \wedge \beta}{\alpha}$
- Biconditional-elim (BE) $\dfrac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$ $\dfrac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$
- Contrapositive (CP) $\dfrac{(\alpha \Rightarrow \beta)}{(\neg\beta \Rightarrow \neg\alpha)}$
- Many more sound rules… (verify through truth-tables)

DeMorgan's

- What do you have now?
  - Not of Ands: NA ¬(A ∧ B)
  - Not of Ors: NO ¬(A ∨ B)
  - Or of Nots: ON (¬A ∨ ¬B)
  - And of Nots: AN (¬A ∧ ¬B)
- Reverse the letters, swap the O's for A's.
  - NO ≡ AN
  - NA ≡ ON
  - AN ≡ NO
  - ON ≡ NA

Proof Example: Pit World

Q: $P_{1,2}$

B_{21}			
$\neg B_{11}$	??		

KB

S_1: $\neg P_{1,1}$

S_2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

S_3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

S_4: $\neg B_{1,1}$

S_5: $B_{2,1}$

[BE, S_2]

S_6: $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

[AE, S_6]

S_7: $(P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$

[contrapositive, S_7]

S_8: $\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1})$

[MP, S_4, S_8]

S_9: $\neg (P_{1,2} \vee P_{2,1})$

[de Morgan's, S_9]

S_{10}: $\neg P_{1,2} \wedge \neg P_{2,1}$

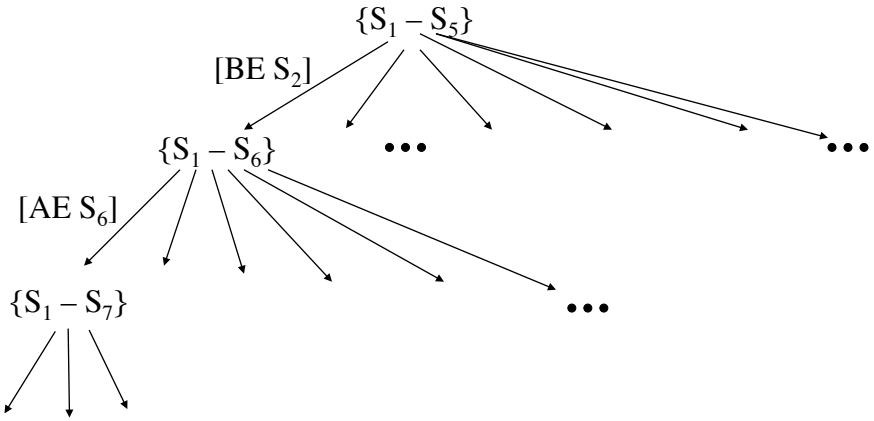
[AE S_{10}]

S_{10}: $\neg P_{1,2}$

Proof as Search

- State: Set of sentences
  - Initial state: KB
- Successors: all possible applications of sound inference rules
- Can use any search method
  - But do they all make sense?
  - Can be dramatically more efficient than model checking
- Inference in PL is NP-complete
  - In the worst-case searching for proofs is no more efficient than enumeration

Propositional Logic Proof Search Tree



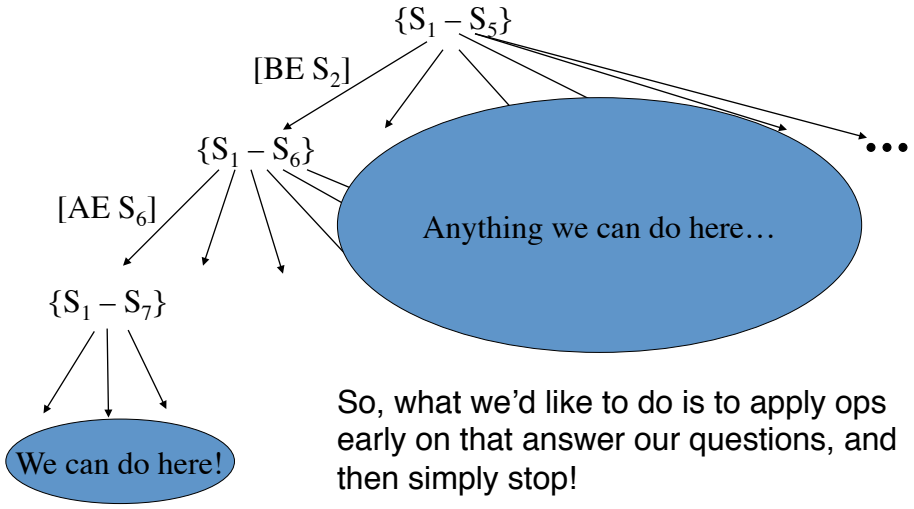
Monotonicity

- The set of entailed sentences can only increase as sentences are added to KB
- For any sentences α and β

$$\textit{if} \quad KB \models \alpha \quad \textit{then} \quad KB \wedge \beta \models \alpha$$

- Implications
  - Conclusions from sound inference rules are never *defeated* by further inference
  - Search method never needs to backtrack: no "branching"

Propositional Logic Proof Search Tree



So, what we'd like to do is to apply ops early on that answer our questions, and then simply stop!

(Unit) Resolution

$$\frac{\alpha \vee \beta, \neg \beta}{\alpha}$$

- "logically equivalent" to modus ponens
  - (-b => a)^(-b)
- Often expressed as requiring *literals*

$$\frac{(c \vee a_1 \vee a_2 \vee \ldots \vee a_j) \wedge \neg c}{a_1 \vee a_2 \vee \ldots \vee a_j}$$

Unit Resolution Example

- In our vacuum cleaner world, we might have 3 propositions: D_a, D_b, R_a.
  - If we know (D_a v D_b) and ¬ D_b then we should be able to conclude D_a without having to consider whether R is in a or not!
  - (Unit) resolution is an inference rule that allows this conclusion!

Full Resolution Rule

$$\frac{(c \vee a_1 \vee a_2 \vee \ldots \vee a_j) \wedge (\neg c \vee b_1 \vee b_2 \vee \ldots \vee b_k)}{a_1 \vee a_2 \vee \ldots \vee a_j \vee b_1 \vee b_2 \vee \ldots \vee b_k}$$

- Resulting clause should contain only one copy of each literal
  - i.e., combine any a's and b's that are the same literal

Soundness of Resolution

$$\frac{(c \vee a_1 \vee a_2 \vee \ldots \vee a_j) \wedge (\neg c \vee b_1 \vee b_2 \vee \ldots \vee b_k)}{a_1 \vee a_2 \vee \ldots \vee a_j \vee b_1 \vee b_2 \vee \ldots \vee b_k}$$

- Suppose c is false. Then:
  - $a_1 \vee a_2 \vee \ldots \vee a_j$ must be true.
- Suppose c is true. Then:
  - $b_1 \vee b_2 \vee \ldots \vee b_k$ must be true.
- C must be either true or false. Thus:
  - Either the a's must be true OR the b's must be true.

Conjunctive Normal Form

- Resolution rule applies directly only to disjunctions of literals
- Every PL sentence is logically equivalent to a conjunction of disjunction of literals (a CNF sentence)
- *k*-CNF has exactly *k* literals per clause
  $(l_{1,1} \vee \ldots \vee l_{1,k}) \wedge (l_{2,1} \vee \ldots \vee l_{2,k}) \wedge \ldots \wedge (l_{n,1} \vee \ldots \vee l_{n,k})$
- Every PL sentence can be transformed into a 3-CNF sentence

Conversion to CNF

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

1) Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ by $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

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

2) Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ by $\neg\alpha \vee \beta$

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

3) Ensure $\neg$ applies only to literals by moving inwards

   $[\neg(\neg\beta) \equiv \beta;\ \neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta;\ \neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta]$

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

4) Distribute $\vee$ over $\wedge$ wherever possible

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

Simple Full Resolution Example

- Some rooms are dirty:
  - S1: $(D_a \vee D_b)$
- Vacuuming cleans them:
  - S2: $(V_a \Rightarrow \neg D_a)$ converted to $(\neg V_a \vee \neg D_a)$
- Resolving S1 and S2 gives us:
  - $(\neg V_a \vee D_b)$

Refutation

$$\alpha \overset{?}{\vdash} \alpha \vee \beta$$

- *Cannot* derive $\alpha \vee \beta$ from α using resolution rule.
- However we *can* use resolution to prove the conclusion.
- Refutation proof:
  - Assume *negation* of goal sentence, derive a contradiction (false).
  - If successful, goal sentence must be entailed.

Resolution Refutation

- Show that $(KB \wedge \neg\alpha)$ is unsatisfiable.
- Convert $(KB \wedge \neg\alpha)$ to CNF
- Apply resolution rule to resulting clauses.
  - Each pair that contains complementary literals is resolved to produce a new clause
  - Add to the set if it is not already present
- Continues until one of two things happen
  - There are no new clauses that can be added, in which case KB does not entail α
  - An application of the resolution rule derives the empty clause, in which case KB does entail α

PL Resolution

function PL-resolution (KB, α) **returns** *true* or *false*
  inputs KB, the knowledge base, a sentence in PL
    α, the query, a sentence in PL
  clauses ← CNF representation of (KB ∧ ¬α)
  new ← {}
  loop do
    for each C_i, C_j **in** clauses **do**
      resolvents ← PL-resolve (C_i, C_j)
      if resolvents contains {} **then return** *true*
      new ← new ∪ resolvents
    if new ⊆ clauses **then return** *false* *// proved nothing new*
  clauses ← clauses ∪ new

Example

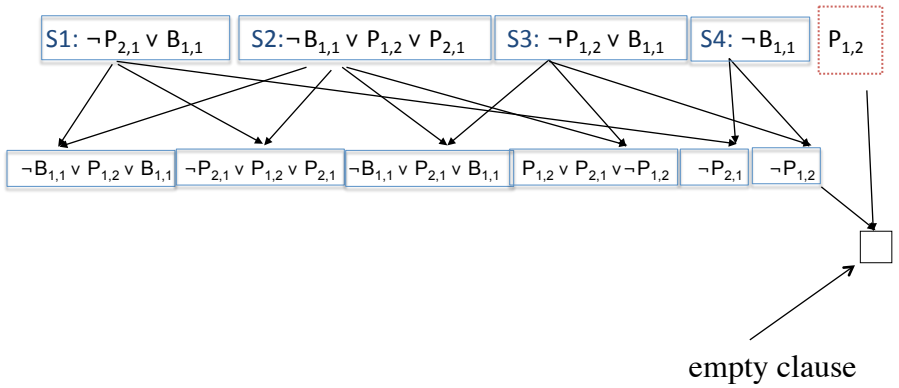
Given P, prove (PvQ):
- S1: P v False (right?)
- S2: ¬(PvQ) →
  - S2a: ¬P
  - S2b: ¬Q

Resolve S1 and S2a to get
- S3: False

- If we can ever derive the empty clause, we've derived that "false" is entailed (as a true statement) by the KB combined with the negated sentence to prove.
- Since False can't be true, there is a contradiction in the KB with the negated sentence.
- Since we assume the KB was correct, the contradiction must have been introduced by the negated sentence.

Example

Try to prove: $\neg P_{1,2}$



Your Turn

Prove Modus Ponens:

[P ^ (P=>Q)] => Q

- Step 1: We'll prove by refutation, so negate the hypothesis.
- Step 2: Convert to CNF.
- Step 3. Find a sequence of resolutions
  - We expect to find a contradiction.

Solution:

Negate it: ¬ { [P ^ (P=>Q)] => Q }

Convert to CNF:
- ¬ { [P ^ (P=>Q)] => Q }
- ¬ {¬[P ^ (¬P v Q)] v Q }
- {¬ ¬[P ^ (¬P v Q)] ^ ¬Q }
- {[P ^ (¬P v Q)] ^ ¬Q }
- P ^ (¬P v Q) ^ ¬Q
- S1: P
- S2: ¬P v Q
- S3: ¬Q

Resolve S1 and S2 to get
- S4: Q

Resolve S3 and S4 to get
- S5: FALSE

Negation is unsatisfiable, so non-negated must be valid.

Some Special Cases

- All sentences *conjunctions* of prop symbols
  - No disjunction or negation allowed
    - E.g., A^B^C^D^…
  - KB is a database
  - How do we do inference?
    - Answer queries by lookup
- All sentences of the form

$$P_1 \wedge P_2 \wedge \ldots \wedge P_n \Rightarrow Q$$

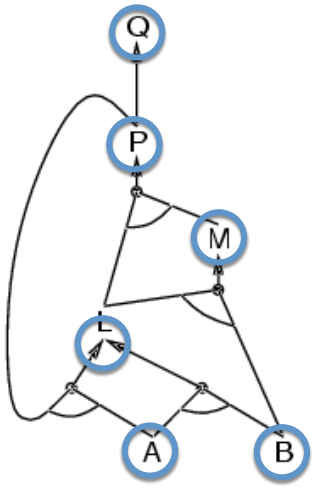
  - Horn sentences
    - At most one positive literal in CNF form. (Why is this the same as the form above?)
  - P_i a premise, Q the head
  - Forward chaining answers all queries in time linear in KB

Forward Chaining

- Maintain for each rule a *count* of unsatisfied premises
- Initialize *agenda* to the known facts
- Loop until empty *agenda*
  - $p \leftarrow$ Pop(*agenda*)
  - if p not already marked as *inferred*
    - mark p as *inferred*
    - decrement *count* for rules with p as a premise
    - when *count* = 0, put rule's head on *agenda*

Forward Chaining Example

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



Revenge of Model Checking

- Our earlier attempts at model checking were very expensive:
  - Consider every possible model (2^N)!
- Consider: A ^ (B v C)
- Couldn't we have determined the futility of A=0 earlier?

A	B	C	A ^(B v C)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Model Checking: Back Tracking

- Idea: Try to explicitly construct a model that satisfies our KB (as opposed to *enumerating* models)
  - BackTracking is a *satisfiability* algorithm
    - Use refutation to prove something!
- DPLL_search(KB, model)
  - If any clause in KB is false in model,
    - Return false
  - If all clauses in KB are true in model,
    - Return true
  - Find an unbound variable **v** in our KB
  - Return DPLL_search(KB, model U v=t) | DPLL_search(KB, model U v = f)

How do we initially invoke DPLL_search?

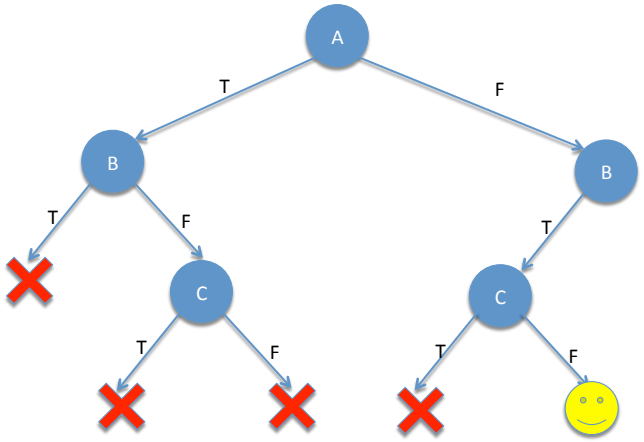
How do we evaluate clauses with a partial model?

How do we apply our CSP optimizations to this problem?

Back Tracking

KB:
¬A v ¬B
B v ¬C
¬A v C

Is KB satisfiable?



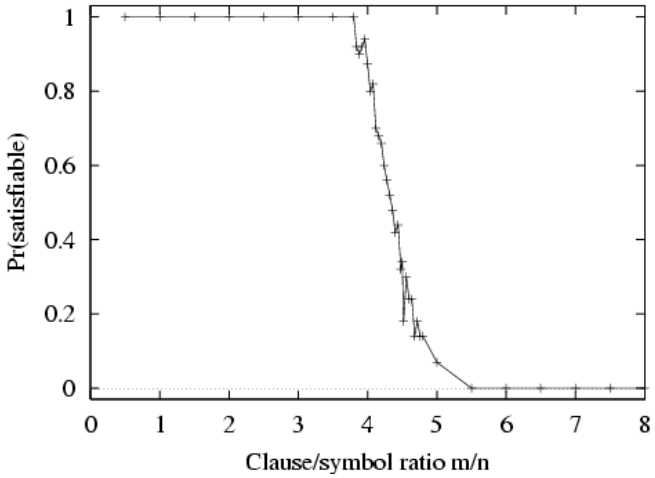
Hard Problems

- At the heart of inference is *satisfiability.*
  - We've been doing constraint satisfaction!
- Why is n-queens easy?
  - Somehow related to the density of solutions
- Consider random 3-CNF sentences. e.g.

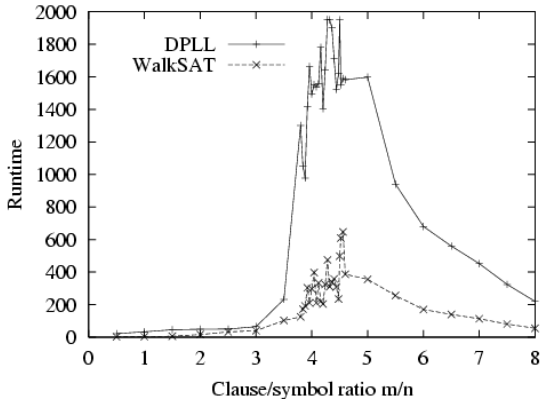
$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge$
$(\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge$
$(B \vee E \vee \neg C)$

  - Hard problems seem to cluster near *clauses/symbols* = 4.3 (critical point)

Hard satisfiability problems



Hard satisfiability problems



- Median runtime for 100 satisfiable random 3-CNF sentences, $n = 50$

Next Time

- First Order Logic
  - Enrich our language to understand objects and relations between those objects
    - Wumpus rules will be easier & less tedious to express