Logic in AI

CSE 573

Logistics

- Monday?
- Reading
  - Ch 8
  - Ch 9 thru p 278
  - Section 10.3
- Projects
  - Due 11/10
  - Teams and project plan due by this Fri

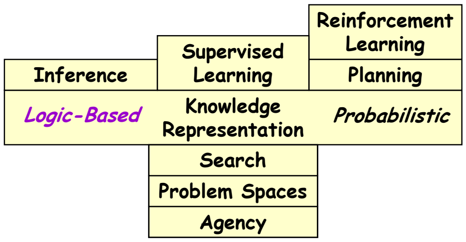
Search

Problem spaces
Blind
Depth-first, breadth-first, iterative-deepening, iterative broadening
Informed
Best-first, Dijkstra's, A*, IDA*, ~~SMA*~~, ~~DFB&B~~, Beam,
Local search
hill climbing, limited discrepancy, RTDP
Heuristics
Evaluation, construction via relaxation
Pattern databases
Constraint satisfaction
Adversary search

Takeaways

- Formulating a problem space (and a CSP!)
- Sampler of methods
- Importance of heuristics
- Speed / completeness tradeoff
- Space complexity

573 Topics



Today

- Review of Propositional Logic
- Inference Algorithms
  - As search: systematic & stochastic
- Themes
  - Expressivity *vs.*
  - Tractability

Some KR Languages

- Propositional Logic
- Predicate Calculus
- Frame Systems
- Rules with Certainty Factors
- Bayesian Belief Networks
- Influence Diagrams
- Semantic Networks
- Concept Description Languages
- Nonmonotonic Logic

In Fact…

- All popular knowledge representation systems are equivalent to (or a subset of)
  - Logic
    - Either Propositional Logic
    - Or Predicate Calculus
  - Probability Theory

What is Propositional Logic?

- And why have you studied it?

- And why are we torturing you again?

Basic Idea of Logic

- By starting with true assumptions, you can deduce true conclusions.

Truth

•Francis Bacon (1561-1626)
No pleasure is comparable to the standing upon the vantage-ground of truth.

•Thomas Henry Huxley (1825-1895)
Irrationally held truths may be more harmful than reasoned errors.

•John Keats (1795-1821)
Beauty is truth, truth beauty; that is all ye know on earth, and all ye need to know.

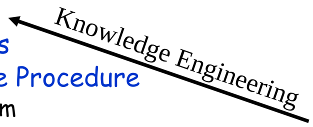
•Blaise Pascal (1623-1662)
We know the truth, not only by the reason, but also by the heart.

•François Rabelais (c. 1490-1553)
Speak the truth and shame the Devil.

•Daniel Webster (1782-1852)
There is nothing so powerful as truth, and often nothing so strange.

AI=Knowledge Representation & Reasoning

- Syntax
- Semantics
- Inference Procedure
  - Algorithm
  - Sound?
  - Complete?
  - Complexity



Propositional Logic

- Syntax
  - Atomic sentences: P, Q, …
  - Connectives: $\wedge$, $\vee$, $\neg$, $\Rightarrow$
- Semantics
  - Truth Tables
- Inference
  - Modus Ponens
  - Resolution
  - DPLL
  - GSAT
- Complexity

Propsitional Logic: Syntax

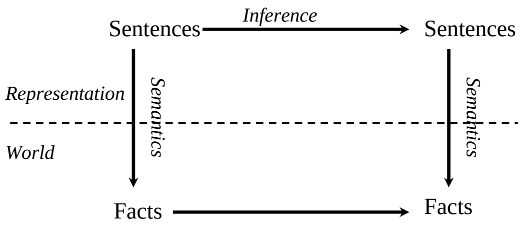
- Atoms
  - P, Q, R, …
- Literals
  - P, $\neg$ P
- Sentences
  - Any literal is a sentence
  - If S is a sentence
    - Then (S $\wedge$ S) is a sentence
    - Then (S $\vee$ S) is a sentence
- Conveniences
  - P $\supset$ Q same as $\neg$ P $\vee$ Q

Special Syntactic Forms

- General Form:
  - $((q \wedge \neg r) \supset s)) \wedge \neg (s \wedge t)$
- Conjunction Normal Form (CNF)
  - $(\neg q \vee r \vee s) \wedge (\neg s \vee \neg t)$
  - Set notation: { ($\neg$ q, r, s), ($\neg$ s, $\neg$ t) }
  - empty clause () = *false*
- Binary clauses: 1 or 2 literals per clause
  - $(\neg q \vee r)$ $(\neg s \vee \neg t)$
- Horn clauses: 0 or 1 positive literal per clause
  - $(\neg q \vee \neg r \vee s)$ $(\neg s \vee \neg t)$
  - $(q \wedge r) \supset s$ $(s \wedge t) \supset$ *false*

Semantics

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



Propositional Logic: **SEMANTICS**

- "Interpretation" (or "possible world")
  - Assignment to each variable either T or F
  - Assignment of T or F to each connective via defns

		Q	
		T	F
P	T		
	F		

P $\wedge$ Q

		Q	
		T	F
P	T		
	F		

P $\vee$ Q

		Q	
P	T		
	F		

$\neg$ P

Satisfiability, Validity, & Entailment

- S is **satisfiable** if it is true in ***some*** world
- S is **unsatisfiable** if it is false ***all*** worlds
- S is **valid** if it is true in ***all*** worlds
- S1 **entails** S2 if ***wherever*** S1 is true S2 is also true

Examples

P => Q

R => ¬R

S ∧ (W ∧ ¬S)

T ∨ ¬T

X => X

Notation

$\Rightarrow$

$\supset$ } **Implication** (syntactic symbol)

$\rightarrow$

$\vdash$ **Proves:** S1 $|-_{ie}$ S2 if \`ie' algorithm says \`S2' from S1

$\models$ **Entails:** S1 |= S2 if wherever S1 is true S2 is also true

- **Sound** $\vdash \supset \models$
- **Complete** $\models \supset \vdash$

Prop. Logic: **Knowledge Engr**

1) One of the women is a biology major
2) Lisa is not next to Dave in the ranking
3) Dave is immediately ahead of Jim
4) Jim is immediately ahead of a bio major
5) Mary or Lisa is ranked first

1. Choose Vocabulary

Universse: Lisa, Dave, Jim, Mary
LD = "Lisa is immediately ahead of Dave"
D = "Dave is a Bio Major"

2. Choose initial sentences (wffs)

Reasoning Tasks

- Model finding
  - KB = background knowledge
  - S = description of problem
  - Show (KB ∧ S) is satisfiable
  - A kind of constraint satisfaction
- Deduction
  - S = question
  - Prove that KB |= S
  - Two approaches:
    - Rules to derive new formulas from old (inference)
    - Show (KB ∧ ¬ S) is unsatisfiable

Propositional Logic: **Inference**

A mechanical process for computing new sentences

1. Backward & Forward Chaining
   Based on rule of ***modus ponens***

   If know $P_1, \ldots, P_n$ & know $(P_1 \wedge \ldots \wedge P_n) =>$ Q
   Then can conclude Q
2. Resolution (Proof by Contradiction)
3. GSAT
4. Davis Putnam

Inference 1: Forward Chaining

Forward (& Backward) Chaining
Based on rule of ***modus ponens***

If know $P_1, \ldots, P_n$ & know $(P_1 \wedge \ldots \wedge P_n) =>$ Q
Then can conclude Q

Pose as Search thru Problem Space?
States?
Operators?

Analysis

- Sound?
- Complete?

Can you prove
$\{ \} \models Q \lor \neg Q$

Special Syntactic Forms: CNF

- General Form:
  $((q \land \neg r) \supset s)) \land \neg (s \land t)$
- Conjunction Normal Form (CNF)
  $(\neg q \lor r \lor s) \land (\neg s \lor \neg t)$
  Set notation: $\{ (\neg q, r, s), (\neg s, \neg t) \}$
  empty clause () = *false*

Inference 2: Resolution

[Robinson 1965]

$\{ (p \lor \alpha), (\neg p \lor \beta \lor \gamma) \} \vdash_R (\alpha \lor \beta \lor \gamma)$

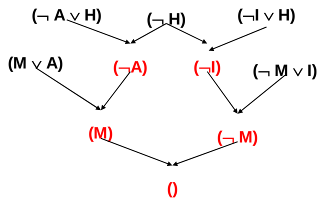
Correctness
 If $S1 \vdash_R S2$ then $S1 \models S2$
Refutation Completeness:
 If S is unsatisfiable then $S \vdash_R ()$

Resolution

If the unicorn is mythical, then it is immortal, but if it is not mythical, it is a mammal. If the unicorn is either immortal or a mammal, then it is horned.

Prove: the unicorn is horned.

M = mythical
I = immortal
A = mammal
H = horned



Resolution as Search

- States?
- Operators

Inference 3: Model Enumeration

```
for (m in truth assignments){
   if  (m makes Φ true)
   then return "Sat!"
 }
 return "Unsat!"
```

View as Search?
Critique?

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 U {P=0})) return true;
  return dpll_1(pa ∪ {P=1});
}
```

Returns true if F is satisfiable, false otherwise

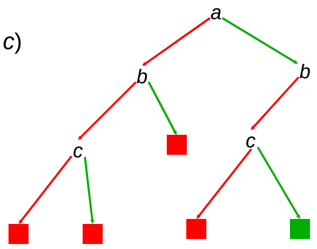
DPLL Version 1

$(a \vee b \vee c)$

$(a \vee \neg b)$

$(a \vee \neg c)$

$(\neg a \vee c)$



DPLL as Search

- Search Space?
- Algorithm?

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 (more)

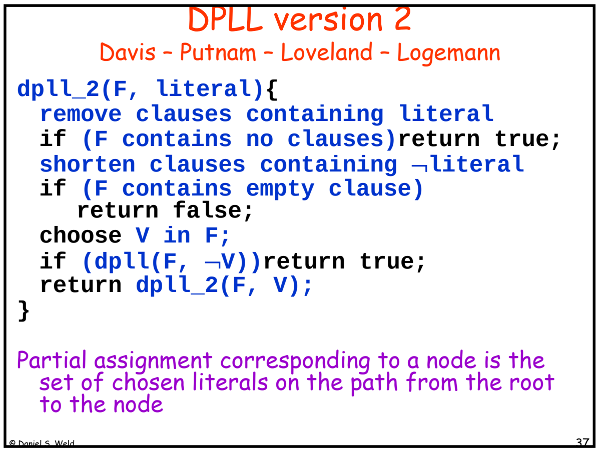
If literal L_1 is false, then clause $(L_1 \vee L_2 \vee L_3 \vee ...)$ has the same value as $(L_2 \vee L_3 \vee ...)$

Therefore: Okay to delete shorten containing false literals!

Observation!

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

Therefore: the empty clause means false!

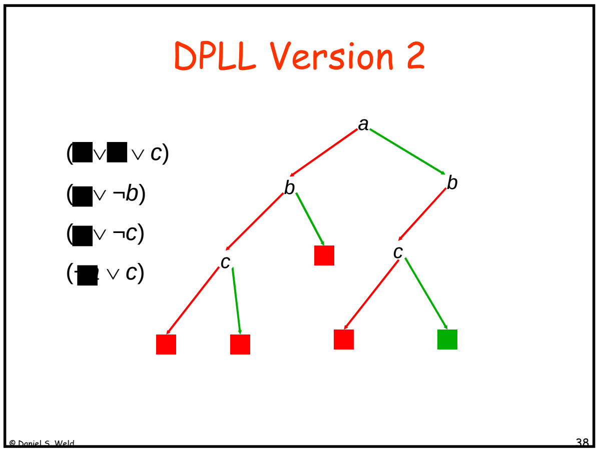


DPLL version 2

Davis – Putnam – Loveland – Logemann

```
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(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



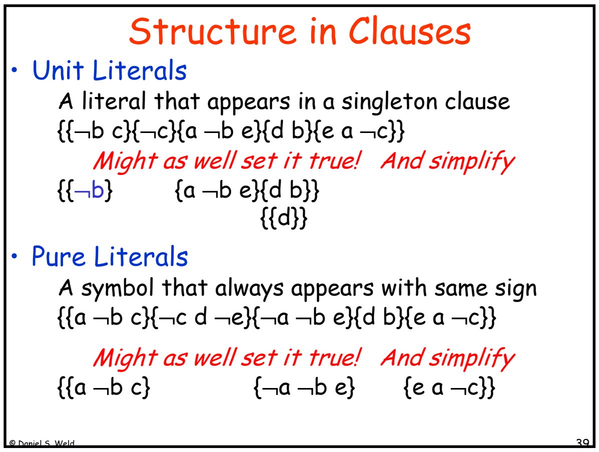
DPLL Version 2

([redacted] ∨ [redacted] ∨ c)

([redacted] ∨ ¬b)

([redacted] ∨ ¬c)

([redacted] ∨ c)



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}}

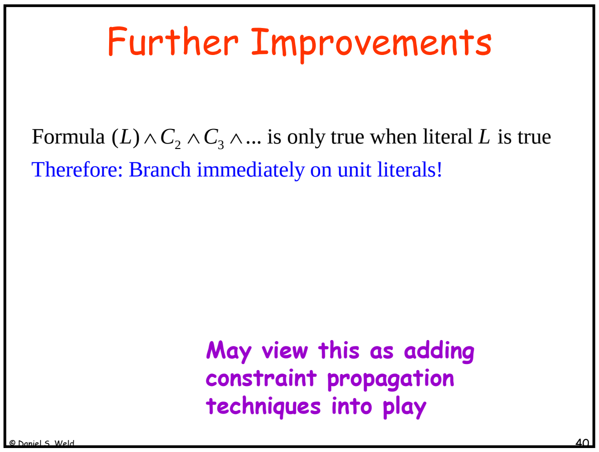
- Pure Literals

  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}}

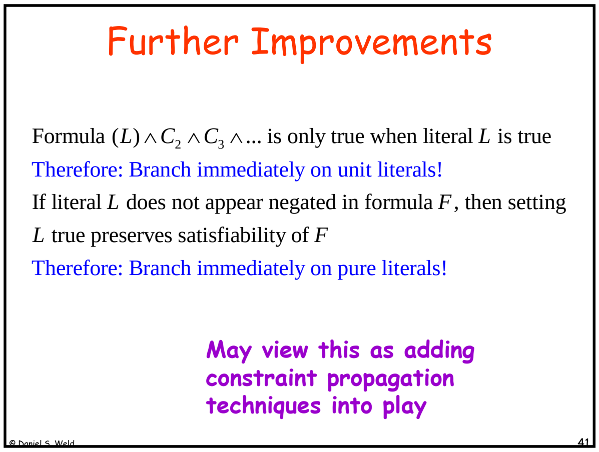


Further Improvements

Formula $(L) \wedge C_2 \wedge C_3 \wedge \ldots$ is only true when literal L is true

Therefore: Branch immediately on unit literals!

May view this as adding constraint propagation techniques into play



Further Improvements

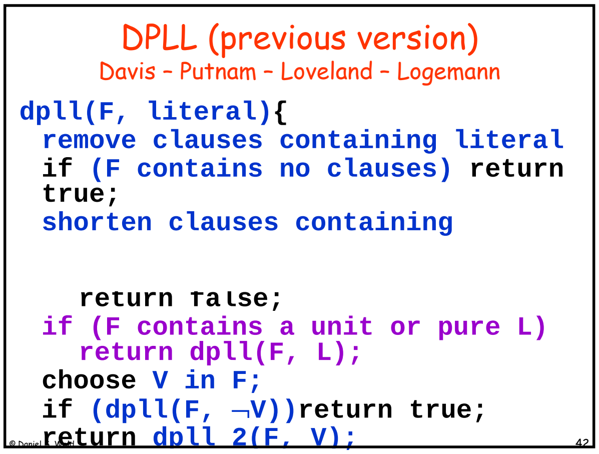
Formula $(L) \wedge C_2 \wedge C_3 \wedge \ldots$ is only true when literal L is true

Therefore: Branch immediately on unit literals!

If literal L does not appear negated in formula F, then setting L true preserves satisfiability of F

Therefore: Branch immediately on pure literals!

May view this as adding constraint propagation techniques into play



DPLL (previous version)

Davis – Putnam – Loveland – Logemann

```
dpll(F, literal){
  remove clauses containing literal
  if (F contains no clauses) return
  true;
  shorten clauses containing


     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_2(F, V);
```

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);
}
```

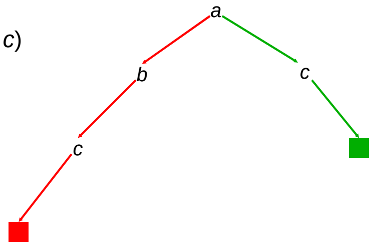
DPLL (for real)

([redacted] ∨ [redacted] ∨ *c*)

([redacted] ∨ ¬*b*)

([redacted] ∨ ¬*c*)

([redacted] ∨ *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);
}
```

Where could we use a heuristic to further improve performance?

What is the search space anyway?

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:
  - Most occurrences in clauses of minimum 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

Horn Theories

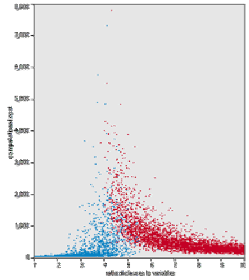
- Recall the special case of Horn clauses:
  - $\{ (\neg q \vee \neg r \vee s),\ (\neg s \vee \neg t) \}$
  - $\{ ((q \wedge r) \supset s),\ ((s \wedge t) \supset \textit{false}) \}$
- Many problems naturally take the form of such if/then rules
  - If (fever) AND (vomiting) then FLU
- Unit propagation is refutation complete for Horn theories
  - Good implementation – linear time!

WalkSat

- Local search over space of ***complete*** truth assignments
  - With probability P: flip any variable in any unsatisfied clause
  - With probability (1-P): flip best variable in any unsat clause
    - Like fixed-temperature simulated annealing
- SAT encodings of N-Queens, scheduling
- Best algorithm for random K-SAT
  - Best DPLL: 700 variables
  - Walksat: 100,000 variables

[Slide #s from 2001]

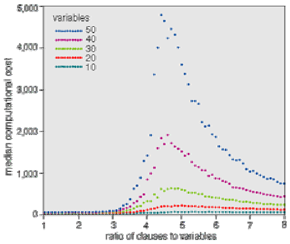
Random 3-SAT



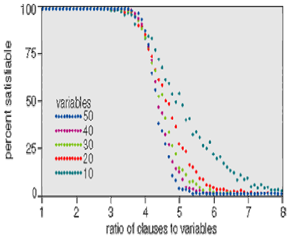
- Random 3-SAT
  - sample uniformly from space of all possible 3-clauses
  - *n* variables, *l* clauses
- Which are the hard instances?
  - around *l/n* = 4.3

Random 3-SAT

- Varying problem size, *n*
- Complexity peak appears to be largely invariant of algorithm
  - backtracking algorithms like Davis-Putnam
  - local search procedures like GSAT
- *What's so special about 4.3?*



Random 3-SAT



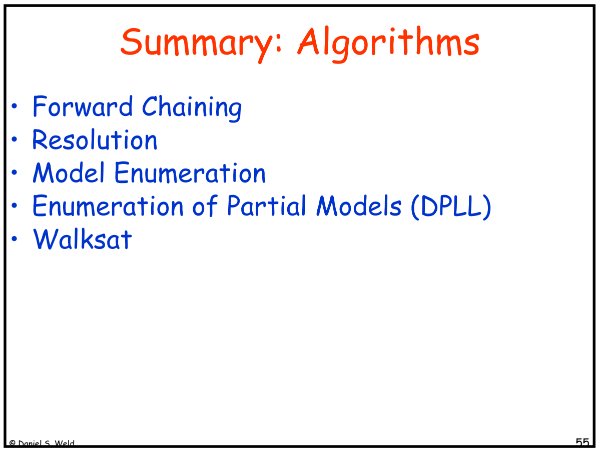
- Complexity peak coincides with solubility transition
  - l/n < 4.3 problems under-constrained and SAT
  - l/n > 4.3 problems over-constrained and UNSAT
  - l/n=4.3, problems on “knife-edge” between SAT and UNSAT

Project Issues

- DPLL *vs.* WalkSAT *vs.* ???
- Heuristics?
- Test problems?

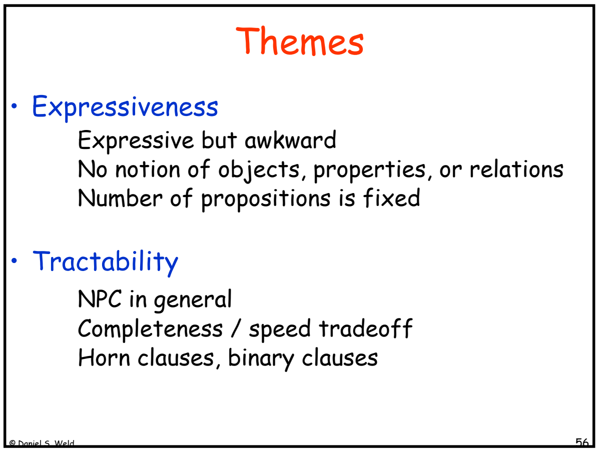
Real-World Phase Transition Phenomena

- Many NP-hard problem distributions show phase transitions -
  - job shop scheduling problems
  - TSP instances from TSPLib
  - exam timetables @ Edinburgh
  - Boolean circuit synthesis
  - Latin squares (alias sports scheduling)
- Hot research topic: predicting hardness of a given instance, & using hardness to control search strategy (Horvitz, Kautz, Ruan 2001-3)



Summary: Algorithms

- Forward Chaining
- Resolution
- Model Enumeration
- Enumeration of Partial Models (DPLL)
- Walksat



Themes

- Expressiveness

  Expressive but awkward
  No notion of objects, properties, or relations
  Number of propositions is fixed

- Tractability

  NPC in general
  Completeness / speed tradeoff
  Horn clauses, binary clauses