Knowledge Representation and Reasoning: Pure logical reasoning

Chapter 7.5, 7.6, Chapter 9

Recall propositional logic

$\neg$	
0	1
1	0

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

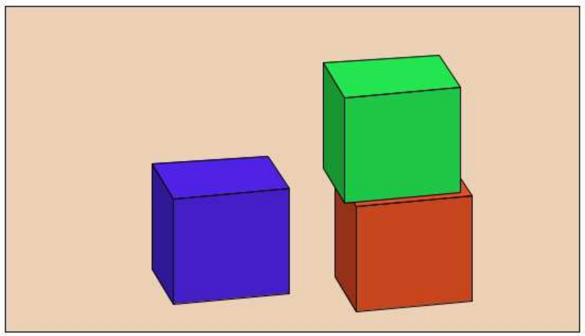
$\rightarrow$	0	1
0	1	1
1	0	1

$\leftrightarrow$	0	1
0	1	0
1	0	1

- ◊ Truth value of any propositional formula can be computed given an assignment of the values 1 (true) and 0 (false) to the atoms
- ◊ This computation is entirely deterministic and easy (linear time)
- ◊ Gives mechanical test for validity of inferences

SAT problems: examples 1

SAT representations of discrete problems
— Any case expressed as a set of yes/no decisions

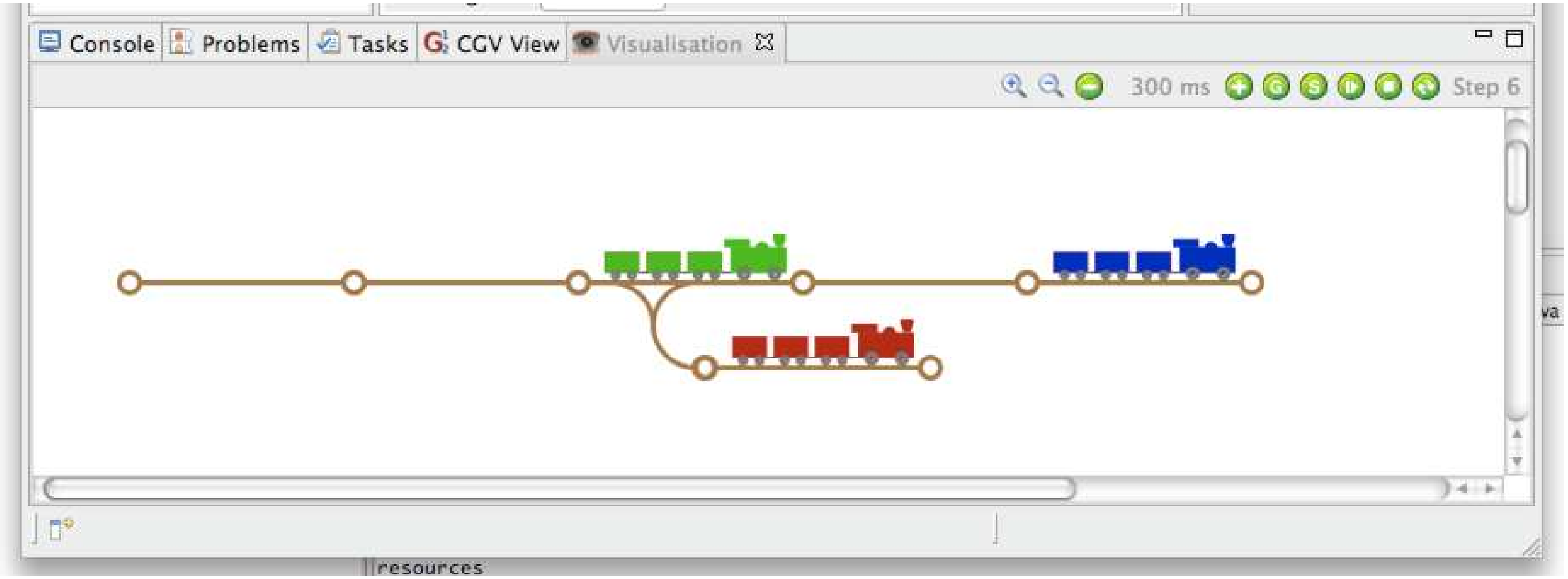


Atomic propositions to describe state
greenOnRed, blueOnGreen, etc.

More to describe possible moves
GreenToTable, blueToGreen, etc.

Can encode sequences of moves (e.g. plans) in this vocabulary

SAT problems: examples 2



◇ Meet-pass planning problems: getting the trains past each other using the given track sectors and the siding, obeying safety conditions

◇ First order problem representation is quite easy

◇ Reduces to SAT because everything is finite

◇ Still a “toy” example (270 atomic formulae) but closer to reality

SAT applications

◇ Industrial scale problems with thousands of variables (or more)

◇ Some obviously discrete problems
— circuit analysis
— model checking for hardware / software verification
— classical planning
— diagnosis
— combinatorial design (experiments, cryptography, drug design, etc)

◇ Often used for sub-problems
— Generating test patterns
— Scheduling (applied in many domains)
— Design and analysis of protocols

SAT: the bad news

◊ Number of possible truth-value assignments grows exponentially
— with n atoms, 2^n assignments of values (possible worlds)
— 2^{2^n} sets of possible worlds (truth functions / propositions)
— Testing for satisfiability (SAT) is (probably) hard in the worst case

◊ Important SAT problems have <u>thousands</u> of variables – even <u>millions</u>

◊ Brute force is hopeless!

◊ SAT is the classic NP-complete problem
— All known solution methods require exponential time
— Generally taken to be intractable

SAT: the better news

- Work towards intelligent search for solutions
- First step: simplify the structure of formulae
- $A \leftrightarrow B$ equivalent to $(A \land B) \lor (\neg A \land \neg B)$
- $A \to B$ equivalent to $\neg A \lor B$
- Every formula has an equivalent using $\land$, $\lor$ and $\neg$ only

  Example:

  $(p \to q) \to (r \land \neg(p \land \neg s))$

  $\neg(p \to q) \lor (r \land \neg(p \land \neg s))$

  $\neg(\neg p \lor q) \lor (r \land \neg(p \land \neg s))$

SAT: better news continues

- $\neg(A \wedge B)$ equivalent to $\neg A \vee \neg B$
- $\neg(A \vee B)$ equivalent to $\neg A \wedge \neg B$
- $\neg\neg A$ equivalent to A
- Every formula has an equivalent using $\wedge$, $\vee$ and $\neg$ only, in which negation ($\neg$) applies only to atoms
- This is **Negation Normal Form** (NNF)

$$\neg(\neg p \vee q) \vee (r \wedge \neg(p \wedge \neg s))$$

$$(\neg\neg p \wedge \neg q) \vee (r \wedge \neg(p \wedge \neg s))$$

$$(p \wedge \neg q) \vee (r \wedge \neg(p \wedge \neg s))$$

$$(p \wedge \neg q) \vee (r \wedge (\neg p \vee \neg\neg s))$$

$$(p \wedge \neg q) \vee (r \wedge (\neg p \vee s))$$

SAT: better and better news

- $A \land B$ equivalent to $B \land A$
- $A \lor B$ equivalent to $B \lor A$
- $(A \land B) \lor C$ equivalent to $(A \lor C) \land (B \lor C)$
- $(A \lor B) \land C$ equivalent to $(A \land C) \lor (B \land C)$
- Every formula has an equivalent which is a conjunction ($\land$) of disjunctions ($\lor$) of literals (atoms and negated atoms)
- This is **Conjunctive Normal Form** (CNF), aka **Clause Form**
- So any technique for reasoning with clauses can do propositional logic

Reduction to CNF: example

$(p \to q) \to (r \wedge \neg(p \wedge \neg s))$

reduces to NNF

$(p \wedge \neg q) \vee (r \wedge (\neg p \vee s))$

then moving conjunction outside disjunction:

$(p \vee (r \wedge (\neg p \vee s))) \wedge (\neg q \vee (r \wedge (\neg p \vee s)))$

$(p \vee r) \wedge (p \vee \neg p \vee s) \wedge (\neg q \vee r) \wedge (\neg q \vee \neg p \vee s)$

Second conjunct is a tautology, so can be deleted without loss, giving a set of clauses equivalent to the original formula:

$p \vee r$
$\neg q \vee r$
$\neg q \vee \neg p \vee s$

Resolution

Resolution is a logical inference rule which operates on clauses:

$$\frac{p_1 \vee \ \dots \ \vee p_n \vee q \qquad\qquad \neg q \vee r_1 \vee \ \dots \ \vee r_m}{p_1 \vee \ \dots \ \vee p_n \vee r_1 \vee \ \dots \ \vee r_m}$$

Alternatively, looking at a clause as a set of literals:

$$\frac{\Gamma \qquad\qquad\qquad\qquad \Delta}{(\Gamma \setminus \{q\}) \cup (\Delta \setminus \{\neg q\})}$$

Resolution derivation (example)

Show $\{p \lor q,\ p \lor \neg q,\ \neg p \lor r,\ \neg r \lor s,\ \neg r \lor \neg s\}$ unsatisfiable

1.	$p \lor q$	given	
2.	$p \lor \neg q$	given	
3.	$\neg p \lor r$	given	
4.	$\neg r \lor s$	given	
5.	$\neg r \lor \neg s$	given	
6.	p	1, 2	(with factoring to reduce $p \lor p$ to p)
7.	$\neg r$	4, 5	(with factoring)
8.	r	3, 6	
9.	$\perp$	7, 8	

A better idea: DPLL

- ♢ Any assignment satisfying a set Γ of clauses must make any specific atom p that occurs in Γ either true or false.

- ♢ Therefore Γ is satisfiable iff either $\Gamma \cup \{p\}$ is satisfiable or else $\Gamma \cup \{\neg p\}$ is satisfiable.

- ♢ Let Γ' be Γ with all clauses containing literal p deleted, and with $\neg p$ removed from all clauses in which it occurs. Then Γ' is satisfiable iff $\Gamma \cup \{p\}$ is satisfiable. Note that Γ' contains
  — fewer clauses than Γ
  — shorter clauses than Γ
  — fewer atoms than Γ

- ♢ The same holds for Γ'', defined similarly using $\neg p$ instead of p.

- ♢ Therefore the problem of deciding whether Γ is satisfiable can be replaced by the two strictly simpler problems of deciding satisfiability of Γ' and Γ''.

Unit propagation

- ◇ A pure literal is one whose complement does not appear anywhere
- ◇ Obviously any pure literal can be made true without bad consequences
- ◇ Therefore any clause containing a pure literal may be deleted
- ◇ A unit clause is a clause consisting of only one literal
- ◇ Obviously this literal has to be set to true
- ◇ Therefore its complement can be deleted from all clauses
  — Literal is then pure and triggers purity deletion
- ◇ Iterating these inference moves is unit propagation
- ◇ DPLL amounts to splitting plus unit propagation

Improving DPLL

◇ Clause learning
— The search backtracks when it runs into a contradiction
— The decisions determining the branch can't all be right
— Add complements of [a subset of] the chosen literals as a new clause
— So we never backtrack twice for the same reason

◇ Choosing good atoms for branching
— E.g. one that occurs most often in shortest clauses (MOMS)
— Or one that occurs most often in currently satisfied clauses

◇ Intelligent backtracking
— Can obviously jump back to a variable in the latest nogood
— May pay to jump back further

◇ Restarts
— Can jump right back to the root of the search tree and probe it
— Depends heavily on learned clauses to prevent repeated work

What about quantifiers?

◇ Sometimes need to reason about large or unspecified domains

◇ Reduction to SAT not possible in such cases

◇ Trivial example: subset transitivity:

$$\forall x \forall y (\mathsf{sub}(x, y) \leftrightarrow \forall z (\mathsf{in}(z, x) \rightarrow \mathsf{in}(z, y)))$$

therefore

$$\forall x \forall y \forall z ((\mathsf{sub}(x, y) \wedge \mathsf{sub}(y, z)) \rightarrow \mathsf{sub}(x, z))$$

Prenex normal form

◊ First problem: get all quantifiers to the front
Assume $\rightarrow$ and $\leftrightarrow$ rewritten using $\wedge$, $\vee$ and $\neg$

◊ Moving quantifiers outside negation
$\neg\forall x A$ equivalent to $\exists x \neg A$
$\neg\exists x A$ equivalent to $\forall x \neg A$
So quantifiers may switch between universal and existential

◊ Moving quantifier binding x outside another one. E.g.:
$\forall x A(x) \vee \forall x B(x)$ goes to $\forall x(A(x) \vee \forall x B(x))$
Solution: rewrite variables:
$\forall x(A(x) \vee \forall y B(y))$
$\forall x \forall y(A(x) \vee B(y))$

Removing the quantifiers

◇ Existential quantifiers removed by Skolemisation
— Variable replaced by a new name or function
— Then quantifier deleted

E.g. $\exists x \forall y \exists z R(x, y, z)$

goes to $\forall y \exists z R(a, y, z)$

then to $\forall y R(a, y, f(y))$

◇ All quantifiers are now universal. They can be removed
— Free variables are implicitly universal

◇ Note: Skolemised formula not equivalent to the original, but they are satisfiable if and only if the original is.

◇ Quantifier-free formula can be put into clause form

First order resolution

♢ Resolution applies to first order clauses too

♢ Usually requires **unification**: substituting terms for variables in order to make literals match

E.g.	$P(x, a) \vee \neg Q(x)$ and $\neg P(b, y) \vee R(y)$
unifier	$[x \leftarrow b, y \leftarrow a]$
gives	$P(b, a) \vee \neg Q(b)$ and $\neg P(b, a) \vee R(a)$
Resolvent:	$\neg Q(b) \vee R(a)$

Example: subset transitivity (1)

$\forall x \forall y(\mathsf{sub}(x, y) \leftrightarrow \forall z(\mathsf{in}(z, x) \rightarrow \mathsf{in}(z, y)))$

$\neg \forall x \forall y \forall z((\mathsf{sub}(x, y) \wedge \mathsf{sub}(y, z)) \rightarrow \mathsf{sub}(x, z))$

clausifies to

$\neg \mathsf{sub}(x, y) \vee \neg \mathsf{in}(z, x) \vee \mathsf{in}(z, y)$

$\mathsf{in}(f(x, y), x) \vee \mathsf{sub}(x, y)$

$\neg \mathsf{in}(f(x, y), y) \vee \mathsf{sub}(x, y)$

$\mathsf{sub}(a, b)$

$\mathsf{sub}(b, c)$

$\neg \mathsf{sub}(a, c)$

Example: subset transitivity (2)

1.	$\neg\mathsf{sub}(x, y) \vee \neg\mathsf{in}(z, x) \vee \mathsf{in}(z, y)$	given
2.	$\mathsf{in}(f(x, y), x) \vee \mathsf{sub}(x, y)$	given
3.	$\neg\mathsf{in}(f(x, y), y) \vee \mathsf{sub}(x, y)$	given
4.	$\mathsf{sub}(a, b)$	given
5.	$\mathsf{sub}(b, c)$	given
6.	$\neg\mathsf{sub}(a, c)$	given
7.	$\neg\mathsf{in}(z, a) \vee \mathsf{in}(z, b)$	from 1, 4 $[x \leftarrow a, y \leftarrow b]$
8.	$\neg\mathsf{in}(z, b) \vee \mathsf{in}(z, c)$	from 1, 5 $[x \leftarrow b, y \leftarrow c]$
9.	$\mathsf{in}(f(a, c), a)$	from 2, 6 $[x \leftarrow a, y \leftarrow c]$
10.	$\neg\mathsf{in}(f(a, c), c)$	from 3, 6 $[x \leftarrow a, y \leftarrow c]$
11.	$\mathsf{in}(f(a, c), b)$	from 7, 9 $[z \leftarrow f(a, c)]$
12.	$\mathsf{in}(f(a, c), c)$	from 8, 11 $[z \leftarrow f(a, c)]$
13.	$\perp$	from 10, 12

Summary

♢ Problems from many domains can be coded as SAT
— Discrete, finite, not too much arithmetic

♢ Intelligent solution methods dominate brute force

♢ Reduction to clause form
— Apply logical equivalences: DeMorgan's laws, distribution

♢ Simple inference rules operate on clauses
— Resolution (not much used for pure SAT problems)
— DPLL and its variants generally preferred
— SAT solvers now useful for real industrial problems

♢ Normal forms also for first order logic
— Prenex, skolem, clause form

♢ Resolution is more useful at the first order level — Resolution-like methods are the state of the art