Search Algorithms

Backtrack Search
1. DFS
2. BFS / Dijkstra's Algorithm
3. Iterative Deepening
4. Best-first search
5. A*

Constraint Propagation
1. Forward Checking
2. k-Consistency
3. DPLL & Resolution

Local Search
1. Hillclimbing
2. Simulated annealing
3. Walksat

Guessing versus Inference

All the search algorithms we've seen so far are variations of guessing and backtracking

But we can reduce the amount of guesswork by doing more reasoning about the consequences of past choices
- Example: planning a trip

Idea:
- Problem solving as constraint satisfaction
- As choices (guesses) are made, propagate constraints

Map Coloring

CSP

- V is a set of variables v1, v2, …, vn
- D is a set of finite domains D1, D2, …, Dn
- C is a set of constraints C1, C2, …, Cm

Each constraint specifies a *restriction* over *joint values* of a subset of the variables

E.g.:

v1 is Spain, v2 is France, v3 is Germany, …

Di = { Red, Blue, Green} for all i

For each adjacent vi, vj there is a constraint Ck

(vi,vj) ∈ { (R,G), (R,B), (G,R), (G,B), (B,R), (B,G) }

Variations

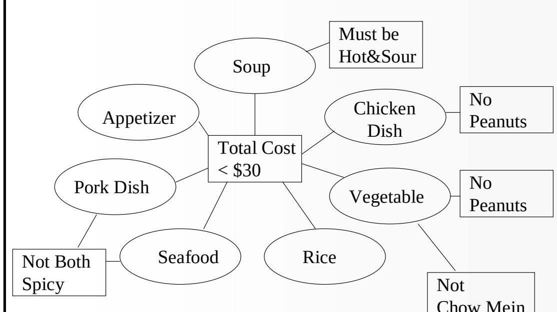
- Find a solution that satisfies all constraints
- Find all solutions
- Find a "tightest form" for each constraint

  (v1,v2) ∈ { (R,G), (R,B), (G,R), (G,B), (B,R), (B,G) }

  ➔

  (v1,v2) ∈ { (R,G), (R,B), (B,G) }
- Find a solution that minimizes some additional *objective function*

Chinese Dinner Constraint Network



Exploiting CSP Structure

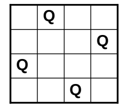
Interleave inference and guessing

- At each *internal* node:
  - Select unassigned variable
  - Select a value in domain
  - Backtracking: try another value
    - Branching factor?
- At each node:
  - *Propagate Constraints*

Running Example: 4 Queens

Variables:

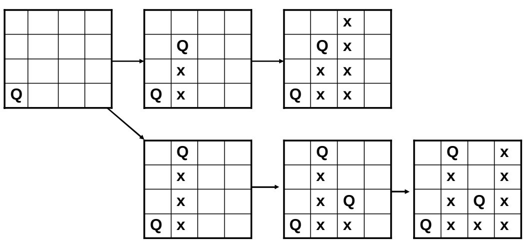
Q1 ∈ {1,2,3,4}
Q2 ∈ {1,2,3,4}
Q3 ∈ {1,2,3,4}
Q3 ∈ {1,2,3,4}



Constraints:

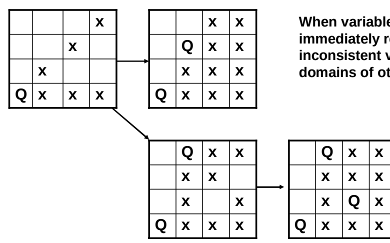
Q1	Q2
1	3
1	4
2	4
3	1
4	1
4	2

Constraint Checking



Takes 5 guesses to determine first guess was wrong

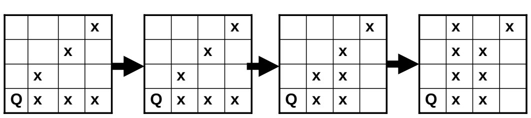
Forward Checking



When variable is set, immediately remove inconsistent values from domains of other variables

Takes 3 guesses to determine first guess was wrong

Arc Consistency



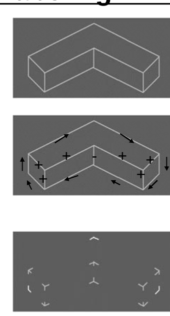
Iterate forward checking

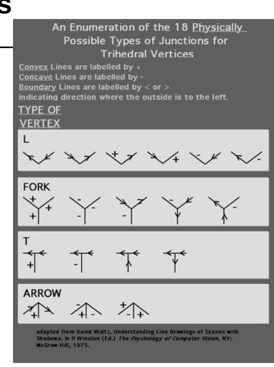
Propagations:

1. Q3=3 inconsistent with Q4 ∈ {2,3,4}
2. Q2=1 and Q2=2 inconsistent with Q3 ∈ {1}

Inference alone determines first guess was wrong!

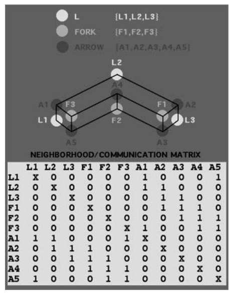
Huffman-Clowes Labeling





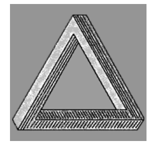
Waltz's Filtering: Arc-Consistency

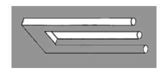
- •Lines: variables
- •Conjunctions: constraints
- •Initially Di = {+,-, ←, →)
- •Repeat until no changes:
  - Choose edge (variable)
  - Delete labels on edge not consistent with both endpoints



	L1	L2	L3	F1	F2	F3	A1	A2	A3	A4	A5
L1	X	0	0	0	0	0	1	0	0	0	1
L2	0	X	0	0	0	0	1	1	0	0	0
L3	0	0	X	0	0	0	0	1	1	0	0
F1	0	0	0	X	0	0	0	1	1	1	0
F2	0	0	0	0	X	0	0	0	1	1	1
F3	0	0	0	0	0	X	1	0	0	1	1
A1	1	1	0	0	0	1	X	0	0	0	0
A2	0	1	1	1	0	0	0	X	0	0	0
A3	0	0	1	1	1	0	0	0	X	0	0
A4	0	0	0	1	1	1	0	0	0	X	0
A5	1	0	0	0	1	1	0	0	0	0	X

No labeling!





Path Consistency

Path consistency (3-consistency):

- **Check every triple of variables**
- **More expensive!**
- **k-consistency:**

$|V|^k$ k-tuples to check

Worst case: each iteration eliminates 1 choice

$|D||V|$ iterations

$|D||V|^{k+1}$ steps! (But usually not this bad)

- ***n*-consistency: backtrack-free search**

Variable and Value Selection

- **Select variable with smallest domain**
  - Minimize branching factor
  - Most likely to propagate: most constrained variable heuristic
- **Which values to try first?**
  - Most likely value for solution
  - Least propagation! Least constrained value
- **Why different?**
  - Every constraint must be eventually satisfied
  - Not every value must be assigned to a variable!
- **Tie breaking?**
  - In general randomized tie breaking best – less likely to get stuck on same bad pattern of choices

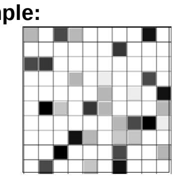
CSPs in the real world

- Scheduling Space Shuttle Repair
- Transportation Planning
- Computer Configuration
  - AT&T CLASSIC Configurator
    - #5ESS Switching System
    - Configuring new orders: 2 months → 2 hours

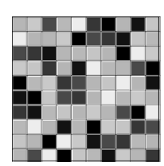
Quasigroup Completion Problem (QCP)

Given a partial assignment of colors (10 colors in this case), can the partial quasigroup (latin square) be completed so we obtain a full quasigroup?

Example:



32% preassignment



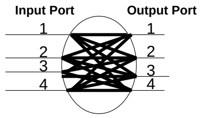
(Gomes & Selman 97)

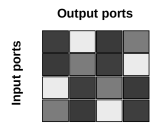
©PGomes - AAAI00

QCP Example Use: Routers in Fiber Optic Networks

Dynamic wavelength routing in Fiber Optic Networks can be directly mapped into the Quasigroup Completion Problem.

- **each channel cannot be repeated in the same input port (row constraints);**
- **each channel cannot be repeated in the same output port (column constraints);**





CONFLICT FREE LATIN ROUTER

(Barry and Humblet 93, Cheung et al. 90, Green 92, Kumar et al. 99)

CPGomes - AAAI00

QCP as a CSP

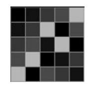
- **Variables -** $O(n^2)$

$$x_{i,j} \;\; color\ of\ cell\ i,j; \;\; i,j=1,2,...,n.$$

$$x_{i,j} \in \{1,2,...,n\}$$

- **Constraints -** $O(n)$

$$alldiff\,(x_{i,1}, x_{i,2}, ..., x_{i,n}); \;\; i=1,2,...,n. \quad \text{row}$$

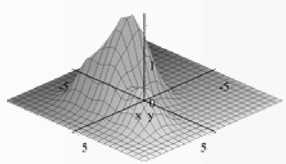


$$alldiff\,(x_{1,j}, x_{2,j}, ..., x_{n,j}); \;\; j=1,2,...,n. \quad \text{column}$$

CPGomes - AAAI00

Hill Climbing

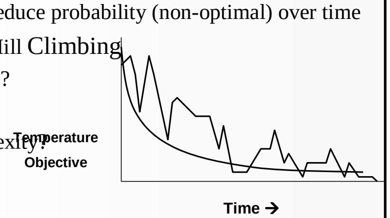
- Idea
  - Always choose best child, no backtracking
- Evaluation
  - Complete?



  - Space Complexity?
  - Complexity of random restart hillclimbing, with success probability P

Simulated Annealing / Random Walk

- Objective: avoid local minima
- Technique:
  - For the most part use hill climbing
  - Occasionally take non-optimal step
  - Annealing: Reduce probability (non-optimal) over time
- Comparison to Hill Climbing
  - Completeness?
  - Speed?
  - Space Complexity?



Backtracking with Randomized Restarts

- Idea:
  - If backtracking algorithm does not find solution quickly, it is like to be stuck in the wrong part of the search space
    - Early decisions were bad!
  - So kill the run after T seconds, and restart
    - Requires randomized heuristic, so choices not always the same
  - Why does it often work?
    - Many problems have a small set of "backdoor" variables – guess them on a restart, and your are done! (Andrew, Selman, Gomes 2003)
  - Completeness?

Demos!

- N-Queens
  Backtracking vs. Local Search
- Quasigroup Completion
  Randomized Restarts
- Travelling Salesman
  Simulated Annealing

Exercise

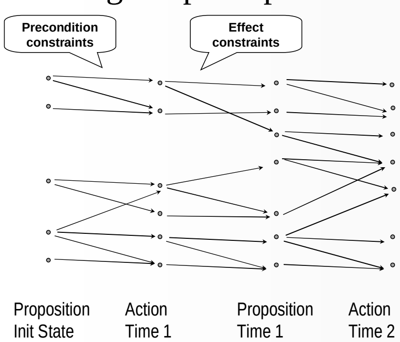
Peer interviews: Real-world constraint satisfaction problems

1. **Break into pairs**
2. **7 minute interview – example of needing to solve a CSP type problem (work or life). Interviewer takes notes:**
   - **Describe problem**
   - **What techniques actually used**
   - **Any techniques from class that could have been used?**
3. **Switch roles**
4. **A few teams present now**
5. **Hand in notes (MSR – have someone collect and mail to me at dept)**

Planning as CSP

- Phase 1 - Convert **planning problem in a CSP**
  - **Choose a fixed plan length**
  - **Boolean variables**
    - **Action executed at a specific time point**
    - **Proposition holds at a specific time point**
  - **Constraints**
    - **Initial conditions true in first state, goals true in final state**
    - **Actions do not interfere**
    - **Relation between action, preconditions, effects**
- Phase 2 - Solution Extraction
  - Solve the CSP

Planning Graph Representation of CSP



Constructing the planning graph…

- Initial proposition layer
  – Just the initial conditions
- Action layer i
  – If all of an action’s preconditionss are in i-1
  – Then add action to layer I
- Proposition layer i+1
  – For each action at layer i
  – Add all its effects at layer i+1

Mutual Exclusion

- Actions A,B ***exclusive (at a level)*** if
  – A deletes B’s precondition, or
  – B deletes A’s precondition, or
  – A & B have inconsistent preconditions
- Propositions P,Q ***inconsistent (at a level)*** if
  – All ways to achieve P exclude all ways to achieve Q
- Constraint propagation (arc consistency)
  – Can force variables to become true or false
  – Can create new mutexes

Solution Extraction

- For each goal G at last time slice N:
- Solve(G, N)

- Solve(G, t):

  CHOOSE action A making G true @t that is not mutex with a previously chosen action

  If no such action, backtrack to last choice point

  For each precondition P of A:

  Solve(P, t-1)

Graphplan

- Create level 0 in planning graph
- Loop
  - If goal $\subseteq$ contents of highest level (nonmutex)
  - Then search graph for solution
    - If find a solution then return and terminate
  - Else Extend graph one more level

A kind of double search: forward direction checks necessary (but insufficient) conditions for a solution, ... Backward search verifies...

31

Dinner Date

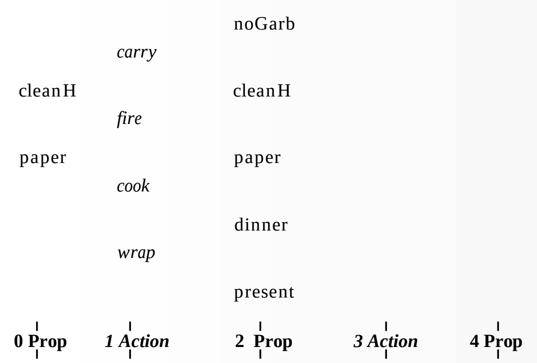
<u>Initial Conditions</u>: (:and (cleanHands) (quiet))

<u>Goal</u>: (:and (noGarbage) (dinner) (present))

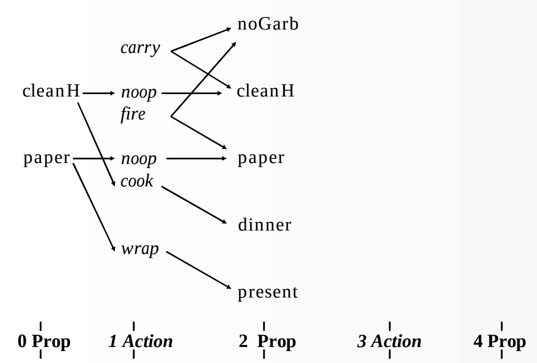
<u>Actions</u>:

(:operator **carry** *:precondition*
:effect (:and (noGarbage) (:not (cleanHands)))
(:operator **fire** *:precondition*
:effect (:and (noGarbage) (:not (paper)))
(:operator **cook** *:precondition* (cleanHands)
:effect (dinner))
(:operator **wrap** *:precondition* (paper)
:effect (present))

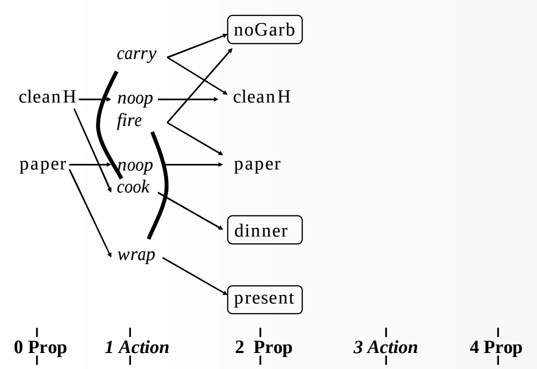
Planning Graph



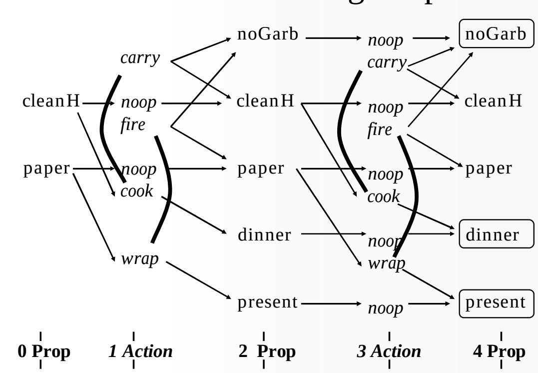
Are there any exclusions?



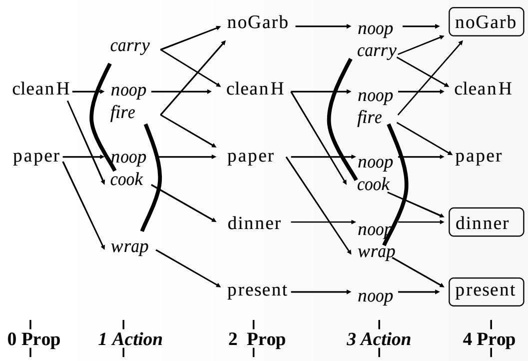
Do we have a solution?



Extend the Planning Graph



One (of 4) Solutions



Search Algorithms

Backtrack Search
1. DFS
2. BFS / Dijkstra's Algorithm
3. Iterative Deepening
4. Best-first search
5. A*

Constraint Propagation
1. Forward Checking
2. k-Consistency
3. DPLL & Resolution

Local Search
1. Hillclimbing
2. Simulated annealing
3. Walksat

Representing Knowledge in Propositional Logic

R&N Chapter 7

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.

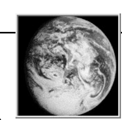
John Keats (1795-1821)
Beauty is truth, truth beauty; that is all
Ye know on earth, and all ye need to know.

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.

Propositional Logic



Ingredients of a sentence:
1. **Propositions (variables)**
2. **Logical Connectives ¬, ∧, ∨, ⊃**
   literal = a variable or a negated variable

- A possible world assigns every proposition the value true or false
- A truth value for a sentence can be derived from the truth value of its propositions by using the truth tables of the connectives
- The meaning of a sentence is the set of possible worlds in which it is true

Truth Tables for Connectives

¬

0	
1	

∧

0	0	
0	1	
1	0	
1	1	

∨

0	0	
0	1	
1	0	
1	1	

⊃

0	0	
0	1	
1	0	
1	1	

Special Syntactic Forms

- **General PL:**
  ((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*

Satisfiability, Validity, & Entailment

- **S is satisfiable if it is true in some world**
  - **Example:**
- **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 true**

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:
  1. **Rules to derive new formulas from old (inference)**
  2. **Show (KB ∧ ¬ S) is unsatisfiable**

Inference

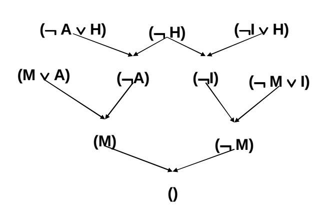
- **Mechanical process for computing new sentences**
- **Resolution**
  { (p ∨ α), (¬ p ∨ β) } $_R$ (α ∨ β)
  - **Correctness**
    If S1 $_R$ S2 then S1 S2
  - **Refutation Completeness:**
    If S is unsatisfiable then S $_R$ ()

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



New Variable Trick

Putting a formula in clausal form may increase its size exponentially

But can avoid this by introducing dummy variables

$(a \wedge b \wedge c) \vee (d \wedge e \wedge f) \Rightarrow \{(a \vee d),(a \vee e),(a \vee f),$
$(b \vee d),(b \vee e),(b \vee f),$
$(c \vee d),(c \vee e),(c \vee f)\ \}$

$(a \wedge b \wedge c) \vee (d \wedge e \wedge f) \Rightarrow \{(\mathbf{g} \vee \mathbf{h}),$
$(\neg a \vee \neg b \vee \neg c \vee \mathbf{g}),(\neg \mathbf{g} \vee a),(\neg \mathbf{g} \vee b),(\neg \mathbf{g} \vee c),$
$(\neg d \vee \neg e \vee \neg f \vee \mathbf{h}),(\neg \mathbf{h} \vee d),(\neg \mathbf{h} \vee e),(\neg \mathbf{h} \vee f)\}$

Dummy variables don't change satisfiability!

DPLL
Davis Putnam Loveland Logmann

- **Model finding: Backtrack search over space of partial truth assignments**

DPLL(wff):

Simplify wff: for each unit clause (Y)

Remove clauses containing Y

if no clause left then return true (satisfiable)

Shorten clauses contain ¬ Y

if empty clause then return false

Choose a variable

Choose a value (0/1) – yields literal X

if DPLL(wff, X) return true (satisfiable)

else return DPLL(wff, ¬ X)

DPLL
Davis Putnam Loveland Logemann

- **Backtrack search over space of partial truth assignments**

DPLL(wff):

Simplify wff: for each unit clause (Y)

Remove clauses containing Y

if no clause left then return true (satisfiable)

Shorten clauses contain ¬ Y

if empty clause then return false

Choose a variable

Choose a value (0/1) – yields literal X

if DPLL(wff, X) return true (satisfiable)

else return DPLL(wff, ¬ X)

unit propagation

= arc consistency

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!**

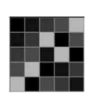
DPLL

- **Developed 1962 – still the best complete algorithm for propositional reasoning**
- **State of the art solvers use:**
  - **Smart variable choice heuristics**
  - **"Clause learning" – at backtrack points, determine minimum set of choices that caused inconsistency, add new clause**
    - **Limited resolution (Agarwal, Kautz, Beame 2002)**
  - **Randomized tie breaking & restarts**
- **Chaff – fastest complete SAT solver**
  - **Created by 2 Princeton undergrads, for a summer project!**
  - **Superscaler processor verification**
  - **AI planning - Blackbox**

Exercise

- **How could we represent the Quasigroup Completion Problem as a Boolean formula in CNF form?**

(take 10 minutes to sketch solution)



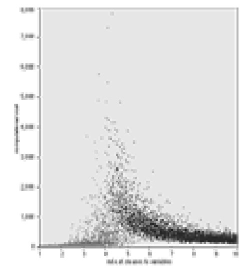
WalkSat

Local search over space of complete truth assignments
- **With probability P: flip any variable in any unsat clause**
- **With probability (1-P): flip best variable in any unsat clause**
  - **Like fixed-temperature simulated annealing**

- **SAT encodings of QCP, N-Queens, scheduling**
- **Best algorithm for random K-SAT**
  - **Best DPLL: 700 variables**
  - **Walksat: 100,000 variables**

CPGomes - AAAI00

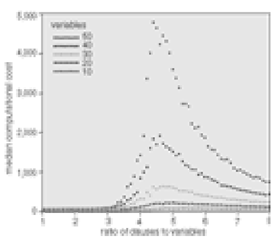
Random 3-SAT



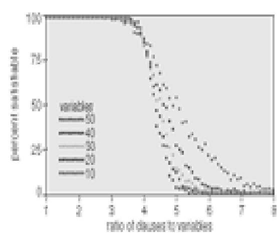
- Random 3-SAT
  - sample uniformly from space of all possible 3-clauses
  - *n* variables, *l* clauses
- Which are the hard instances?
  - around *l/n* = 4.3

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?*



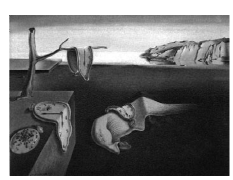
Random 3-SAT



- 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

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)



Salvidor Dali, *Persistance of Memory*

Logical Reasoning about Time & Change

AKA

Planning as Satisfiability

Actions

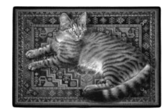
We want to relate changes in the world over time to actions associated with those changes

How are actions represented?

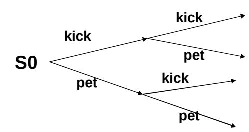
1. **As functions from one state to another**
2. **As predicates that are true in the state in which they (begin to) occur**

Actions as Functions: "Situation Calculus"

On(cat, mat, S0)
Happy(cat, S0)
¬ On(cat, mat, kick(S0))
¬ Happy(cat, kick(S0))
Happy(cat, pet(kick(S0)))



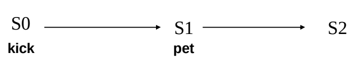
Branching time:



Actions as Predicates: "Action Calculus"

On(cat, mat, S0) ∧ Happy(S0)
Kick(cat, S0)
¬ On(cat, mat, S1) ∧ ¬Happy(CAT99, S1)
Pet(CAT99, S1)
¬ On(CAT99, MAT37, S2) ∧ Happy(CAT99, S2)

Linear time:



Relating Actions to Preconditions & Effects

Strips notation:

- **Action:** Fly(plane, start, dest)
- **Precondition**: Airplane(plane), City(start), City(dest), At(plane, start)
- **Effect**: At(plane, dest), ¬ At(plane, start)

Pure strips: no negative preconditions!

Need to represent logically:

- **An action requires it's predications**
- **An action causes it's effects**
- **Interfering actions do not co-occur**
- **Changes in the world are the result of actions.**

Preconditions & Effects

∀ plane, start, dest, s . Fly(plane, start, dest, s) ⊃
[At(plane, start, s) ∧
Airplane(plane,s) ∧ City(start) ∧ City(dest)]

- **Note: state indexes on predicates that never change not necessary.**

∀ plane, start, dest, s . Fly(plane, start, dest, s) ⊃
At(plane, dest, s+1)

- **In action calculus, the logical representation of "requires" and "causes" is the same!**
- **Not a full blown theory of causation, but good enough…**

Interfering Actions

Want to rule out:

Fly(PLANE32, NYC, DETROIT, S4) ∧
Fly(PLANE32, NYC, DETROIT, S4)

Actions interfere if one changes a precondition or effect of the other

They are mutually exclusive – "mutex"

∀ p, c1, c2, c3, c4, s .
[Fly(p, c1, c2, s) ∧ (c1 ≠ c3 ∨ c2 ≠ c4)] ⊃
¬ Fly(p, c3, c4, s)

(Similar for any other actions Fly is mutex with)

Explanatory Axioms

- **Don't want world to change "by magic" – only actions change things**
  - **If a proposition changes from true to false (or vice-versa), then some action that can change it must have occurred**

$\forall$ plane, start, s . [Airplane(plane) $\wedge$ City(start)
 At(plane,start,s) $\wedge$ $\neg$At(plane,city,s+1)] $\supset$
 $\exists$ dest . [City(dest) $\wedge$ Fly(plane, start, dest, s)]

$\forall$ plane, dest, s . [Airplane(plane) $\wedge$ City(start)
 $\neg$At(plane,dest,s) $\wedge$ At(plane,dest,s+1)] $\supset$
 $\exists$ start . [City(start) $\wedge$ Fly(plane, start, dest, s)]

The Frame Problem

General form of explanatory axioms:
[p(s) $\wedge$ $\neg$p(s+1)] $\supset$ [A1(s) $\vee$ A2(s) $\vee$ … $\vee$ An(s)]

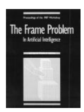
As a logical consequence, if none of these actions occurs, the proposition does not change

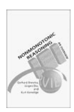
[$\neg$A1(s) $\wedge$ $\neg$A2(s) $\wedge$ … $\wedge$ $\neg$An(s)] $\supset$ [p(s) $\supset$ p(s+1)]

This solves the "frame problem" – being able to deduce what does not change when an action occurs

Frame Problem in AI

- **Frame problem identified by McCarthy in his first paper on the situation calculus (1969)**
  667 papers in researchindex !
- **Lead to a (misguided?) 20 year effort to develop non-standard logics where no frame axioms are required ("non-monotonic")**
  7039 papers!
- **1990 - Haas and Schubert independently pointed out that explanatory axioms are pretty easy to write down**





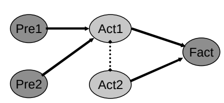
Planning as Satisfiability

- **Idea: in action calculus assert that initial state holds at time 0 and goal holds at some time (in the future):**
  - **Axioms $\wedge$ Initial $\wedge$ $\exists$ s . Goal(s)**
- **Any model that satisfies these assertions and the axioms for actions corresponds to a plan**
- **Bounded model finding, *i.e.* satisfiability testing:**
  1. **Assert goal holds at a particular time K**
  2. **Ground out (instantiate) the theory up to time K**
  3. **Try to find a model; if so, done!**
  4. **Otherwise, increment K and repeat**

Reachability Analysis

- **Problem: many irrelevant propositions, large formulas**
- **Reachability analysis: what propositions actually connect to initial state or goal in K steps?**
- **Graphplan's plan graph computes reachable set!**
- **Blackbox (Kautz & Selman 1999)**
  - **Run graphplan to generate plan graph**
  - **Translate plan graph to CNF formula**
  - **Run any SAT solver**

Translation of Plan Graph



Fact $\supset$ Act1 $\vee$ Act2
Act1 $\supset$ Pre1 $\wedge$ Pre2
$\neg$Act1 $\vee$ $\neg$Act2

Improved Encodings

Translations of Logistics.a:

STRIPS → Axiom Schemas → SAT

- **3,510 variables, 16,168 clauses**
- **24 hours to solve**

STRIPS → Plan Graph → SAT

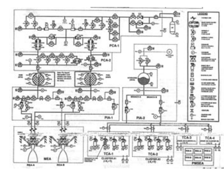
(Blackbox)

- **2,709 variables, 27,522 clauses**
- **5 seconds to solve!**

Model-Based Diagnosis

Idea:

- **Create a logical model of the correct functioning of a device**
- **When device is broken, observations + model is inconsistent**
- **Create diagnosis by restoring consistency**



Simplified KB

Knowledge Base:
SignalValueA $\land$ ValveAok $\supset$ ValveAopen
SignalValueB $\land$ ValveBok $\supset$ ValveBopen
SignalValueC $\land$ ValveCok $\supset$ ValveCopen
ValveAopen$\land$ $\supset$ EngineHasFuel
ValveBopen$\land$ $\supset$ EngineHasFuel
ValveCopen $\supset$ EngineHasOxy
EngineHasFuel $\land$ EngineHasOxy $\supset$ EngineFires
Normal Assumptions:
ValveAok, ValveBok, ValveCok
Direct Actions (cannot fail):
SignalValveA, SignalValveB, SignalValveC
Observed:
$\neg$ EngineFires

Diagnosis: 1

Knowledge Base:
SignalValueA $\land$ ValveAok $\supset$ ValveAopen
SignalValueB $\land$ ValveBok $\supset$ ValveBopen
SignalValueC $\land$ ValveCok $\supset$ ValveCopen
ValveAopen$\land$ $\supset$ EngineHasFuel
ValveBopen$\land$ $\supset$ EngineHasFuel
ValveCopen $\supset$ EngineHasOxy
EngineHasFuel $\land$ EngineHasOxy $\supset$ EngineFires
Normal Assumptions:
ValveAok, ValveBok, ValveCok
Direct Actions (cannot fail):
SignalValveA, SignalValveB, SignalValveC
Observed:
$\neg$ EngineFires

Inconsistent by Unit Propagation

Diagnosis: 2

Knowledge Base:
SignalValueA $\land$ ValveAok $\supset$ ValveAopen
SignalValueB $\land$ ValveBok $\supset$ ValveBopen
SignalValueC $\land$ ValveCok $\supset$ ValveCopen
ValveAopen$\land$ $\supset$ EngineHasFuel
ValveBopen$\land$ $\supset$ EngineHasFuel
ValveCopen $\supset$ EngineHasOxy
EngineHasFuel $\land$ EngineHasOxy $\supset$ EngineFires
Normal Assumptions:
ValveAok, ValveBok, ValveCok
Direct Actions (cannot fail):
SignalValveA, SignalValveB, SignalValveC
Observed:
$\neg$ EngineFires

Still Inconsistent

Diagnosis: 3

Knowledge Base:
SignalValueA $\land$ ValveAok $\supset$ ValveAopen
SignalValueB $\land$ ValveBok $\supset$ ValveBopen
SignalValueC $\land$ ValveCok $\supset$ ValveCopen
ValveAopen$\land$ $\supset$ EngineHasFuel
ValveBopen$\land$ $\supset$ EngineHasFuel
ValveCopen $\supset$ EngineHasOxy
EngineHasFuel $\land$ EngineHasOxy $\supset$ EngineFires
Normal Assumptions:
ValveAok, ValveBok, ValveCok
Direct Actions (cannot fail):
SignalValveA, SignalValveB, SignalValveC
Observed:
$\neg$ EngineFires

Consistency Restored!

Diagnosis: Valve A and Valve B broken (double fault)

Diagnosis: 4

Knowledge Base:

SignalValueA $\land$ ValveAok $\supset$ ValveAopen
SignalValueB $\land$ ValveBok $\supset$ ValveBopen
SignalValueC $\land$ ValveCok $\supset$ ValveCopen
ValveAopen$\land$ $\supset$ EngineHasFuel
ValveBopen$\land$ $\supset$ EngineHasFuel
ValveCopen $\supset$ EngineHasOxy
EngineHasFuel $\land$ EngineHasOxy $\supset$ EngineFires

Normal Assumptions:

ValveAok, ValveBok, ValveCok

Direct Actions (cannot fail):

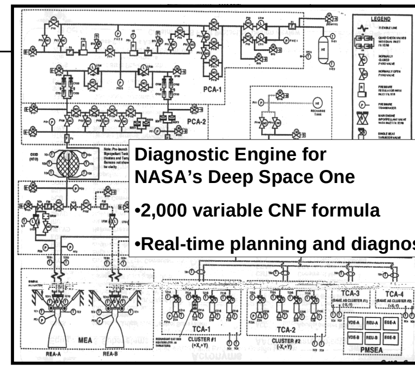
SignalValveA, SignalValveB, SignalValveC

Observed:

$\neg$ EngineFires

A different way to restore consistency

Diagnosis: Valve C broken (single fault)



Diagnostic Engine for NASA's Deep Space One

•2,000 variable CNF formula

•Real-time planning and diagnosis

Beyond Logic

- **Often you want most likely diagnosis rather than all possible diagnoses**
- **Can assign probabilities to sets of fault, and search for most likely way to restore consistency**
- **But suppose observations and model of the device are also uncertain?**

- **Next: Probabilistic Reasoning in Bayesian Networks**