Artificial Intelligence

Planning as Satisfiability

Russell and Norvig: Chapters 5, 11

we follow the following paper

Pushing the Envelope:
Planning, Propositional Logic, and Stochastic Search

Kautz and Selman

http://www.cs.cornell.edu/home/selman/papers-ftp/plan.ps

AIPS-98 Planning Competition

Round	Planner	Av. Time	Solved	Shortest
Round 1	**BLACKBOX**	**1.49**	**63**	**55**
	HSP	35.48	82	61
	IPP	7.40	63	49
	STAN	55.41	64	47
Round 2	**BLACKBOX**	**2.46**	**8**	**6**
	HSP	25.87	9	5
	IPP	17.37	11	8
	STAN	1.33	7	4

first-order logic (= situation calculus)

initial state

At(Home, s0)
AND NOT Have(Milk, s0)
AND NOT Have(Bananas, s0)
AND NOT Have(Drill, s0)

goal state (= query)

EXISTS s
At(Home, s)
AND Have(Milk, s)
AND Have(Bananas, s)
AND Have(Drill, s)

operators

FORALL a, s
Have(Milk, Result(a,s))
EQUIV
a = Buy(Milk) AND At(Supermarket, s)
OR
Have(Milk, s) AND NOT a = Drop(Milk)

problems with first-order logic

- inefficient
- does not necessarily generate a GOOD plan

propositional logic - initial and final situation

initial state	goal state
At(Home,s0)	At(Home,s9)
NOT At(SM,s0)	
NOT At(HWS,s0)	
NOT Have(Drill,s0)	Have(Drill,s9)
NOT Have(Milk,s0)	Have(Milk,s9)
NOT Have(Bananas,s0)	Have(Bananas,s9)

notice:
these are not really predicates
they are variables
(for example, At(home,s0) could be replaced with x)

we use knowledge to eliminate
variables such as Sells(SM, Milk)
from the encoding

propositional logic - operators (1)

different encodings are possible

here: - graphplan-based encodings
- linear encodings
- state-based encodings

propositional logic - operators (2)

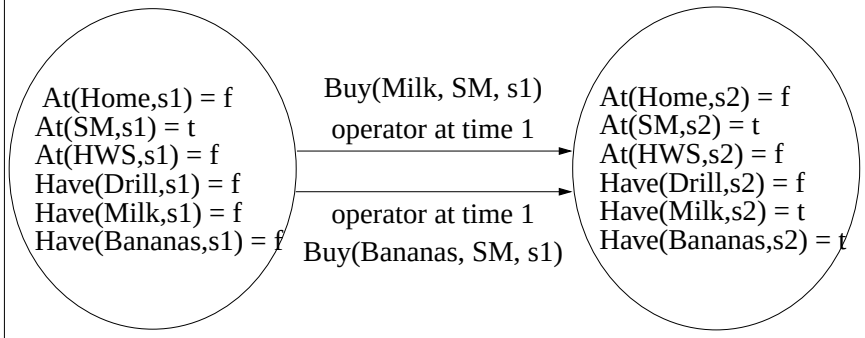
operators are just variables

Go(Home, SM, s0)
Go(Home, HWS, s0)
Go(SM, Home, s0)
Go(SM, HWS, s0)
Go(HWS, SM, s0)
...
Buy(Drill, HWS, s0)
Buy(Milk, SM, s0)
Buy(Bananas, SM, s0)
Buy(Drill, HWS, s1)
...

initial state	time 1	time 2
At(Home,s0) = t	At(Home,s1) = f	At(Home,s2) = f
At(SM,s0) = f	At(SM,s1) = t	At(SM,s2) = t
At(HWS,s0) = f	At(HWS,s1) = f	At(HWS,s2) = f
Have(Drill,s0) = f	Have(Drill,s1) = f	Have(Drill,s2) = f
Have(Milk,s0) = f	Have(Milk,s1) = f	Have(Milk,s2) = t
Have(Bananas,s0) = f	Have(Bananas,s1) = f	Have(Bananas,s2) = t
Go(Home, SM, s0) = t	Go(Home, SM, s1) = f	Go(Home, SM, s2)
Go(Home, HWS, s0) = f	Go(Home, HWS, s1) = f	Go(Home, HWS, s2)
Go(SM, Home, s0) = f	Go(SM, Home, s1) = f	Go(SM, Home, s2)
Go(SM, HWS, s0) = f	Go(SM, HWS, s1) = f	Go(SM, HWS, s2)
Go(HWS, SM, s0) = f	Go(HWS, SM, s1) = f	Go(HWS, SM, s2)
...	...	...
Buy(Drill, HWS, s0) = f	Buy(Drill, HWS, s1) = f	Buy(Drill, HWS, s2)
Buy(Milk, SM, s0) = f	Buy(Milk, SM, s1) = t	Buy(Milk, SM, s2)
Buy(Bananas, SM, s0) = f	Buy(Bananas, SM, s1) = t	Buy(Bananas, SM, s2)
...	...	...

propositional logic - operators (4)

partial-order plan



propositional logic - operators (5)

For every fact (and its negation, respectively), we add a “maintain” operator that simply has that fact (or its negation, respectively) as a precondition

MaintainAt(x)
Precond: At(x)
Effect:

...

propositional logic - constraints (1)

Go(here, there)
Precond: At(here)
Effect: At(there) AND NOT At(here)

Buy(x, store)
Precond: At(store) and Sells(store,x)
Effect: Have(x)

MaintainAt(x)
Precond: At(x)
Effect:

each fact (and its negation, respectively) at time t implies the disjunction of all the operators at time t-1 that have it as an add-effect (or delete-effect)

Have(Milk, s1) IMPLIES Buy(Milk, SM, s0) OR MaintainHave(Milk, s0)
NOT Have(Milk, s1) IMPLIES MaintainNotHave(Milk,s0)
At(SM, s2) IMPLIES Go(Home, SM, s1) OR Go(HWS, SM, s1) OR ...

propositional logic - constraints (2)

Go(here, there)
Precond: At(here)
Effect: At(there) AND NOT At(here)

Buy(x, store)
Precond: At(store) and Sells(store,x)
Effect: Have(x)

MaintainAt(x)
Precond: At(x)
Effect:

operators imply their preconditions

Go(HWS, SM, s1) IMPLIES At(HWS, s1)
Buy(Bananas, SM, s2) IMPLIES At(SM, s2)
...

propositional logic - constraints (3)

Go(here, there)
Precond: At(here)
Effect: At(there) AND NOT At(here)

Buy(x, store)
Precond: At(store) and Sells(store,x)
Effect: Have(x)

MaintainAt(x)
Precond: At(x)
Effect:

conflicting actions cannot be executed at the same time

NOT Go(HWS, SM, s1) OR NOT Go(HWS, Home, s1)
NOT Buy(HWS, Drill, s1) OR NOT Go(HWS, SM, s1)
...

Reasoning

knowledge-base

At(Home,s0)
NOT At(SM,s0)
NOT At(HWS,s0)
NOT Have(Drill,s0)
NOT Have(Milk,s0)
NOT Have(Bananas,s0)

At(Home,s9)
Have(Drill,s9)
Have(Milk,s9)
Have(Bananas,s9)

Have(Milk, s1) IMPLIES Buy(Milk, SM, s0) OR MaintainHave(Milk, s0)
NOT Have(Milk, s1) IMPLIES MaintainNotHave(Milk,s0)
At(SM, s2) IMPLIES Go(Home, SM, s1) OR Go(HWS, SM, s1) OR ...
...

Go(HWS, SM, s1) IMPLIES At(HWS, s1)
Buy(Bananas, SM, s2) IMPLIES At(SM, s2)
...

NOT Go(HWS, SM, s1) OR NOT Go(HWS, Home, s1)
NOT Buy(HWS, Drill, s1) OR NOT Go(HWS, SM, s1)
...

planning = solving SAT problem

we now have a huge propositional sentence
we need to find an interpretation that makes it true

satisfiability problem (= SAT)

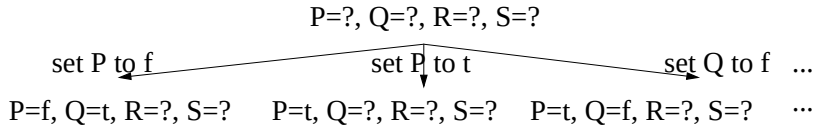
in general: NP hard

smaller sentences are easier to solve
this is why we used knowledge to eliminate some predicates

constraint satisfaction

SAT problems are search problems of a special kind. Why?

(P OR Q) AND (R OR NOT S)

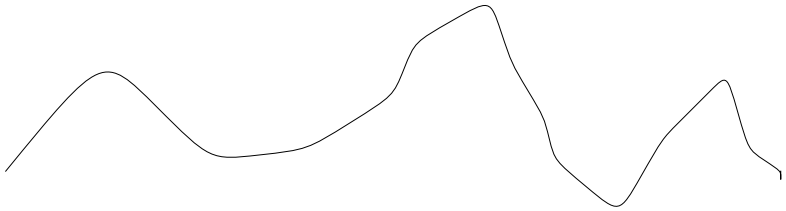


here: - use systematic search methods
- use heuristic methods

GSAT (1)

Russell and Norvig, Exercise 6.15

uses random restarts and hillclimbing

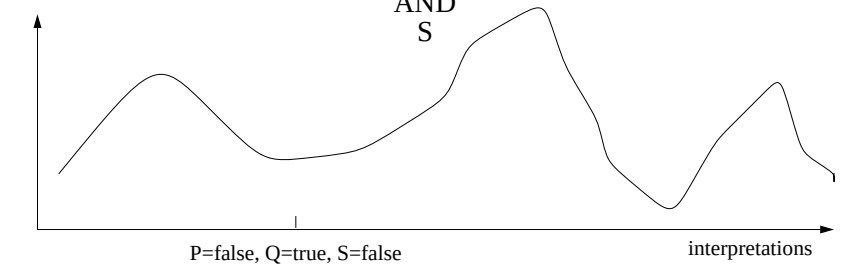


GSAT (2)

given a propositional sentence

represent it in CNF (= conjunctive normal form)

(P OR Q OR NOT S) ← clause
AND
(NOT P OR Q)
AND
S



successors (neighbors) of an interpretation are all the interpretations that differ from the interpretation by the assignment to one symbol

GSAT (3)

function GSAT (sentence, max-restarts, max-climbs)
returns a truth assignment or failure

```
for i := 1 to max-restarts
    A := a randomly generated truth assignment
    for j := 1 to max-climbs do
        if A satisfies sentence then return A
        A := a random choice of one of the best successors of A
return failure
```

completeness?
soundness?

GSAT has been improved to WalkSAT

summary

planning problem in STRIPS notation

↓

propositional sentence

↓ WalkSAT (or other search methods)

interpretation that satisfies the sentence

↓

plan (delete “Noops”)

one of the fastest known planners