

Automatic Planning

Jörg Hoffmann

University of Innsbruck
STI

Masterstudium Informatik Vertiefungsmodul (WM 9)

- ▶ **Planning as Satisfiability**
- ▶ The “Linear” Encoding
- ▶ The “Graphplan-Based” Encoding
- ▶ “State-Based” Encodings
- ▶ Behind the Curtains



Jörg Hoffmann

Automatic Planning

Planning as Satisfiability

Transform Planning into a series of SAT Tests:

```
b := 0
while TRUE do
   $\Phi(b)$  := a CNF that is satisfiable iff
    there exists a plan with  $b$  steps
  if DPLL( $\Phi(b)$ ,  $\emptyset$ ) then
    output Plan encoded by satisfying assignment  $a$ 
  endif
   $b := b + 1$ 
endwhile
```

- ▶ *What do we know about the found plan?*
- ▶ *What can you say about the “completeness” of the algorithm?*
- ▶ *SAT is in NP but Bounded-PLANSAT is PSPACE-hard???*



Jörg Hoffmann

Automatic Planning

Jörg Hoffmann

Automatic Planning



How to encode b -step STRIPS plan existence as a CNF?

Many possible answers. All (I know) share:

- ▶ Time steps $0 \leq t \leq b$
- ▶ Fact variables p_t : is p TRUE or FALSE, at this time step?
- ▶ Action variables a_t : will a be applied or not, at this time step?

The size of the encoding grows linear in b – exponential if b is exponentially large!

- ▶ Planning as Satisfiability
- ▶ The “Linear” Encoding
- ▶ The “Graphplan-Based” Encoding
- ▶ “State-Based” Encodings
- ▶ Behind the Curtains

The “Linear” Encoding, I

- ▶ Notations: facts P , actions A , initial state I , goals G , time steps $0 \leq t \leq b$
- ▶ Decision variables: p_t for all $0 \leq t \leq b$ and $p \in P$, a_t for all $0 \leq t \leq b - 1$ and $a \in A$
- ▶ Initial State Clauses: “specify initial values”
For all $p \in P : \{p_0\}$ if $p \in I$, $\{???\}$ if $p \notin I$
- ▶ Goal Clauses: “specify goal values”
For all $g \in G : \{???\}$

The “Linear” Encoding, II

- ▶ Action Precondition Clauses: “action implies its preconditions”
For all $0 \leq t \leq b - 1$, $a \in A$, $p \in pre(a) : \{\neg a_t, p_t\}$
- ▶ Action Effect Clauses (add): “action implies its add effects”
For all $0 \leq t \leq b - 1$, $a \in A$, $p \in add(a) : \{???\}$
- ▶ Action Effect Clauses (del): “action implies its delete effects”
For all $0 \leq t \leq b - 1$, $a \in A$, $p \in del(a) : \{???\}$

- ▶ Positive Frame Clauses: “if $p \notin del(a)$ was true and a is applied, then p is still true”

For all $0 \leq t \leq b - 1, a \in A, p \notin del(a) : \{\neg a_t, \neg p_t, p_{t+1}\}$

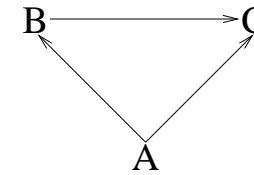
- ▶ Negative Frame Clauses: “if $p \notin add(a)$ was false and a is applied, then p is still false”

For all $0 \leq t \leq b - 1, a \in A, p \notin add(a) : \{???\}$

- ▶ Linearity Clauses: “apply exactly one action in each time step”

For all $0 \leq t \leq b - 1 : \{a_t \mid a \in A\}$

For all $0 \leq t \leq b - 1, a \in A, a \neq a' \in A : \{???\}$



$P = \{A, B, C, visB, visC\}, I = \{A\}, G = \{visB, visC\},$
 $drAB = (\{A\}, \{B, visB\}, \{A\}), drAC = (\{A\}, \{C, visC\}, \{A\}),$
 $drBC = (\{B\}, \{C, visC\}, \{B\})$

Bound $b = 1$ linear encoding: Blackboard.

Why do we need the linearity clauses?

- ▶ Planning as Satisfiability
- ▶ The “Linear” Encoding
- ▶ The “Graphplan-Based” Encoding
- ▶ “State-Based” Encodings
- ▶ Behind the Curtains

- ▶ Notations: facts P , actions A , with NOOPs A^N , initial state I , goals G , time steps $0 \leq t \leq b$

- ▶ Decision variables: p_t for all $0 \leq t \leq b$ and $p \in P$, a_t for all $0 \leq t \leq b - 1$ and $a \in A^N$

- ▶ Initial State Clauses: “specify initial values”

For all $p \in P : \{p_0\}$ if $p \in I, \{???\}$ if $p \notin I$

- ▶ Goal Clauses: “specify goal values”

For all $g \in G : \{???\}$

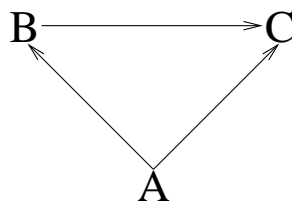
- ▶ Action Precondition Clauses: “action implies its preconditions”
For all $0 \leq t \leq b - 1, a \in A^N, p \in pre(a) : \{-a_t, p_t\}$
- ▶ Action Interference Clauses: “do not apply interfering actions in the same time step”
For all $0 \leq t \leq b - 1, a, a' \in A^N, a \neq a' : \{???\}$
[[$pre(a) \cup add(a) \cap del(a') \neq \emptyset$ or vice versa]
- ▶ Fact Achievement Clauses: “fact implies disjunction of its achievers”
For all $1 \leq t \leq b, p \in P : \{-p_t\} \cup \{???\}$

- ▶ In the Linear encoding, satisfying assignments are sequences of legal world states
- ▶ Not so for the parallel encoding: Blackboard
- ▶ Still sufficient for plan validity: goals and preconditions are satisfied, no conflicting actions
- ▶ *Optimal parallel plans are often shorter than optimal sequential plans!*
- ▶ So we need fewer iterations and consider smaller formulas!

Do we need anything more?



Example “SIMPLE”



$P = \{A, B, C, visB, visC\}, I = \{A\}, G = \{visB, visC\},$
 $drAB = (\{A\}, \{B, visB\}, \{A\}), drAC = (\{A\}, \{C, visC\}, \{A\}),$
 $drBC = (\{B\}, \{C, visC\}, \{B\})$

Bound $b = 1$ basic parallel encoding: Blackboard.

The Graphplan-Based Encoding, I

Use a 2-Planning Graph:

- ▶ Notations: facts P , actions A , with NOOPs A^N , initial state I , goals G , time steps $0 \leq t \leq b$, fact layers F_t , action layers A_t , fact mutexes EF_t , action mutexes EA_t
- ▶ Decision variables: p_t for all $1 \leq t \leq b$ and $p \in F_t, a_t$ for all $0 \leq t \leq b - 1$ and $a \in A_t$
- ▶ Goal Clauses: “specify goal values”
For all $g \in G : \{???\}$
- ▶ Action Precondition Clauses: “action implies its preconditions”
For all $1 \leq t \leq b - 1, a \in A_t, p \in pre(a) : \{???\}$



- ▶ Action Mutex Clauses: “do not apply mutex actions in the same time step”

For all $0 \leq t \leq b - 1, a, a' \in A_t, \{a, a'\} \in EA_t : \{???\}$

- ▶ Fact Achievement Clauses: “fact implies disjunction of its achievers”

For all $1 \leq t \leq b, p \in F_t : \{\neg p_t\} \cup \{???\}$

- ▶ Fact Mutex Clauses: “do not make mutex facts TRUE in the same time step”

For all $1 \leq t \leq b, p, p' \in F_t, \{p, p'\} \in EF_t : \{???\}$

- ▶ We don't need vars for the initial facts since $pre(a) \subseteq I$ anyway for all $a \in A_0$

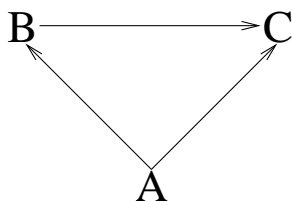
- ▶ Graphplan-Based Encoding == Basic Parallel Encoding pruned and enhanced by information contained in 2-Planning Graph

- ▶ Pruned: less decision vars p_t and a_t

- ▶ Enhanced: more mutex clauses $\{\neg a_t, \neg a'_t\}$ and $\{\neg p_t, \neg p'_t\}$



Example “SIMPLE”



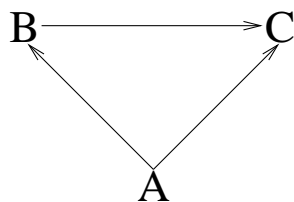
$P = \{A, B, C, visB, visC\}, I = \{A\}, G = \{visB, visC\},$
 $drAB = (\{A\}, \{B, visB\}, \{A\}), drAC = (\{A\}, \{C, visC\}, \{A\}),$
 $drBC = (\{B\}, \{C, visC\}, \{B\})$

Bound $b = 1$ graphplan-based encoding: Blackboard.

Basic Parallel vs. Graphplan-Based Encoding, II

- ▶ All the new clauses ϕ (the pruned $\{\neg a_t\}$ and $\{\neg p_t\}$ and all new exclusion clauses) follow from the Basic Parallel CNF Φ !
- ▶ We do some of the reasoning devoted to the SAT solver with a specialized algorithm instead
- ▶ Potentially exponential savings if (since) the SAT solver uses, in constraint propagation, 1-Resolution only: for exclusion relations we need 2-Resolution! [Brafman, IJCAI-1999, JAIR-2001]
- ▶ *What sort of resolution do we need to capture k-planning graphs in the constraint propagation procedure?*
- ▶ With the Graphplan-Based encoding, the “Planning as Satisfiability” approach became competitive! (It is today state-of-the-art for *optimal* planning)



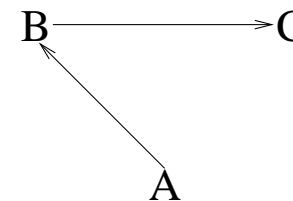


$P = \{A, B, C, visB, visC\}$, $I = \{A\}$, $G = \{visB, visC\}$,
 $drAB = (\{A\}, \{B, visB\}, \{A\})$, $drAC = (\{A\}, \{C, visC\}, \{A\})$,
 $drBC = (\{B\}, \{C, visC\}, \{B\})$

Bound $b = 1$ resolution for mutexes on basic parallel encoding:
Blackboard.

- ▶ Planning as Satisfiability
- ▶ The “Linear” Encoding
- ▶ The “Graphplan-Based” Encoding
- ▶ “State-Based” Encodings
- ▶ Behind the Curtains

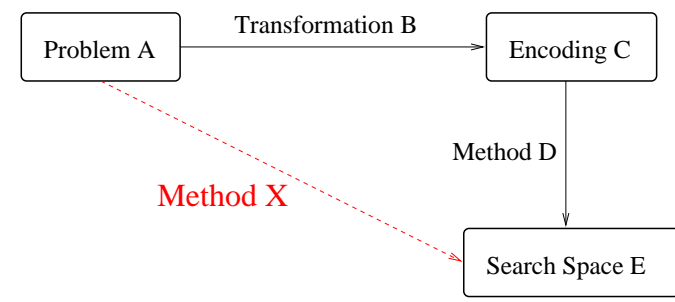
- ▶ Use only fact-at-time variables, extract plan from state changes
- ▶ First set of constraints describes state invariants
- ▶ Second set of constraints describes additional properties following from available transitions/actions
- ▶ No automatic CNF generation method (I know of) existent!
- ▶ Well, [Kautz&McAllester&Selman, KR-1996] propose an exponential method using resolution to remove action variables from a Graphplan-Based encoding ...
- ▶ Create by hand



$P = \{A, B, C\}$, $I = \{A\}$, $G = \{C\}$, $drAB = (\{A\}, \{B\}, \{A\})$,
 $drBC = (\{B\}, \{C\}, \{B\})$

Bound $b = 1$ state-based encoding: Blackboard.

- ▶ Planning as Satisfiability
- ▶ The “Linear” Encoding
- ▶ The “Graphplan-Based” Encoding
- ▶ “State-Based” Encodings
- ▶ Behind the Curtains



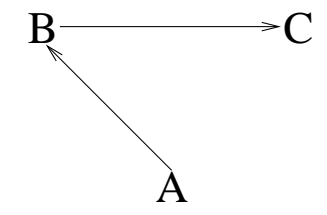
What are A, B, C, D, E in our case?

What is X?

A Very Simple Encoding

Use a 1-Planning Graph:

- ▶ Notations: actions A , with NOOPs A^N , goals G , time steps $0 \leq t \leq b$, action layers A_t
- ▶ Decision variables: a_t for all $0 \leq t \leq b - 1$ and $a \in A_t$
- ▶ Goal Clauses: “at least one achiever”
For all $g \in G : \{a_{b-1} \mid a \in A_{b-1}, g \in \text{addd}(a)\}$
- ▶ Action Precondition Clauses: “action implies disjunction of its precondition achievers”
For all $1 \leq t \leq b - 1, a \in A_t, p \in \text{pre}(a) :$
 $\{\neg a_t\} \cup \{a_{t-1} \mid a \in A_{t-1}, p \in \text{addd}(a)\}$
- ▶ Action Interference Clauses: as in basic parallel encoding



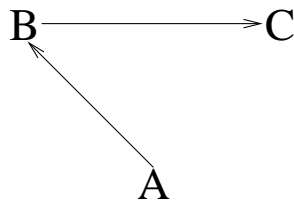
$P = \{A, B, C\}, I = \{A\}, G = \{C\}, drAB = (\{A\}, \{B\}, \{A\}),$
 $drBC = (\{B\}, \{C\}, \{B\})$

Bound $b = 2$ “very simple” encoding: Blackboard.

- ▶ Propagate $a_t = TRUE$:
 - Set a IN at t
 - if $t > 0$ then
 - for all $p \in pre(a)$ do
 - if all $a' \in A_{t-1}, p \in add(a')$ are OUT at $t-1$ then fail endif
 - if all $a' \in A_{t-1}, p \in add(a')$ are OUT at $t-1$, except a'' then propagate a'' IN at $t-1$ endif
 - endif
 - for all $a \neq a' \in A_t$ that interfere with a do propagate a' OUT at t
 - endif

- ▶ Propagate $a_t = FALSE$:
 - Set a OUT at t
 - if $t = b-1$ then
 - for all $g \in add(a) \cap G$ do
 - if all $a' \in A_t, g \in add(a')$ are OUT at t then fail endif
 - if all $a' \in A_t, g \in add(a')$ are OUT at t , except a'' then propagate a'' IN at t endif
 - endif
 - endif
 - if $t < b-1$ then
 - ???
 - endif
- ▶ See [Brafman, IJCAI-1999, JAIR-2001] for much more

Example “EVEN SIMPLER”



$P = \{A, B, C\}, I = \{A\}, G = \{C\}, drAB = (\{A\}, \{B\}, \{A\}),$
 $drBC = (\{B\}, \{C\}, \{B\})$

Bound $b = 2$ UP in “very simple” encoding: Blackboard.

Behind the Curtains, DPLL

- ▶ DPLL makes commitments of the form “I will/won’t apply action a at time t ”
 - ▶ The search state is a sequence of such commitments
- d0: “I will move the truck from x to y at time 17”
 d1: UP: “truck at x at time 17”, “truck at y at time 18”
 d1: “I will sell the truck at time 7”
 d2: UP: “no truck at times 8, ..., 25”
 d2: FALSE
 d1: “I will not sell the truck at time 7”
- ▶ The order of the commitments in the sequence is **completely independent of the individual times t** ...
 - ▶ ... which is why we also call this **undirected search**

- ▶ H. Kautz and B. Selman, *Planning as Satisfiability*, Proceedings ECAI 1992
- ▶ H. Kautz and B. Selman, *Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search*, Proceedings AAAI 1996
- ▶ H. Kautz, D. McAllester, and B. Selman, *Encoding Plans in Propositional Logic*, Proceedings KR 1996
- ▶ H. Kautz and B. Selman, *Unifying SAT-based and Graph-based Planning*, Proceedings IJCAI 1999

- ▶ J. Rintanen, *A Planning Algorithm not Based on Directional Search*, Proceedings KR 1998
- ▶ J. Hoffmann and H. Geffner, *Branching Matters: Alternative Branching in Graphplan*, Proceedings ICAPS 2003
- ▶ R. Brafman, *Reachability, Relevance, Resolution, and the Planning as Satisfiability Approach*, Proceedings IJCAI 1999, JAIR, vol. 14, 2001