Principles of AI Planning

November 24th, 2006 — Planning by satisfiability testing

Principles of AI Planning

Planning by satisfiability testing

Malte Helmert Bernhard Nebel

Albert-Ludwigs-Universität Freiburg

November 24th, 2006

Planning in the propositional logic

- Early work on deductive planning viewed plans as proofs that lead to a desired goal (theorem).
- Planning as satisfiability testing was proposed in 1992.
  1. A propositional formula represents all length n action sequences from the initial state to a goal state.
  2. If the formula is satisfiable then a plan of length n exists (and can be extracted from the satisfying valuation).
- Heuristic search and satisfiability planning are currently the best approaches for planning.
  - Satisfiability planning is often more efficient for small, but difficult problems.
  - Heuristic search is often more efficient for big, but easy problems.
- Bounded model-checking in Computer Aided Verification was introduced in 1998 as an extension of satisfiability planning after the success of the latter had been noticed outside the AI community.

Planning in the propositional logic

Abstractly

1. Represent actions (= binary relations) as propositional formulae.
2. Construct a formula saying "execute one of the actions".
3. Construct a formula saying "execute a sequence of n actions, starting from the initial state, ending in a goal state".
4. Test the satisfiability of this formula by a satisfiability algorithm.
5. If the formula is satisfiable, construct a plan from a satisfying valuation.

Satisfiability testing vs. state-space search

- Like our earlier algorithms (progression and regression planning, possibly with heuristics), planning as satisfiability testing can be interpreted as a search algorithm.
- However, unlike these algorithms, satisfiability testing is undirected search:
  - As the first decision, the algorithm may decide to include a certain action as the 7th operator of the plan.
  - As the second decision, it may require a certain state variable to be true after the 5th operator of the plan.
  - ...

Sets (of states) as formulae

Reminder: Formulae on A as sets of states

We view formulae ϕ as representing sets of states $s : A \to \{0, 1\}$.

Example

Formula $a \vee b$ on the state variables a, b, c represents the set $\{010, 011, 100, 101, 110, 111\}$.

Relations/actions as formulae

Formulae on $A \cup A'$ as binary relations

Let $A = \{a_1, \dots, a_n\}$ represent state variables in the current state, and $A' = \{a'_1, \dots, a'_n\}$ state variables in the successor state.
Formulae ϕ on $A \cup A'$ represent binary relations on states: a valuation of $A \cup A' \to \{0, 1\}$ represents a pair of states $s : A \to \{0, 1\}$, $s' : A' \to \{0, 1\}$.

Example

Formula $(a \to a') \wedge ((a' \vee b) \to b')$ on a, b, a', b' represents the binary relation $\{(00, 00), (00, 01), (00, 11), (01, 01), (01, 11), (10, 11), (11, 11)\}$.

Matrices as formulae

Example (Formulae as relations as matrices)

Binary relation $\{(00, 00), (00, 01), (00, 11), (01, 01), (01, 11), (10, 11), (11, 11)\}$ can be represented as the adjacency matrix:

	$a'b'$	$a'b'$	$a'b'$	$a'b'$
ab	00	01	10	11
00	1	1	0	1
01	0	1	0	1
10	0	0	0	1
11	0	0	0	1

Representation of big matrices is possible

For n state variables, a formula (over $2n$ variables) represents an adjacency matrix of size $2^n \times 2^n$.
For $n = 20$, matrix size is $2^{20} \times 2^{20} \sim 10^6 \times 10^6$.

Actions/relations as propositional formulae

Example

$\phi = (a_1 \leftrightarrow \neg a'_1) \wedge (a_2 \leftrightarrow \neg a'_2)$ as a matrix

$a_1 a_2$	$a'_1 a'_2$ 00	$a'_1 a'_2$ 01	$a'_1 a'_2$ 10	$a'_1 a'_2$ 11
00	0	0	0	1
01	0	0	1	0
10	0	1	0	0
11	1	0	0	0

and as a conventional truth table:

a_1	a_2	a'_1	a'_2	ϕ
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Actions/relations as propositional formulae

Example

$(a_1 \leftrightarrow a'_2) \wedge (a_2 \leftrightarrow a'_3) \wedge (a_3 \leftrightarrow a'_1)$ represents the matrix:

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	0	0	0	1	0	0	0
010	0	1	0	0	0	0	0	0
011	0	0	0	0	0	1	0	0
100	0	0	1	0	0	0	0	0
101	0	0	0	0	0	0	1	0
110	0	0	0	1	0	0	0	0
111	0	0	0	0	0	0	0	1

This action rotates the value of the state variables a_1, a_2, a_3 one step forward.

Translating operators into formulae

- Any operator can be translated into a propositional formula.
- Translation takes polynomial time.
- Resulting formula has polynomial size.
- Two main applications in planning algorithms are:
  1. planning as satisfiability and
  2. progression & regression for state sets as used in symbolic state-space traversal, typically implemented with the help of binary decision diagrams.

Translating operators into formulae

Definition (operators in propositional logic)

Let $o = \langle c, e \rangle$ be an operator and A a set of state variables. Define $\tau_A(o)$ as the conjunction of

$$c \quad (1)$$
$$\bigwedge_{a \in A}((EPC_a(e) \vee (a \wedge \neg EPC_{\neg a}(e))) \leftrightarrow a') \quad (2)$$
$$\bigwedge_{a \in A} \neg(EPC_a(e) \wedge EPC_{\neg a}(e)) \quad (3)$$

Condition (1) states that the precondition of o is satisfied.
Condition (2) states that the new value of a, represented by a', is 1 if the old value was 1 and it did not become 0, or if it became 1.
Condition (3) states that none of the state variables is assigned both 0 and 1. Together with (1), this encodes applicability of the operator.

Translating operators into formulae

Example

Example

Let the state variables be $A = \{a, b, c\}$.
Consider the operator $\langle a \vee b, (b \rhd a) \wedge (c \rhd \neg a) \wedge (a \rhd b)\rangle$.
The corresponding propositional formula is

$$\begin{aligned} & (a \vee b) \wedge ((b \vee (a \wedge \neg c)) \leftrightarrow a') \\ & \quad \wedge ((a \vee (b \wedge \neg\bot)) \leftrightarrow b') \\ & \quad \wedge ((\bot \vee (c \wedge \neg\bot)) \leftrightarrow c') \\ & \quad \wedge \neg(b \wedge c) \wedge \neg(a \wedge \bot) \wedge \neg(\bot \wedge \bot) \\ \equiv\ & (a \vee b) \wedge ((b \vee (a \wedge \neg c)) \leftrightarrow a') \\ & \quad \wedge ((a \vee b) \leftrightarrow b') \\ & \quad \wedge (c \leftrightarrow c') \\ & \quad \wedge \neg(b \wedge c) \end{aligned}$$

Translating operators into formulae

Example

Example

Let $A = \{a, b, c, d, e\}$ be the state variables.
Consider the operator $\langle a \wedge b, c \wedge (d \rhd e)\rangle$.
After simplifications, the formula $\tau_A(o)$ is

$$(a \wedge b) \wedge (a \leftrightarrow a') \wedge (b \leftrightarrow b') \wedge c' \wedge (d \leftrightarrow d') \wedge ((d \vee e) \leftrightarrow e')$$

Correctness

Lemma

Let s and s' be states and o an operator. Let $v : A \cup A' \to \{0,1\}$ be a valuation such that

1. *for all $a \in A$, $v(a) = s(a)$, and*
2. *for all $a \in A$, $v(a') = s'(a)$.*

Then $v \models \tau_A(o)$ if and only if $s' = app_o(s)$.

Planning as satisfiability

1. Encode operator sequences of length 0, 1, 2, ... as formulae Φ_0^{seq}, Φ_1^{seq}, Φ_2^{seq}, ... (see next slide).
2. Test satisfiability of Φ_0^{seq}, Φ_1^{seq}, Φ_2^{seq}, ...
3. If a satisfying valuation v is found, a plan can be constructed from v.

Planning as satisfiability

Definition (transition relation in propositional logic)

For $\langle A, I, O, G \rangle$ define $\mathcal{R}_1(A, A') = \bigvee_{o \in O} \tau_A(o)$.

Definition (bounded-length plans in propositional logic)

Existence of plans of length t is represented by the following formula over propositions $A^0 \cup \cdots \cup A^t$, where $A^i = \{ a^i \mid a \in A \}$ for all $i \in \{0, \ldots, t\}$:

$$\Phi_t^{seq} = \iota^0 \wedge \mathcal{R}_1(A^0, A^1) \wedge \mathcal{R}_1(A^1, A^2) \wedge \cdots \wedge \mathcal{R}_1(A^{t-1}, A^t) \wedge G^t$$

where $\iota^0 = \bigwedge_{a \in A, I(a)=1} a^0 \wedge \bigwedge_{a \in A, I(a)=0} \neg a^0$
and G^t is G with propositions a replaced by a^t.

Planning as satisfiability

Example

Example

Consider

$I \models b \wedge c$
$G = (b \wedge \neg c) \vee (\neg b \wedge c)$
$o_1 = \langle \top, (c \rhd \neg c) \wedge (\neg c \rhd c) \rangle$
$o_2 = \langle \top, (b \rhd \neg b) \wedge (\neg b \rhd b) \rangle$

The formula Φ_3^{seq} for plans of length 3 is:

$$\begin{aligned}
&(b^0 \wedge c^0) \\
\wedge &(((b^0 \leftrightarrow b^1) \wedge (c^0 \leftrightarrow \neg c^1)) \vee ((b^0 \leftrightarrow \neg b^1) \wedge (c^0 \leftrightarrow c^1))) \\
\wedge &(((b^1 \leftrightarrow b^2) \wedge (c^1 \leftrightarrow \neg c^2)) \vee ((b^1 \leftrightarrow \neg b^2) \wedge (c^1 \leftrightarrow c^2))) \\
\wedge &(((b^2 \leftrightarrow b^3) \wedge (c^2 \leftrightarrow \neg c^3)) \vee ((b^2 \leftrightarrow \neg b^3) \wedge (c^2 \leftrightarrow c^3))) \\
\wedge &((b^3 \wedge \neg c^3) \vee (\neg b^3 \wedge c^3)).
\end{aligned}$$

Planning as satisfiability

Existence of (optimal) plans

Theorem

*Let Φ_t^{seq} be the formula for $\langle A, I, O, G \rangle$ and plan length t.
The formula Φ_t^{seq} is satisfiable if and only if there is a sequence of states $s_0, \ldots, s_t$ and operators $o_1, \ldots, o_t$ such that $s_0 = I$, $s_i = app_{o_i}(s_{i-1})$ for all $i \in \{1, \ldots, t\}$, and $s_t \models G$.*

Consequence

If $\Phi_0^{seq}, \Phi_1^{seq}, \ldots, \Phi_{i-1}^{seq}$ are unsatisfiable and Φ_i^{seq} is satisfiable, then the length of shortest plans is i.
Satisfiability planning with Φ_i^{seq} yields optimal plans, like heuristic search with admissible heuristics and optimal algorithms like A* or IDA*.

Planning as satisfiability

Plan extraction

All satisfiability algorithms give a valuation v that satisfies Φ_i^{seq} upon finding out that Φ_i^{seq} is satisfiable.
This makes it possible to construct a plan.

Constructing a plan from a satisfying valuation

Let v be a valuation so that $v \models \Phi_t^{seq}$. Then define $s_i(a) = v(a^i)$ for all $a \in A$ and $i \in \{0, \ldots, t\}$.
The i-th operator in the plan is $o \in O$ if $app_o(s_{i-1}) = s_i$. Note: There may be more than one such operator, in which case any of them may be chosen.

Planning as satisfiability

Example, continued

Example

One valuation that satisfies Φ_3^{seq}:

	time i			
	0	1	2	3
b^i	1	1	0	0
c^i	1	0	0	1

Note:

1. There also exists a plan of length 1.
2. No plan of length 2 exists.

Conjunctive normal form

Many satisfiability algorithms require formulas in the conjunctive normal form: transformation by repeated applications of the following equivalences.

$$\begin{aligned}
\neg(\phi \vee \psi) &\equiv \neg\phi \wedge \neg\psi \\
\neg(\phi \wedge \psi) &\equiv \neg\phi \vee \neg\psi \\
\neg\neg\phi &\equiv \phi \\
\phi \vee (\psi_1 \wedge \psi_2) &\equiv (\phi \vee \psi_1) \wedge (\phi \vee \psi_2)
\end{aligned}$$

The formula is a conjunction of clauses (disjunctions of literals).

Example

$(A \vee \neg B \vee C) \wedge (\neg C \vee \neg B) \wedge A$

Note: Transformation to conjunctive normal form can increase formula size exponentially. There are also polynomial translations which introduce additional variables.

The unit resolution rule

Unit resolution

From $l_1 \vee l_2 \vee \cdots \vee l_n$ (here $n \geq 1$) and $\overline{l_1}$, infer $l_2 \vee \cdots \vee l_n$.

Example

From $a \vee b \vee c$ and $\neg a$ infer $b \vee c$.

Unit resolution: a special case

From A and $\neg A$ we get the empty clause $\perp$ ("disjunction consisting of zero disjuncts").

Unit subsumption

The clause $l_1 \vee l_2 \vee \cdots \vee l_n$ can be eliminated if we have the unit clause l_1.

The Davis-Putnam-Logemann-Loveland procedure

- The first efficient decision procedure for any logic (Davis, Putnam, Logemann & Loveland, 1960/62).
- Based on binary search through the valuations of a formula.
- Unit resolution and unit subsumption help pruning the search tree.
- The currently most efficient satisfiability algorithms are variants of the DPLL procedure.
  (Although there is currently a shift toward viewing these procedures as performing more general reasoning: clause learning.)

Satisfiability test by the DPLL procedure

Davis-Putnam-Logemann-Loveland Procedure

def DPLL(C: clauses):
- **while** there are clauses $(l_1 \vee \cdots \vee l_n) \in C$ and $\overline{l_1} \in C$:
  - $C := (C \setminus \{l_1 \vee \cdots \vee l_n\}) \cup \{l_2 \vee \cdots \vee l_n\}$
- **while** there are clauses $(l_1 \vee \cdots \vee l_n) \in C$ $(n \geq 2)$ and $l_1 \in C$:
  - $C := C \setminus \{l_1 \vee \cdots \vee l_n\}$
- **if** $\bot \in C$:
  - **return** false
- **if** C contains only unit clauses:
  - **return** true
- Pick some variable a such that $a \notin C$ and $\neg a \notin C$.
- **return** DPLL($C \cup \{a\}$) **or** DPLL($C \cup \{\neg a\}$)

Planning as satisfiability

Example: plan search with DPLL

Consider the problem from a previous slide, with two operators each inverting the value of one state variable, for plan length 3.

$$
\begin{aligned}
& (b^0 \wedge c^0) \\
\wedge & (((b^0 \leftrightarrow b^1) \wedge (c^0 \leftrightarrow \neg c^1)) \vee ((b^0 \leftrightarrow \neg b^1) \wedge (c^0 \leftrightarrow c^1))) \\
\wedge & (((b^1 \leftrightarrow b^2) \wedge (c^1 \leftrightarrow \neg c^2)) \vee ((b^1 \leftrightarrow \neg b^2) \wedge (c^1 \leftrightarrow c^2))) \\
\wedge & (((b^2 \leftrightarrow b^3) \wedge (c^2 \leftrightarrow \neg c^3)) \vee ((b^2 \leftrightarrow \neg b^3) \wedge (c^2 \leftrightarrow c^3))) \\
\wedge & ((b^3 \wedge \neg c^3) \vee (\neg b^3 \wedge c^3)).
\end{aligned}
$$

Planning as satisfiability

Example: plan search with DPLL

To obtain a short CNF formula, we introduce auxiliary variables o_1^i and o_2^i for $i \in \{1, 2, 3\}$ denoting operator applications.

b^0	$o_1^1 \rightarrow ((b^0 \leftrightarrow b^1) \wedge (c^0 \leftrightarrow \neg c^1))$
c^0	$o_2^1 \rightarrow ((b^0 \leftrightarrow \neg b^1) \wedge (c^0 \leftrightarrow c^1))$
$o_1^1 \vee o_2^1$	$o_1^2 \rightarrow ((b^1 \leftrightarrow b^2) \wedge (c^1 \leftrightarrow \neg c^2))$
$o_1^2 \vee o_2^2$	$o_2^2 \rightarrow ((b^1 \leftrightarrow \neg b^2) \wedge (c^1 \leftrightarrow c^2))$
$o_1^3 \vee o_2^3$	$o_1^3 \rightarrow ((b^2 \leftrightarrow b^3) \wedge (c^2 \leftrightarrow \neg c^3))$
$(b^3 \wedge \neg c^3) \vee (\neg b^3 \wedge c^3)$	$o_2^3 \rightarrow ((b^2 \leftrightarrow \neg b^3) \wedge (c^2 \leftrightarrow c^3))$

Planning as satisfiability

Example: plan search with DPLL

We rewrite the formulae for operator applications by using the equivalence $\phi \rightarrow (l \leftrightarrow l') \equiv ((\phi \wedge l \rightarrow l') \wedge (\phi \wedge \bar{l} \rightarrow \overline{l'}))$.

b^0	$o_1^1 \wedge b^0 \rightarrow b^1$	$o_1^2 \wedge b^1 \rightarrow b^2$	$o_1^3 \wedge b^2 \rightarrow b^3$
c^0	$o_1^1 \wedge \neg b^0 \rightarrow \neg b^1$	$o_1^2 \wedge \neg b^1 \rightarrow \neg b^2$	$o_1^3 \wedge \neg b^2 \rightarrow \neg b^3$
$o_1^1 \vee o_2^1$	$o_1^1 \wedge c^0 \rightarrow \neg c^1$	$o_1^2 \wedge c^1 \rightarrow \neg c^2$	$o_1^3 \wedge c^2 \rightarrow \neg c^3$
$o_1^2 \vee o_2^2$	$o_1^1 \wedge \neg c^0 \rightarrow c^1$	$o_1^2 \wedge \neg c^1 \rightarrow c^2$	$o_1^3 \wedge \neg c^2 \rightarrow c^3$
$o_1^3 \vee o_2^3$	$o_2^1 \wedge b^0 \rightarrow \neg b^1$	$o_2^2 \wedge b^1 \rightarrow \neg b^2$	$o_2^3 \wedge b^2 \rightarrow \neg b^3$
$b^3 \vee c^3$	$o_2^1 \wedge \neg b^0 \rightarrow b^1$	$o_2^2 \wedge \neg b^1 \rightarrow b^2$	$o_2^3 \wedge \neg b^2 \rightarrow b^3$
$\neg c^3 \vee \neg b^3$	$o_2^1 \wedge c^0 \rightarrow c^1$	$o_2^2 \wedge c^1 \rightarrow c^2$	$o_2^3 \wedge c^2 \rightarrow c^3$
	$o_2^1 \wedge \neg c^0 \rightarrow c^1$	$o_2^2 \wedge \neg c^1 \rightarrow c^2$	$o_2^3 \wedge \neg c^2 \rightarrow c^3$

Planning as satisfiability

Example: plan search with DPLL

Eliminate implications with $((l_1 \wedge l_2) \rightarrow l_3) \equiv (\overline{l_1} \vee \overline{l_2} \vee l_3)$.

b^0	$\neg o_1^1 \vee \neg b^0 \vee b^1$	$\neg o_1^2 \vee \neg b^1 \vee b^2$	$\neg o_1^3 \vee \neg b^2 \vee b^3$
c^0	$\neg o_1^1 \vee b^0 \vee \neg b^1$	$\neg o_1^2 \vee b^1 \vee \neg b^2$	$\neg o_1^3 \vee b^2 \vee \neg b^3$
$o_1^1 \vee o_2^1$	$\neg o_1^1 \vee \neg c^0 \vee \neg c^1$	$\neg o_1^2 \vee \neg c^1 \vee \neg c^2$	$\neg o_1^3 \vee \neg c^2 \vee \neg c^3$
$o_1^2 \vee o_2^2$	$\neg o_1^1 \vee c^0 \vee c^1$	$\neg o_1^2 \vee c^1 \vee c^2$	$\neg o_1^3 \vee c^2 \vee c^3$
$o_1^3 \vee o_2^3$	$\neg o_2^1 \vee \neg b^0 \vee \neg b^1$	$\neg o_2^2 \vee \neg b^1 \vee \neg b^2$	$\neg o_2^3 \vee \neg b^2 \vee \neg b^3$
$b^3 \vee c^3$	$\neg o_2^1 \vee b^0 \vee b^1$	$\neg o_2^2 \vee b^1 \vee b^2$	$\neg o_2^3 \vee b^2 \vee b^3$
$\neg c^3 \vee \neg b^3$	$\neg o_2^1 \vee \neg c^0 \vee c^1$	$\neg o_2^2 \vee \neg c^1 \vee c^2$	$\neg o_2^3 \vee \neg c^2 \vee c^3$
	$\neg o_2^1 \vee c^0 \vee \neg c^1$	$\neg o_2^2 \vee c^1 \vee \neg c^2$	$\neg o_2^3 \vee c^2 \vee \neg c^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i				
c^i				

	1	2	3
o_1^i			
o_2^i			

Planning as satisfiability

Example: plan search with DPLL

Identify unit clauses.

b^0	$\neg o_1^1 \vee \neg b^0 \vee b^1$	$\neg o_1^2 \vee \neg b^1 \vee b^2$	$\neg o_1^3 \vee \neg b^2 \vee b^3$
c^0	$\neg o_1^1 \vee b^0 \vee \neg b^1$	$\neg o_1^2 \vee b^1 \vee \neg b^2$	$\neg o_1^3 \vee b^2 \vee \neg b^3$
$o_1^1 \vee o_2^1$	$\neg o_1^1 \vee \neg c^0 \vee \neg c^1$	$\neg o_1^2 \vee \neg c^1 \vee \neg c^2$	$\neg o_1^3 \vee \neg c^2 \vee \neg c^3$
$o_1^2 \vee o_2^2$	$\neg o_1^1 \vee c^0 \vee c^1$	$\neg o_1^2 \vee c^1 \vee c^2$	$\neg o_1^3 \vee c^2 \vee c^3$
$o_1^3 \vee o_2^3$	$\neg o_2^1 \vee \neg b^0 \vee \neg b^1$	$\neg o_2^2 \vee \neg b^1 \vee \neg b^2$	$\neg o_2^3 \vee \neg b^2 \vee \neg b^3$
$b^3 \vee c^3$	$\neg o_2^1 \vee b^0 \vee b^1$	$\neg o_2^2 \vee b^1 \vee b^2$	$\neg o_2^3 \vee b^2 \vee b^3$
$\neg c^3 \vee \neg b^3$	$\neg o_2^1 \vee \neg c^0 \vee c^1$	$\neg o_2^2 \vee \neg c^1 \vee c^2$	$\neg o_2^3 \vee \neg c^2 \vee c^3$
	$\neg o_2^1 \vee c^0 \vee \neg c^1$	$\neg o_2^2 \vee c^1 \vee \neg c^2$	$\neg o_2^3 \vee c^2 \vee \neg c^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1			
c^i	1			

	1	2	3
o_1^i			
o_2^i			

Planning as satisfiability

Example: plan search with DPLL

Perform unit resolution with b^0 and c^0.

b^0	$\neg o_1^1 \vee \neg b^0 \vee b^1$	$\neg o_1^2 \vee \neg b^1 \vee b^2$	$\neg o_1^3 \vee \neg b^2 \vee b^3$
c^0	$\neg o_1^1 \vee b^0 \vee \neg b^1$	$\neg o_1^2 \vee b^1 \vee \neg b^2$	$\neg o_1^3 \vee b^2 \vee \neg b^3$
$o_1^1 \vee o_2^1$	$\neg o_1^1 \vee \neg c^0 \vee \neg c^1$	$\neg o_1^2 \vee \neg c^1 \vee \neg c^2$	$\neg o_1^3 \vee \neg c^2 \vee \neg c^3$
$o_1^2 \vee o_2^2$	$\neg o_1^1 \vee c^0 \vee c^1$	$\neg o_1^2 \vee c^1 \vee c^2$	$\neg o_1^3 \vee c^2 \vee c^3$
$o_1^3 \vee o_2^3$	$\neg o_2^1 \vee \neg b^0 \vee \neg b^1$	$\neg o_2^2 \vee \neg b^1 \vee \neg b^2$	$\neg o_2^3 \vee \neg b^2 \vee \neg b^3$
$b^3 \vee c^3$	$\neg o_2^1 \vee b^0 \vee b^1$	$\neg o_2^2 \vee b^1 \vee b^2$	$\neg o_2^3 \vee b^2 \vee b^3$
$\neg c^3 \vee \neg b^3$	$\neg o_2^1 \vee \neg c^0 \vee c^1$	$\neg o_2^2 \vee \neg c^1 \vee c^2$	$\neg o_2^3 \vee \neg c^2 \vee c^3$
	$\neg o_2^1 \vee c^0 \vee \neg c^1$	$\neg o_2^2 \vee c^1 \vee \neg c^2$	$\neg o_2^3 \vee c^2 \vee \neg c^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1			
c^i	1			

	1	2	3
o_1^i			
o_2^i			

Planning as satisfiability

Example: plan search with DPLL

Perform unit subsumption with b^0 and c^0.

b^0	$\neg o_1^1 \vee b^1$	$\neg o_1^2 \vee \neg b^1 \vee b^2$	$\neg o_1^3 \vee \neg b^2 \vee b^3$
c^0	$\neg o_1^1 \vee b^0 \vee \neg b^1$	$\neg o_1^2 \vee b^1 \vee \neg b^2$	$\neg o_1^3 \vee b^2 \vee \neg b^3$
$o_1^1 \vee o_2^1$	$\neg o_1^1 \vee \neg c^1$	$\neg o_1^2 \vee \neg c^1 \vee \neg c^2$	$\neg o_1^3 \vee \neg c^2 \vee \neg c^3$
$o_1^2 \vee o_2^2$	$\neg o_1^1 \vee c^0 \vee c^1$	$\neg o_1^2 \vee c^1 \vee c^2$	$\neg o_1^3 \vee c^2 \vee c^3$
$o_1^3 \vee o_2^3$	$\neg o_2^1 \vee \neg b^1$	$\neg o_2^2 \vee \neg b^1 \vee \neg b^2$	$\neg o_2^3 \vee \neg b^2 \vee \neg b^3$
$b^3 \vee c^3$	$\neg o_2^1 \vee b^0 \vee b^1$	$\neg o_2^2 \vee b^1 \vee b^2$	$\neg o_2^3 \vee b^2 \vee b^3$
$\neg c^3 \vee \neg b^3$	$\neg o_2^1 \vee c^1$	$\neg o_2^2 \vee \neg c^1 \vee c^2$	$\neg o_2^3 \vee \neg c^2 \vee c^3$
	$\neg o_2^1 \vee c^0 \vee \neg c^1$	$\neg o_2^2 \vee c^1 \vee \neg c^2$	$\neg o_2^3 \vee c^2 \vee \neg c^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1			
c^i	1			

	1	2	3
o_1^i			
o_2^i			

Planning as satisfiability

Example: plan search with DPLL

No unhandled unit clauses exist. Must branch.

$o_1^1 \vee o_2^1$
$o_1^2 \vee o_2^2$
$o_1^3 \vee o_2^3$
$b^3 \vee c^3$
$\neg c^3 \vee \neg b^3$

$\neg o_1^1 \vee b^1$
$\neg o_1^1 \vee \neg c^1$
$\neg o_2^1 \vee \neg b^1$
$\neg o_2^1 \vee c^1$

$\neg o_1^2 \vee \neg b^1 \vee b^2$
$\neg o_1^2 \vee b^1 \vee \neg b^2$
$\neg o_1^2 \vee \neg c^1 \vee \neg c^2$
$\neg o_1^2 \vee c^1 \vee c^2$
$\neg o_2^2 \vee \neg b^1 \vee \neg b^2$
$\neg o_2^2 \vee b^1 \vee b^2$
$\neg o_2^2 \vee \neg c^1 \vee c^2$
$\neg o_2^2 \vee c^1 \vee \neg c^2$

$\neg o_1^3 \vee \neg b^2 \vee b^3$
$\neg o_1^3 \vee b^2 \vee \neg b^3$
$\neg o_1^3 \vee \neg c^2 \vee \neg c^3$
$\neg o_1^3 \vee c^2 \vee c^3$
$\neg o_2^3 \vee \neg b^2 \vee \neg b^3$
$\neg o_2^3 \vee b^2 \vee b^3$
$\neg o_2^3 \vee \neg c^2 \vee c^3$
$\neg o_2^3 \vee c^2 \vee \neg c^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1			
c^i	1			

	1	2	3
o_1^i			
o_2^i			

Planning as satisfiability

Example: plan search with DPLL

We branch on b^1, first trying out $b^1 = 1$.

$o_1^1 \vee o_2^1$
$o_1^2 \vee o_2^2$
$o_1^3 \vee o_2^3$
$b^3 \vee c^3$
$\neg c^3 \vee \neg b^3$

$\neg o_1^1 \vee b^1$
$\neg o_1^1 \vee \neg c^1$
$\neg o_2^1 \vee \neg b^1$
$\neg o_2^1 \vee c^1$

$\neg o_1^2 \vee \neg b^1 \vee b^2$
$\neg o_1^2 \vee b^1 \vee \neg b^2$
$\neg o_1^2 \vee \neg c^1 \vee \neg c^2$
$\neg o_1^2 \vee c^1 \vee c^2$
$\neg o_2^2 \vee \neg b^1 \vee \neg b^2$
$\neg o_2^2 \vee b^1 \vee b^2$
$\neg o_2^2 \vee \neg c^1 \vee c^2$
$\neg o_2^2 \vee c^1 \vee \neg c^2$

$\neg o_1^3 \vee \neg b^2 \vee b^3$
$\neg o_1^3 \vee b^2 \vee \neg b^3$
$\neg o_1^3 \vee \neg c^2 \vee \neg c^3$
$\neg o_1^3 \vee c^2 \vee c^3$
$\neg o_2^3 \vee \neg b^2 \vee \neg b^3$
$\neg o_2^3 \vee b^2 \vee b^3$
$\neg o_2^3 \vee \neg c^2 \vee c^3$
$\neg o_2^3 \vee c^2 \vee \neg c^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1		
c^i	1			

	1	2	3
o_1^i			
o_2^i			

Planning as satisfiability

Example: plan search with DPLL

Perform unit resolution and unit subsumption with b^1.

$o_1^1 \vee o_2^1$
$o_1^2 \vee o_2^2$
$o_1^3 \vee o_2^3$
$b^3 \vee c^3$
$\neg c^3 \vee \neg b^3$

$\neg o_1^1 \vee b^1$
$\neg o_1^1 \vee \neg c^1$
$\neg o_2^1 \vee \neg b^1$
$\neg o_2^1 \vee c^1$

$\neg o_1^2 \vee \neg b^1 \vee b^2$
$\neg o_1^2 \vee b^1 \vee \neg b^2$
$\neg o_1^2 \vee \neg c^1 \vee \neg c^2$
$\neg o_1^2 \vee c^1 \vee c^2$
$\neg o_2^2 \vee \neg b^1 \vee \neg b^2$
$\neg o_2^2 \vee b^1 \vee b^2$
$\neg o_2^2 \vee \neg c^1 \vee c^2$
$\neg o_2^2 \vee c^1 \vee \neg c^2$

$\neg o_1^3 \vee \neg b^2 \vee b^3$
$\neg o_1^3 \vee b^2 \vee \neg b^3$
$\neg o_1^3 \vee \neg c^2 \vee \neg c^3$
$\neg o_1^3 \vee c^2 \vee c^3$
$\neg o_2^3 \vee \neg b^2 \vee \neg b^3$
$\neg o_2^3 \vee b^2 \vee b^3$
$\neg o_2^3 \vee \neg c^2 \vee c^3$
$\neg o_2^3 \vee c^2 \vee \neg c^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1		
c^i	1			

	1	2	3
o_1^i			
o_2^i			

Planning as satisfiability

Example: plan search with DPLL

Perform unit resolution and unit subsumption with $\neg o_2^1$.

$o_1^1 \vee o_2^1$
$o_1^2 \vee o_2^2$
$o_1^3 \vee o_2^3$
$b^3 \vee c^3$
$\neg c^3 \vee \neg b^3$

$\neg o_1^1 \vee \neg c^1$
$\neg o_2^1$
$\neg o_2^1 \vee c^1$

$\neg o_1^2 \vee b^2$
$\neg o_1^2 \vee \neg c^1 \vee \neg c^2$
$\neg o_1^2 \vee c^1 \vee c^2$
$\neg o_2^2 \vee \neg b^2$
$b^1 \vee b^2$
$\neg o_2^2 \vee \neg c^1 \vee c^2$
$\neg o_2^2 \vee c^1 \vee \neg c^2$

$\neg o_1^3 \vee \neg b^2 \vee b^3$
$\neg o_1^3 \vee b^2 \vee \neg b^3$
$\neg o_1^3 \vee \neg c^2 \vee \neg c^3$
$\neg o_1^3 \vee c^2 \vee c^3$
$\neg o_2^3 \vee \neg b^2 \vee \neg b^3$
$\neg o_2^3 \vee b^2 \vee b^3$
$\neg o_2^3 \vee \neg c^2 \vee c^3$
$\neg o_2^3 \vee c^2 \vee \neg c^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1		
c^i	1			

	1	2	3
o_1^i			
o_2^i	0		

Planning as satisfiability

Example: plan search with DPLL

Perform unit resolution and unit subsumption with o_1^1.

o_1^1
$o_1^2 \lor o_2^2$
$o_1^3 \lor o_2^3$
$b^3 \lor c^3$
$\neg c^3 \lor \neg b^3$

$\neg o_1^1 \lor \neg c^1$

$\neg o_1^2 \lor b^2$
$\neg o_1^2 \lor \neg c^1 \lor \neg c^2$
$\neg o_1^2 \lor c^1 \lor c^2$
$\neg o_2^2 \lor \neg b^2$
$b^1 \lor b^2$
$\neg o_2^2 \lor \neg c^1 \lor c^2$
$\neg o_2^2 \lor c^1 \lor \neg c^2$

$\neg o_1^3 \lor \neg b^2 \lor b^3$
$\neg o_1^3 \lor b^2 \lor \neg b^3$
$\neg o_1^3 \lor \neg c^2 \lor \neg c^3$
$\neg o_1^3 \lor c^2 \lor c^3$
$\neg o_2^3 \lor \neg b^2 \lor \neg b^3$
$\neg o_2^3 \lor b^2 \lor b^3$
$\neg o_2^3 \lor \neg c^2 \lor c^3$
$\neg o_2^3 \lor c^2 \lor \neg c^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1		
c^i	1			

	1	2	3
o_1^i	1		
o_2^i	0		

Planning as satisfiability

Example: plan search with DPLL

Perform unit resolution and unit subsumption with $\neg c^1$.

$o_1^2 \lor o_2^2$
$o_1^3 \lor o_2^3$
$b^3 \lor c^3$
$\neg c^3 \lor \neg b^3$

$\neg c^1$

$\neg o_1^2 \lor b^2$
$\neg o_1^2 \lor \neg c^1 \lor \neg c^2$
$\neg o_1^2 \lor c^1 \lor c^2$
$\neg o_2^2 \lor \neg b^2$
$b^1 \lor b^2$
$\neg o_2^2 \lor \neg c^1 \lor c^2$
$\neg o_2^2 \lor c^1 \lor \neg c^2$

$\neg o_1^3 \lor \neg b^2 \lor b^3$
$\neg o_1^3 \lor b^2 \lor \neg b^3$
$\neg o_1^3 \lor \neg c^2 \lor \neg c^3$
$\neg o_1^3 \lor c^2 \lor c^3$
$\neg o_2^3 \lor \neg b^2 \lor \neg b^3$
$\neg o_2^3 \lor b^2 \lor b^3$
$\neg o_2^3 \lor \neg c^2 \lor c^3$
$\neg o_2^3 \lor c^2 \lor \neg c^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1		
c^i	1	0		

	1	2	3
o_1^i	1		
o_2^i	0		

Planning as satisfiability

Example: plan search with DPLL

No unhandled unit clauses exist. Must branch a second time.

$o_1^2 \lor o_2^2$
$o_1^3 \lor o_2^3$
$b^3 \lor c^3$
$\neg c^3 \lor \neg b^3$

$\neg o_1^2 \lor b^2$
$\neg o_1^2 \lor c^2$
$\neg o_2^2 \lor \neg b^2$
$b^1 \lor b^2$
$\neg c^1 \lor c^2$
$\neg o_2^2 \lor \neg c^2$

$\neg o_1^3 \lor \neg b^2 \lor b^3$
$\neg o_1^3 \lor b^2 \lor \neg b^3$
$\neg o_1^3 \lor \neg c^2 \lor \neg c^3$
$\neg o_1^3 \lor c^2 \lor c^3$
$\neg o_2^3 \lor \neg b^2 \lor \neg b^3$
$\neg o_2^3 \lor b^2 \lor b^3$
$\neg o_2^3 \lor \neg c^2 \lor c^3$
$\neg o_2^3 \lor c^2 \lor \neg c^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1		
c^i	1	0		

	1	2	3
o_1^i	1		
o_2^i	0		

Planning as satisfiability

Example: plan search with DPLL

We branch on c^3, first trying out $c^3 = 1$.

$o_1^2 \lor o_2^2$
$o_1^3 \lor o_2^3$
$b^3 \lor c^3$
$\neg c^3 \lor \neg b^3$

$\neg o_1^2 \lor b^2$
$\neg o_1^2 \lor c^2$
$\neg o_2^2 \lor \neg b^2$
$b^1 \lor b^2$
$\neg c^1 \lor c^2$
$\neg o_2^2 \lor \neg c^2$

$\neg o_1^3 \lor \neg b^2 \lor b^3$
$\neg o_1^3 \lor b^2 \lor \neg b^3$
$\neg o_1^3 \lor \neg c^2 \lor \neg c^3$
$\neg o_1^3 \lor c^2 \lor c^3$
$\neg o_2^3 \lor \neg b^2 \lor \neg b^3$
$\neg o_2^3 \lor b^2 \lor b^3$
$\neg o_2^3 \lor \neg c^2 \lor c^3$
$\neg o_2^3 \lor c^2 \lor \neg c^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1		
c^i	1	0		1

	1	2	3
o_1^i	1		
o_2^i	0		

Planning as satisfiability

Example: plan search with DPLL

Perform unit resolution and unit subsumption with c^3.

$\neg o_1^2 \lor b^2$

$\neg o_1^3 \lor \neg b^2 \lor b^3$

$\neg o_1^3 \lor b^2 \lor \neg b^3$

$\neg o_1^3 \lor \neg c^2 \lor \neg c^3$

$o_1^2 \lor o_2^2$

$\neg o_1^2 \lor c^2$

$\neg o_1^3 \lor c^2 \lor c^3$

$o_1^3 \lor o_2^3$

$\neg o_2^2 \lor \neg b^2$

$\neg o_2^3 \lor \neg b^2 \lor \neg b^3$

$b^3 \lor c^3$

$b^1 \lor b^2$

$\neg o_2^3 \lor b^2 \lor b^3$

$\neg c^3 \lor \neg b^3$

$\neg c^1 \lor c^2$

$\neg o_2^3 \lor \neg c^2 \lor c^3$

$\neg o_2^2 \lor \neg c^2$

$\neg o_2^3 \lor c^2 \lor \neg c^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1		
c^i	1	0		1

	1	2	3
o_1^i	1		
o_2^i	0		

Planning as satisfiability

Example: plan search with DPLL

Perform unit resolution and unit subsumption with $\neg b^3$.

$\neg o_1^2 \lor b^2$

$\neg o_1^3 \lor \neg b^2 \lor b^3$

$\neg o_1^3 \lor b^2 \lor \neg b^3$

$\neg o_1^3 \lor \neg c^2$

$o_1^2 \lor o_2^2$

$\neg o_1^2 \lor c^2$

$o_1^3 \lor o_2^3$

$\neg o_2^2 \lor \neg b^2$

$\neg o_2^3 \lor \neg b^2 \lor \neg b^3$

$b^1 \lor b^2$

$\neg o_2^3 \lor b^2 \lor b^3$

$\neg b^3$

$\neg c^1 \lor c^2$

$\neg o_2^2 \lor \neg c^2$

$\neg o_2^3 \lor c^2$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1		0
c^i	1	0		1

	1	2	3
o_1^i	1		
o_2^i	0		

Planning as satisfiability

Example: plan search with DPLL

No unhandled unit clauses exist. Must branch a third time.

$\neg o_1^2 \lor b^2$

$\neg o_1^3 \lor \neg b^2$

$\neg o_1^3 \lor \neg c^2$

$o_1^2 \lor o_2^2$

$\neg o_1^2 \lor c^2$

$o_1^3 \lor o_2^3$

$\neg o_2^2 \lor \neg b^2$

$b^1 \lor b^2$

$\neg o_2^3 \lor b^2$

$\neg c^1 \lor c^2$

$\neg o_2^2 \lor \neg c^2$

$\neg o_2^3 \lor c^2$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1		0
c^i	1	0		1

	1	2	3
o_1^i	1		
o_2^i	0		

Planning as satisfiability

Example: plan search with DPLL

We branch on o_2^2, first trying out $o_2^2 = 1$.

$\neg o_1^2 \lor b^2$

$\neg o_1^3 \lor \neg b^2$

$\neg o_1^3 \lor \neg c^2$

$o_1^2 \lor o_2^2$

$\neg o_1^2 \lor c^2$

$o_1^3 \lor o_2^3$

$\neg o_2^2 \lor \neg b^2$

$b^1 \lor b^2$

$\neg o_2^3 \lor b^2$

$\neg c^1 \lor c^2$

$\neg o_2^2 \lor \neg c^2$

$\neg o_2^3 \lor c^2$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1		0
c^i	1	0		1

	1	2	3
o_1^i	1		
o_2^i	0	1	

Planning as satisfiability

Example: plan search with DPLL

Perform unit resolution and unit subsumption with o_2^2.

$o_1^2 \vee o_2^2$

$o_1^3 \vee o_2^3$

$\neg o_1^2 \vee b^2$

$\neg o_1^3 \vee \neg b^2$

$\neg o_1^3 \vee \neg c^2$

$\neg o_1^2 \vee c^2$

$\neg o_2^2 \vee \neg b^2$

$b^1 \vee b^2$

$\neg o_2^3 \vee b^2$

$\neg c^1 \vee c^2$

$\neg o_2^2 \vee \neg c^2$

$\neg o_2^3 \vee c^2$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1		0
c^i	1	0		1

	1	2	3
o_1^i	1		
o_2^i	0	1	

Planning as satisfiability

Example: plan search with DPLL

Perform unit resolution and unit subsumption with with $\neg b^2$ and $\neg c^2$.

$o_1^3 \vee o_2^3$

$\neg o_1^2 \vee b^2$

$\neg o_1^3 \vee \neg b^2$

$\neg o_1^3 \vee \neg c^2$

$\neg o_1^2 \vee c^2$

$\neg b^2$

$\neg o_2^3 \vee b^2$

$\neg c^2$

$\neg o_2^3 \vee c^2$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1	0	0
c^i	1	0	0	1

	1	2	3
o_1^i	1		
o_2^i	0	1	

Planning as satisfiability

Example: plan search with DPLL

Perform unit resolution and unit subsumption with with $\neg o_1^2$ and $\neg o_2^3$.

$\neg o_1^2$

$\neg o_1^2$

$o_1^3 \vee o_2^3$

$\neg o_2^3$

$\neg o_2^3$

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1	0	0
c^i	1	0	0	1

	1	2	3
o_1^i	1	0	
o_2^i	0	1	0

Planning as satisfiability

Example: plan search with DPLL

Perform unit resolution and unit subsumption with o_1^3.

o_1^3

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1	0	0
c^i	1	0	0	1

	1	2	3
o_1^i	1	0	1
o_2^i	0	1	0

Planning as satisfiability

Example: plan search with DPLL

The formula is satisfiable.

Valuation constructed by the DPLL procedure

	0	1	2	3
b^i	1	1	0	0
c^i	1	0	0	1

	1	2	3
o_1^i	1	0	1
o_2^i	0	1	0

Planning as satisfiability with parallel plans

- Efficiency of satisfiability planning is strongly dependent on the plan length because satisfiability algorithms have runtime $O(2^n)$ where n is the formula size, and formula sizes are linearly proportional to plan length.
- Formula sizes can be reduced by allowing several operators in parallel.
- On many problems this leads to big speed-ups.
- However there are no guarantees of optimality.

Parallel operator application

Definition attempt

Similar to relaxed planning graphs, we consider the possibility of executing several operators simultaneously.

Definition (?)

Let σ be a set of operators (a plan step) and s a state.
Define $app_\sigma(s)$ as the state that is obtained from s by making the literals in $\bigcup_{\langle c,e\rangle\in\sigma}[e]_s$ true.
For $app_\sigma(s)$ to be defined, we require that $s \models c$ for all $o = \langle c, e\rangle \in \sigma$ and $\bigcup_{\langle c,e\rangle\in\sigma}[e]_s$ is consistent.

Unfortunately, the definition is flawed. Why?

Parallel actions

Non-interleavable actions

Example

According to the definition attempt, the operators $\langle a, \neg b\rangle$ and $\langle b, \neg a\rangle$ may be executed simultaneously in state $\{a \mapsto 1, b \mapsto 1\}$, resulting in the state $\{a \mapsto 0, b \mapsto 0\}$.
But this state is not reachable by the two operators sequentially, because executing any one operator makes the precondition of the other false.

Parallel actions

Comparison to relaxed planning tasks

- When discussing relaxed planning tasks, we gave a conservative definition of parallel operator application:
  - It is not guaranteed that each serialization of a plan step σ (or even one of them) leads to the state $app_\sigma(s)$.
  - However, the resulting state of the serialized plan is guaranteed to be at least as good as $app_\sigma(s)$.
- Our general definition attempt was not conservative – not even if we require positive normal form (as the example shows).
- A conservative definition extending the earlier one for relaxed planning tasks is possible, but complicated.
- Instead, we use a semantic definition based on serializations.

Parallel actions

Serializations and semantics

Definition (serialization)

A serialization of plan step $\sigma = \{o_1, \ldots, o_n\}$ is a sequence $o_{\pi(1)}, \ldots, o_{\pi(n)}$ where π is a permutation of $\{1, \ldots, n\}$.

Definition (semantics of plan steps)

A plan step $\sigma = \{o_1, \ldots, o_n\}$ is applicable in a state s iff each serialization of σ is applicable in s and results in the same state s'.
The result of applying σ in s is then defined as $app_\sigma(s) = s'$.

Note: This definition does not extend the earlier definition for relaxed planning tasks.

Parallel plans

Definition (parallel plan)

A parallel plan for a general planning task $\langle A, I, O, G\rangle$ is a sequence of plan steps $\sigma_1, \ldots, \sigma_n$ of operators in O with:

- $s_0 := I$
- For $i = 1, \ldots, n$, step σ_i is applicable in s_{i-1} and $s_i := app_{\sigma_i}(s_{i-1})$.
- $s_n \models G$

Remark: By ordering the operators within each single step arbitrarily, we obtain a (regular, non-parallel) plan.

Parallel plans

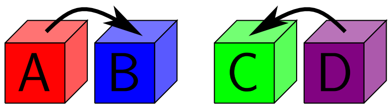
Sufficient conditions

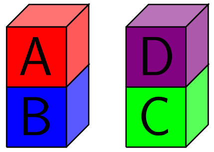
- Testing the condition for parallel applicability is difficult: even testing whether a set σ of operators is applicable in all serializations is co-NP-hard.
- Representing the executability test exactly as a propositional formula seems complicated: doing this test exactly would seem to cancel the benefits of parallel plans.
- Instead, all work on parallel plans so far has used sufficient but not necessary conditions that can be tested in polynomial-time.
- We use a simple syntactic test (which may be overly strict).

Interference

Example

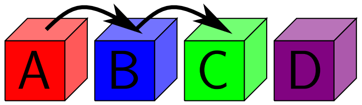
Actions do not interfere





Actions can be taken simultaneously.

Actions interfere



If A is moved first, B will not be clear and cannot be moved.

Interference

Auxiliary definition: affects

Definition (affect)

Let A be a set of state variables and $o = \langle c, e \rangle$ and $o' = \langle c', e' \rangle$ operators over A. Then o affects o' if there is $a \in A$ such that

1. a is an atomic effect in e and a occurs in a formula in e' or it occurs negatively in c', or
2. $\neg a$ is an atomic effect in e and a occurs in a formula in e' or it occurs positively in c'.

Example

$\langle c, d \rangle$ affects $\langle \neg d, e \rangle$ and $\langle e, d \rhd f \rangle$.
$\langle c, d \rangle$ does not affect $\langle d, e \rangle$ nor $\langle e, \neg c \rangle$.

Interference

Definition (interference)

Operators o and o' interfere if o affects o' or o' affects o.

Example

$\langle c, d \rangle$ and $\langle \neg d, e \rangle$ interfere.
$\langle c, d \rangle$ and $\langle e, f \rangle$ do not interfere.

Interference

Sufficient condition for applying a plan step

Lemma

Let s be a state and σ a set of operators so that each operator in σ is applicable in s, no two operators in σ interfere, and $\bigcup_{\langle c,e \rangle \in \sigma} [e]_s$ is consistent.

Then σ is applicable in s and results in the state that is obtained from s by making the literals in $\bigcup_{\langle c,e \rangle \in \sigma} [e]_s$ true.

Parallel operator application

We cannot simply use our current definition of $\tau_A(o)$ within a satisfiability encoding for parallel planning:

- The formula $\tau_A(o)$ completely defines the relationship between current state and successor state when o is applied.
- It leaves no room for applying another operator in sequence.

Basic idea for parallel plan encodings:

- Decouple the parts of the formula that describe what changes from parts that describe what does not change.

Parallel operator application

Representation in propositional logic

Consider the formula $\tau_A(o)$ representing operator $o = \langle c, e\rangle$:

$$\begin{aligned} & c \\ \wedge & \textstyle\bigwedge_{a\in A}((EPC_a(e) \vee (a \wedge \neg EPC_{\neg a}(e))) \leftrightarrow a') \\ \wedge & \textstyle\bigwedge_{a\in A} \neg(EPC_a(e) \wedge EPC_{\neg a}(e)). \end{aligned}$$

This can be logically equivalently written as follows:

$$\begin{aligned} & c \\ \wedge & \textstyle\bigwedge_{a\in A}(EPC_a(e) \rightarrow a') \\ \wedge & \textstyle\bigwedge_{a\in A}(EPC_{\neg a}(e) \rightarrow \neg a') \\ \wedge & \textstyle\bigwedge_{a\in A}((a \wedge \neg EPC_{\neg a}(e)) \rightarrow a') \\ \wedge & \textstyle\bigwedge_{a\in A}((\neg a \wedge \neg EPC_a(e)) \rightarrow \neg a') \end{aligned}$$

This separates the changes from non-changes.

The explanatory frame axioms

The formula states that the only explanation for a changing its value is the application of one operator:

$$\begin{aligned} & \textstyle\bigwedge_{a\in A}((a \wedge \neg a') \rightarrow EPC_{\neg a}(e)) \\ & \textstyle\bigwedge_{a\in A}((\neg a \wedge a') \rightarrow EPC_a(e)) \end{aligned}$$

When several operators could be applied in parallel, we have to consider all operators as possible explanations:

$$\begin{aligned} & \textstyle\bigwedge_{a\in A}((a \wedge \neg a') \rightarrow \bigvee_{i=1}^{n}(o_i \wedge EPC_{\neg a}(e_i))) \\ & \textstyle\bigwedge_{a\in A}((\neg a \wedge a') \rightarrow \bigvee_{i=1}^{n}(o_i \wedge EPC_a(e_i))) \end{aligned}$$

where $\sigma = \{o_1, \dots, o_n\}$ and $e_1, \dots, e_n$ are the respective effects.

Parallel actions

Formula in propositional logic

Definition (plan step application in propositional logic)

Let σ be a plan step. Let $\tau_A(\sigma)$ denote the conjunction of formulae

$$\begin{aligned} & (o \rightarrow c) \\ \wedge & \textstyle\bigwedge_{a\in A}(o \wedge EPC_a(e) \rightarrow a') \\ \wedge & \textstyle\bigwedge_{a\in A}(o \wedge EPC_{\neg a}(e) \rightarrow \neg a') \end{aligned}$$

for all $o = \langle c, e\rangle \in \sigma$ and

$$\begin{aligned} & \textstyle\bigwedge_{a\in A}((a \wedge \neg a') \rightarrow \bigvee_{i=1}^{n}(o_i \wedge EPC_{\neg a}(e_i))) \\ & \textstyle\bigwedge_{a\in A}((\neg a \wedge a') \rightarrow \bigvee_{i=1}^{n}(o_i \wedge EPC_a(e_i))) \end{aligned}$$

where $\sigma = \{o_1, \dots, o_n\}$ and $e_1, \dots, e_n$ are the respective effects.

Correctness

The formula $\tau_A(\sigma)$ exactly matches the definition of $app_\sigma(s)$ provided that no actions in σ interfere.

Lemma

Let s and s' be states and σ a set of operators. Let $v : A \cup A' \cup \sigma \to \{0, 1\}$ be a valuation such that

1. *for all $o \in \sigma$, $v(o) = 1$,*
2. *for all $a \in A$, $v(a) = s(a)$, and*
3. *for all $a \in A$, $v(a') = s'(a)$.*

If σ is applicable in s, then:
$v \models \tau_A(\sigma)$ if and only if $s' = app_\sigma(s)$.

Translation of parallel plans into propositional logic

Definition

Define $\mathcal{R}_2(A, A', O)$ as the conjunction of $\tau_A(O)$ and

$$\neg(o \wedge o')$$

for all $o \in O$ and $o' \in O$ such that o and o' interfere and $o \neq o'$.

Planning as satisfiability

Existence of plans

Definition (bounded step number plans in propositional logic)

Existence of parallel plans of length t is represented by the following formula over propositions $A^0 \cup \cdots \cup A^t \cup O^1 \cup \cdots \cup O^t$ where $A^i = \{ a^i \mid a \in A \}$ for all $i \in \{0, \ldots, t\}$ and $O^i = \{ o^i \mid o \in O \}$ for all $i \in \{1, \ldots, t\}$:

$$\Phi_t^{par} = \iota^0 \wedge \mathcal{R}_2(A^0, A^1, O^1) \wedge \cdots \wedge \mathcal{R}_2(A^{t-1}, A^t, O^t) \wedge G^t$$

where $\iota^0 = \bigwedge_{a \in A, I(a)=1} a^0 \wedge \bigwedge_{a \in A, I(a)=0} \neg a^0$
and G^t is G with propositions a replaced by a^t.

Planning as satisfiability

Existence of plans

Theorem

Let Φ_t^{par} be the formula for $\langle A, I, O, G \rangle$ and plan length t. The formula Φ_t^{par} is satisfiable if and only if there is a sequence of states $s_0, \ldots, s_t$ and plan steps $\sigma_1, \ldots, \sigma_t$, each consisting of non-interfering operators, such that $s_0 = I$, $s_i = app_{\sigma_i}(s_{i-1})$ for all $i \in \{1, \ldots, t\}$, and $s_t \models G$.

Why is optimality lost?

Minimal step count does not imply minimal length

That a plan has the smallest number of steps does not guarantee that it has the smallest number of actions.

- Satisfiability algorithms return any satisfying valuation of Φ_i^{par}, and this does not have to be the one with the smallest number of operators.
- There could be better solutions with more time points.
- Moreover, even optimality in the number of time steps is not guaranteed because the non-interference requirement is only sufficient, but not necessary, for parallel applicability.

Why is optimality lost?

Example

Example

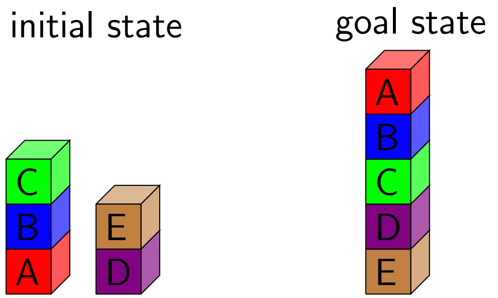
Let I be a state such that $s \models \neg c \wedge \neg d \wedge \neg e \wedge \neg f$.
Let $G = c \wedge d \wedge e$, and let:
$o_1 = \langle \top, c \rangle$
$o_2 = \langle \top, d \rangle$
$o_3 = \langle \top, e \rangle$
$o_4 = \langle \top, f \rangle$
$o_5 = \langle f, c \wedge d \wedge e \rangle$

Now $\pi_1 = \{o_1, o_2, o_3\}$ is a plan with one step, and $\pi_2 = \{o_4\}; \{o_5\}$ is a plan with two steps.

Plan π_1 is optimal with respect to the number of steps, but not with respect to the number of actions, where π_2 is optimal. There is no plan which minimizes both measures.

Planning as satisfiability

Example



The DPLL procedure solves the problem quickly:

- Formulae for lengths 0 to 4 shown unsatisfiable without any search.
- Formula for plan length 5 is satisfiable: 3 nodes in the search tree.
- Plans have 5 to 7 operators, optimal plan has 5.

Planning as satisfiability

Example

```
v0.9 13/08/1997 19:32:47
30 propositions 100 operators
Length 0
Length 1
Length 2
Length 3
Length 4
Length 5
branch on -clear(b)[1] depth 0
branch on clear(a)[3] depth 1
Found a plan.
  0 totable(e,d)
  1 totable(c,b) fromtable(d,e)
  2 totable(b,a) fromtable(c,d)
  3 fromtable(b,c)
  4 fromtable(a,b)
Branches 2 last 2 failed 0; time 0.0
```

Planning as satisfiability

Example

	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
clear(a)	0	0					0	0	0		1	1	0	0	0	1	1	1
clear(b)	0					0	0	0		1	1	0	0	0	1	1	1	0
clear(c)	1	1			0	0	1	1	1	1	0	0	1	1	1	1	0	0
clear(d)	0	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0
clear(e)	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0
on(a,b)	0	0	0			1	0	0	0	0	0	1	0	0	0	0	0	1
on(a,c)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on(a,d)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on(a,e)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on(b,a)	1	1			0	0	1	1	1		0	0	1	1	1	0	0	0
on(b,c)	0	0			1	1	0	0	0	0	1	1	0	0	0	0	1	1
on(b,d)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on(b,e)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on(c,a)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on(c,b)	1			0	0	0	1	1		0	0	0	1	1	0	0	0	0
on(c,d)	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1
on(c,e)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on(d,a)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on(d,b)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on(d,c)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on(d,e)	0	0	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1
on(e,a)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on(e,b)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on(e,c)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
on(e,d)	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
ontable(a)	1	1	1			0	1	1	1	1	1	0	1	1	1	1	1	0
ontable(b)	0	0			0	0	0	0	0		0	0	0	0	0	1	0	0
ontable(c)	0			0	0	0	0	0		0	0	0	0	0	1	0	0	0
ontable(d)	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0
ontable(e)	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1

1. Infer state variable values from initial values and goals.
2. Branch: $\neg$ clear(b)[1].
3. Branch: clear(a)[3].
4. Plan found:

	0	1	2	3	4
fromtable(a,b)	.	.	.	.	1
fromtable(b,c)	.	.	.	1	.
fromtable(c,d)	.	.	1	.	.
fromtable(d,e)	.	1	.	.	.
totable(b,a)	.	.	1	.	.
totable(c,b)	.	1	.	.	.
totable(e,d)	1	.	.	.	.

Final remarks

- All successful satisfiability-based planners use some kind of parallel encoding.
- Sequential encodings are not regarded as competitive with (admissible) heuristic search planners.
- In practice, the presented encoding is further refined to be able to rule out bad variable assignments early in the SAT solving procedure.
- The state-of-the-art SATPLAN06 (formerly SATPLAN04, formerly Blackbox) planner supports a number of different encodings.
- The ones that typically perform best are based on (non-relaxed) planning graphs.