Parallel plans

- Plans are not sequences $o_1, \ldots, o_n$ of operators, but sequences $S_1, \ldots, S_n$ of sets of operators.
- All operators at a given step are applied simultaneously.
- Requirement: result of simultaneous application must be the same as application in any order (interleaving semantics)

Parallel plans: example

Simultaneous actions possible (actions do not interfere):

![Diagram showing simultaneous actions]

Not possible (B not movable when A is on top of it):

![Diagram showing non-simultaneous actions]

Parallel plans

Let $S$ be a set of operators and $s$ a state.
Define $\text{app}_S(s)$ as the result of simultaneously applying all operators $o \in S$ in state $s$:

1. the preconditions of all operators in $S$ must be true in $s$, and
2. the state $\text{app}_S(s)$ is obtained from $s$ by making the literals in $\bigcup_{(p,e) \in S} [e]_s$ true.

Parallel plans

For a set of operators $O$ and an initial state $I$, a parallel plan is a sequence $T = S_1, \ldots, S_l$ of sets of operators such that there is a sequence of states $s_0, \ldots, s_l$ (the execution of $T$) such that

1. $s_0 = I$,
2. $\bigcup_{(p,e) \in S_i} [e]_{s_{i-1}}$ is consistent for every $i \in \{1, \ldots, l\}$,
3. $s_i = \text{app}_S(s_{i-1})$ for $i \in \{1, \ldots, l\}$,
4. for all $i \in \{1, \ldots, l\}$ and $(p,e) \in S_i$ and $S \subseteq S_i \setminus \{o\}$,
   (a) $\text{app}_S(s_{i-1}) \models p$ and
   (b) $[e]_{s_{i-1}} = [e]_{\text{app}_S(s_{i-1})}$.
Parallel plans

**Lemma A** Let $T = S_1, \ldots, S_k, \ldots, S_l$ be a parallel plan. Let $T' = S_1, \ldots, S_k, S_k', \ldots, S_l$ be the parallel plan obtained from $T$ by splitting the step $S_k$ into two steps $S_k^0$ and $S_k^1$ such that $S_k = S_k^0 \cup S_k^1$ and $S_k^0 \cap S_k^1 = \emptyset$.

If $s_0, \ldots, s_k, \ldots, s_l$ is the execution of $T$ then $s_0, \ldots, s_k', s_k, \ldots, s_l$ for some $s_k'$ is the execution of $T'$.

**Theorem** Let $T = S_1, \ldots, S_k, \ldots, S_l$ be a parallel plan. Then any $\sigma = o_1^2; \ldots; o_n^2; \ldots; o_1^1; \ldots; o_n^1$ such that for every $i \in \{1, \ldots, l\}$ the sequence $o_i^1; \ldots; o_i^1$ is a total ordering of $S_i$, is a plan, and its execution leads to the same terminal state as that of $T$.

**Proof:** First, all empty steps can be removed from the parallel plan. By Lemma A non-singleton steps can be split repeatedly to two smaller non-empty steps until every step is singleton and the singleton steps are in the desired order.

Planning as satisfiability: parallel encoding

To obtain valid parallel plans, include in $R_2(P, P')$ the formula

$$\neg(o_i \land o_j)$$

for every $o_i, o_j \in O$ such that $i \neq j$ and there is a state variable $p \in P$ such that

1. $p$ occurs in an effect in $o_i$, and
2. $p$ occurs in a formula in $o_j$ (in the precondition or in the antecedent of a conditional effect in $o_j$).

**Theorem** $S_1, \ldots, S_l$ satisfies the definition of parallel plans.

**Proof Idea:** For every $S \subseteq S_i$, applying $S$ does not change the values of the precondition or antecedents of conditionals of any operator in $S_i \setminus S$, because the state variables in the effects in $S$ are disjoint from those in the formulae.
Conjunctive normal form
Many satisfiability algorithms require formulas in the conjunctive normal form: transformation by repeated applications of the following equivalences.

\[ \neg(\phi \lor \psi) \equiv \neg\phi \land \neg\psi \]
\[ \neg(\phi \land \psi) \equiv \neg\phi \lor \neg\psi \]
\[ \neg\neg\phi \equiv \phi \]
\[ \phi \lor (\psi_1 \land \psi_2) \equiv (\phi \lor \psi_1) \land (\phi \lor \psi_2) \]

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

EXAMPLE: \((A \lor \neg B \lor C) \land (\neg C \lor \neg B) \land A\)

The unit resolution rule
From \(l_1 \lor l_2 \lor \cdots \lor l_n\) (here \(n \geq 1\)) and \(\neg l_i\) infer \(l_2 \lor \cdots \lor l_n\).

EXAMPLE: From \(A \lor B \lor C\) and \(\neg A\) infer \(B \lor C\).

SPECIAL CASE: from \(A\) and \(\neg A\) we get the empty clause \(\bot\) ("disjunction consisting of zero disjuncts").

Satisfiability test by the Davis-Putnam procedure
1. Let \(C\) be a set of clauses.
2. For all clauses \(l_1 \lor l_2 \lor \cdots \lor l_n \in C\) and \(\neg l_i \in C\), remove \(l_1 \lor l_2 \lor \cdots \lor l_i\) from \(C\) and add \(l_2 \lor \cdots \lor l_n\) to \(C\).
3. For all clauses \(l_1 \lor l_2 \lor \cdots \lor l_n \in C\) and \(l_1 \in C\), remove \(l_1 \lor l_2 \lor \cdots \lor l_n\) from \(C\). (UNIT SUBSUMPTION)
4. If \(\bot \in C\), return FALSE.
5. If \(C\) contains only unit clauses, return TRUE.
6. Pick some \(p \in P\) such that \(\{p, \neg p\} \cap C = \emptyset\)
7. Recursive call: if \(C \cup \{p\}\) is satisfiable, return TRUE.
8. Recursive call: if \(C \cup \{\neg p\}\) is satisfiable, return TRUE.
9. Return FALSE.

Planning as satisfiability: example
\(\text{clear}(C), \text{on}(C,B), \text{on}(B,A), \text{ontable}(A), \text{clear}(E), \text{on}(E,D), \text{ontable}(D)\) are initially true (there are two stacks, CBA and ED.)

The goal is \((\text{on}(A,B) \land \text{on}(B,C) \land \text{on}(C,D) \land \text{on}(D,E))\)

The Davis-Putnam procedure solves the problem quickly:
- Formulae for lengths 1 to 4 shown unsatisfiable by unit resolution.
- Formula for plan length 5 is satisfiable: 3 nodes in the search tree.
Planning as satisfiability: example

v0.9 13/08/1997 19:32:47
30 propositions 100 operators
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

on(c,e) FFFFFF FFFFFF FFFFFF
on(d,a) FFFFFF FFFFFF FFFFFF
on(d,b) FFFFFF FFFFFF FFFFFF
on(d,c) FFFFFF FFFFFF FFFFFF
on(d,e) FTTTTT FTTTTT FTTTTT
on(e,a) FFFFFF FFFFFF FFFFFF
on(e,b) FFFFFF FFFFFF FFFFFF
on(e,c) FFFFFF FFFFFF FFFFFF
on(e,d) FFFFFF FFFFFF FFFFFF
ontable(a) FTT F TTTTF TTTTF
ontable(b) FFF F FFFF FFFFFF
ontable(c) F FFF FFFF FFFFFF
ontable(d) FTTTTT FTTTTT FTTTTT
ontable(e) FTTTTT FTTTTT FTTTTT

012345 012345 012345
012345

clear(a) FF FFF TT FTTTTT
clear(b) F F FFF TFF FTTTTT
clear(c) TT FF TTTTT TTFFFFF
clear(d) FTTFFF TTFFFF FTTFFF
clear(e) TTTFFF TTFFFF TTFFFF
on(a,b) FFFF T FFFFFF FFFFFF
on(a,c) FFFFFF FFFFFF FFFFFF
on(a,d) FFFFFF FFFFFF FFFFFF
on(a,e) FFFFFF FFFFFF FFFFFF
on(b,a) TT FF TTT FF TTTFFF
on(b,c) FF TT FTTTT TTFFFF
on(b,d) FFFFFF FFFFFF FFFFFF
on(b,e) FFFFFF FFFFFF FFFFFF
on(c,a) FFFFFF FFFFFF FFFFFF
on(c,b) T FFF TT FFF TTFFFF
on(c,d) FTTTT TTFFFF FTTTTT
on(c,e) FFFFFF FFFFFF FFFFFF
on(d,a) FFFFFF FFFFFF FFFFFF
on(d,b) FFFFFF FFFFFF FFFFFF
on(d,c) FFFFFF FFFFFF FFFFFF
on(d,e) FTTTTTT TTFFFFF TTFFFF
on(e,a) FFFFFF FFFFFF FFFFFF
on(e,b) FFFFFF FFFFFF FFFFFF
on(e,c) FFFFFF FFFFFF FFFFFF
on(e,d) FFFFFF FFFFFF FFFFFF
ontable(a) TTT F TTTTF TTTTF
ontable(b) FFF F FFFF FFFFFF
ontable(c) F FFF FFFF FFFFFF
ontable(d) FTTTT TTTTT TTFFF
ontable(e) FTTTTT FTTTTT FTTTTT