1 Definition: Transition System

A **transition system** is a 6-tuple $T = \langle S, L, c, T, s_0, S^\star \rangle$ where:

- S: finite set of states
- L: finite set of transition labels
- $c : L \mapsto \mathbb{R}_0^+$: label cost function
- $T \subseteq S \times L \times S$: transition relation
- $s_0 \in S$: initial state
- $S^\star \subseteq S$: set of goal states

1.1 Forms and Properties

1.1.1 Heuristics

- Admissable: $h(s) \leq h^*(s)$
- Consistent: $h(s) \leq c(s, s') + h(s')$
- Goal aware: $h(s \in S^\star) = 0$
- Safe: $h(s) = \infty \to h^*(s) = \infty$

1.1.2 Task

- Positive normal form: All ops and goal are positive and flat
  - o is positive if pre(o) and eff(o) are positive
  - A logical proposition is positive if $\neg$ doesn't appear (including $\leftarrow$ and $\leftrightarrow$
  - o is flat, if eff(o) is flat (i.e. contains only atomics or $(x \rhd y)$)
- STRIPS: If all ops are STRIPS and goal follows: $\bigwedge_{v \in V} v$
  - o is STRIPS if pre(o) follows same form, and eff(o) is atomic.
- Algorithm is **sound** $\to$ plans are correct, and "unsolveable" answer is correct.

1.2 On-Set and Dominating states

- The on-set is the set of propositional variables that are true in a interpretation.
- Domainiting interpretations for $\text{on}(s) \subseteq \text{on}(s'), s, s' \in S$

1.3 Complexity

- P $\subseteq$ NP $\subseteq$ PSPACE $=$ NPSPACE
- (PlanEx)istance $\underset{p}{\leq}$ (B)ounded (C)ost PlanEx
- (PlanEx)istance $\in$ PSPACE
- True for both optimal and satisfycing

2 Satisfiability & Equivalence

φ satisfiable iff $\exists I : I \vDash \varphi$

φ valid iff $\forall I : I \vDash \varphi$

$\varphi \vDash \psi$ iff $\forall I : I \vDash \varphi \to I \vDash \psi$

$\varphi \equiv \psi$ iff $\varphi \vDash \psi \land \psi \vDash \varphi$

3 STRIPS Regression

Let $\varphi = \varphi_1 \land \ldots \land \varphi_n$ be a conjunction of atoms, and o's add effects be $\{a_1, \ldots, a_k\}$, and delete effects $\{d_1, \ldots, d_l\}$

$\text{sregr}(\varphi, o) := \begin{cases} \bot \text{ if } \exists (i,j) \varphi_i = d_j \\ \text{pre}(o) \land (\{\varphi_1, \ldots, \varphi_n\} / \{a_1, \ldots, a_k\}) \text{ otherwise} \end{cases}$

4 SAT Planning style-algorithm

Algorithm 1: SAT Planning

```
procedure SATPLAN("Pi")
    for T in {0, 1, 2, ...} do
        φ ← build_sat_formula(Π, T)
        I ← sat_solver(φ)
        if I != none then
            return extract_plan(Π, T, I)
        end
    end
end
```

4.1 SAT: Operator Selection Clauses

- o_j^i (operator chosen at step i)
- $o_1^i \lor \ldots \lor o_n^i$ for $1 \leq i \leq T$
- $\neg o_j^i \lor \neg o_k^i$ for $1 \leq i \leq T, 1 \leq j < k \leq n$ (at most one operator per step)
  - This is equal to $\neg\left(o_j^i \land o_k^i\right)$

4.2 Transition Clauses

Precondition:

- $\neg o^i \lor \text{pre}(o)^{i-1}$ for $1 \leq i \leq T, o \in O$

Positive/Negative Effects Clauses:

- $\neg o^i \lor \neg \alpha^{i-1} \lor v^i$
- $\neg o^i \lor \alpha^{i-1} \lor \neg \delta^{i-1} \lor \neg v^i$

Positive/Negative Frame Clauses:

- $\neg o^i \lor \neg v^{i-1} \lor \delta^{i-1} \lor v^i$
- $\neg o^i \lor \alpha^{i-1} \lor v^{i-1} \lor \neg v^i$

where $\alpha = \text{effcond}(v, \text{eff}(o))$ $\delta = \text{effcond}(\neg v, \text{eff}(o))$

5 BDD Complexity

	Hash table	Formula	BDD
$s \in S?$	$O(k)$	$O(\lVert S \rVert)$	$O(k)$
$S := S \cup \{s\}$	$O(k)$	$O(k)$	$O(k)$
$S := S \setminus \{s\}$	$O(k)$	$O(k)$	$O(k)$
$S \cup S'$	$O(k\lvert S \rvert + k\lvert S' \rvert)$	$O(1)$	$O(\lVert S \rVert \lVert S' \rVert)$
$S \cap S'$	$O(k\lvert S \rvert + k\lvert S' \rvert)$	$O(1)$	$O(\lVert S \rVert \lVert S' \rVert)$
$S \setminus S'$	$O(k\lvert S \rvert + k\lvert S' \rvert)$	$O(1)$	$O(\lVert S \rVert \lVert S' \rVert)$
$\overline{S}$	$O(k2^k)$	$O(1)$	$O(\lVert S \rVert)$
$\{s \mid s(v) = \mathbf{T}\}$	$O(k2^k)$	$O(1)$	$O(1)$
$S = \emptyset?$	$O(1)$	co-NP-complete	$O(1)$
$S = S'?$	$O(k\lvert S \rvert)$	co-NP-complete	$O(1)$
$\lvert S \rvert$	$O(1)$	#P-complete	$O(\lVert S \rVert)$

6 BDD Operators

6.1 Conditioning

Conditioning variable v in formula φ to T or F:

- $\varphi\left[\frac{T}{v}\right]$ or $\varphi\left[\frac{F}{v}\right]$: restrict v to a given value
- Time: $O(|\varphi|)$

6.2 Forgetting

Forgetting (existential abstraction): allow both $v = T$ and $v = F$ and eliminate v.

- On formulas: $\exists v \varphi = \varphi\left[\frac{T}{v}\right] \lor \varphi\left[\frac{F}{v}\right]$
- On sets: $\exists v S = S\left[\frac{T}{v}\right] \cup S\left[\frac{F}{v}\right]$
- Time: $O(|\varphi|)$

6.3 Renaming

Renaming X to Y in formula φ, written $\varphi[X \to Y]$: replace all X by Y in φ (Y not present in φ).

- Time: $O(|\varphi|)$

7 BDD Transitions

7.1 Transition BDD

$T_{V(O)} = \bigvee_{o \in O} t_{V(o)}$

$\cdot t_{V(O)} = \text{pre}(o) \land \bigwedge_{v \in V} (\text{effcond}(v, e) \lor (v \land \neg \text{ effcond}(\neg v, e)) \leftrightarrow v')$

7.2 Apply

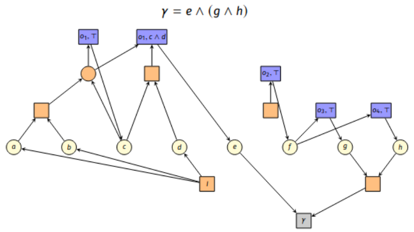
Algorithm 2: BDD Apply

```
procedure APPLY(reached, O)
    B ← T_V(O)
    B ← bdd-intersection(B, reached)
    for v ∈ V do
        B ← bdd-forget(B, v)
    end
    for v ∈ V do
        B ← bdd-rename(B, v', v)
    end
    return B
end
```

By then taking the union of the out and the previous reached, you get the reached for the following timestep.

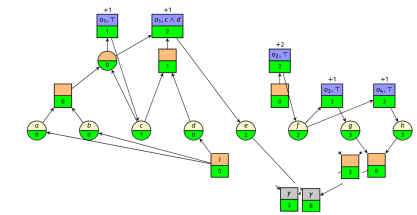
8 Relaxed Task Graph (RTG)

$o_1 = \langle c \lor (a \land b), c \land ((c \land d) \rhd e), 1 \rangle$
$o_2 = \langle \top, f, 2 \rangle$
$o_3 = \langle f, g, 1 \rangle$
$o_4 = \langle f, h, 1 \rangle$



8.1 $h^{\max}$ & h^{add}

- $h^{\max} \leq h^+ \leq h^{\text{FF}} \leq h^{\text{add}}$
- $h^{\max(s)} = \infty \leftrightarrow h^{+(s)} = \infty \leftrightarrow h^{\text{FF}(s)} = \infty \leftrightarrow h^{\text{add}(s)} = \infty$
- $h^{\max}$ and h^{add} $\to$ admissible and consistent
- h^+ and h^{FF} $\to$ NOT admissible and consistent
- All are safe and goal-aware.



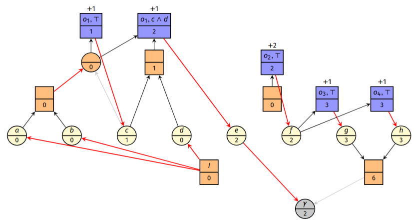
Above, only nodes where $h^{\max}$ (left) and h^{add} (right) differ are recorded.

- $h^{\max}$: Pick the max predecessor at AND node, and the min at OR
- h^{add}: Add the predecessors at AND node, and pick the min at OR
- Both can be computed efficently by expanding the minimum/ newest node that CAN be updated

8.2 h^{FF} and Best Achiever Graphs (G)

- BAG can be achieved by removing all incoming edges into a OR node, except the minimum cost one

best achievers for h^{add}; modified goal $e \lor (g \land h)$



- h^{FF} can be achieved by adding all operators participating in the G^{add} for h^{add}
- G are also useful for analysis when h^{add} overapprox and when $h^{\max}$ under approx.

9 Invariant/Mutex/FDR

- Validating invariant is AS HARD as planning.
- Mutex group is a set of variables where AT MOST one can be true
- A Mutex cover is a set of mutex groups where each variable occurs in exactly one group
- A mutex group is positive if it contains no negations of variables

9.1 Mutex-based Reformulation of Propositional

Given a conflict-free propositional planning task Π w/ positive mutex cover $\{G_1, \ldots, G_N\}$

- In all condition where variable $v \in G_i$ occurs, replace v with $v_{G_i} := v$
- In all effects e where variable $v \in G_i$ occurs,
  - Replace all atomic add effects v with $v_{G_i} := v$
  - Replace all atomic delete effects $\neg v$ with:
    - $\left(v_{G_i} = v \land \neg \bigvee_{v' \in G_i \setminus \{v\}} \text{effcond}(v', e)\right) \rhd v_{G_i} := \text{none}$
    - Practically, this means, if v_{G_i} is being deleted AND IS NOT BEING SET TO ANOTHER VARIABLE, set it to none. This is keep it conflict-free.

The consistency condition $\text{consist}(e)$ prohibits two simultaneous assignments to the same mutex group.
I.e. $\neg(\text{effcond}(v := d, e) \land \text{effcond}(v := d', e))$

9.2 SAS+

An operator of an FDR operator is a SAS+ operator if

- $\text{pre}(o)$ is a satisfiable conjunction of atoms, and
- $\text{eff}(o)$ is a conflict-free conjunction of atomic effects.

An FDR task is a SAS+ task if all operators are SAS+ and the goal is a satisfiable conjunction of atoms