1 Definition: Transition System

A **planning task** is a 4-tuple $\Pi = \langle V, I, O, \gamma \rangle$

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: $h(s) = \infty \rightarrow h^*(s) = \infty$

1.1.2 Task forms and Misc.

- 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.
- i-g Form: STRIPS form. $\{i, g\} \subseteq V$. $I := \{i\}$. $\gamma := \{g\}$. $\forall (o \in O)(|\mathrm{pre}(o)| > 0)$
  - Any task can be made i-g form trivially, if already STRIPS.
- Algorithm is **sound** $\rightarrow$ 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 $\mathrm{on}(s) \subseteq \mathrm{on}(s'), s, s' \in S$

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 $\forall I : I \vDash \varphi \rightarrow I \vDash \psi$

$\varphi \equiv \psi$ iff $\varphi \vDash \psi \wedge \psi \vDash \varphi$

3 STRIPS Regression

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

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

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

4.2 Transition Clauses

Precondition:

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

Positive/Negative Effects Clauses:

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

Positive/Negative Frame Clauses:

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

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

5 BDD Complexity

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

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: $\exists v \varphi = \varphi\left[\frac{T}{v}\right] \vee \varphi\left[\frac{F}{v}\right]$
- 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 $\varphi[X \rightarrow Y]$: replace all X by Y in φ (Y not present in φ).

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

7 BDD Transitions

7.1 Transition BDD

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

$\cdot t_{V(O)} = \mathrm{pre}(o) \wedge \bigwedge_{v \in V} (\mathrm{effcond}(v,e) \vee (v \wedge \neg\, \mathrm{effcond}(\neg v, e)) \leftrightarrow v')$

7.2 Apply

Algorithm 2: BDD Apply

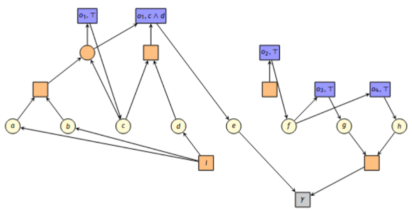
```
procedure APPLY(reached, O)
    B ← T_V(O)
    B ← bdd-intersection(B, reached)
    for v ∈ V do
        B ← bdd-forget(B, v)
    end
    for v ∈ V do
        B ← bdd-rename(B, v′, v)
    end
    return B
end
```

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)

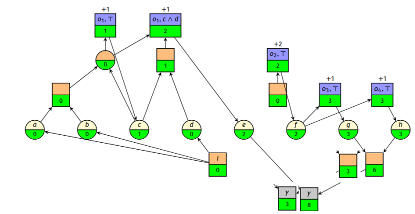
$o_1 = \langle c \vee (a \wedge b), c \wedge ((c \wedge d) \rhd e), 1 \rangle$
$o_2 = \langle \top, f, 2 \rangle$
$o_3 = \langle f, g, 1 \rangle$
$o_4 = \langle f, h, 1 \rangle$

$\gamma = e \wedge (g \wedge h)$



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

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



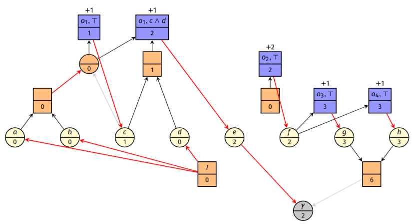
Above, only nodes where $h^{\max}$ (left) and h^{add} (right) differ are recorded.

- $h^{\max}$: 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 $e \vee (g \wedge h)$



- 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:
    - $\left(v_{G_i} = v \wedge \neg \bigvee_{v' \in G_i \setminus \{v\}} \mathrm{effcond}(v', e)\right) \rhd v_{G_i} := \mathrm{none}$
    - 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 consist(e) prohibits two simultaneous assignments to the same mutex group.
I.e. $\neg(\mathrm{effcond}(v := d, e) \wedge \mathrm{effcond}(v := d', e))$

9.2 SAS+

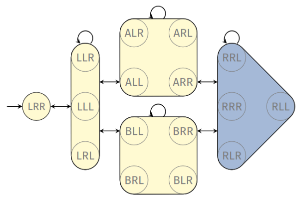
An operator of an FDR operator is a SAS+ operator if

- pre(o) is a satisfiable conjunction of atoms, and
- 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

10 Abstraction

- $s \in \gamma \rightarrow \alpha(s) \in \gamma_\alpha$
- $\langle s, o, s' \rangle \in \mathcal{T} \rightarrow \langle \alpha(s), o, \alpha(s') \rangle \in \mathcal{T}_\alpha$
- Abstraction are composable, i.e. $(\beta \circ \alpha)$ is a valid abstraction.
- Abstraction are surjective.
- Abstraction uses coarsening/refinement terminology.
- $h^{\beta \circ \alpha} \leq h^\alpha \leq h^*$
  - h^α is safe, goal-aware, admissible and consistent.



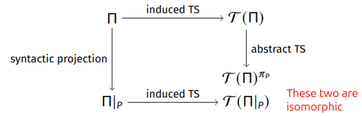
$h^\alpha(\{p \mapsto \mathrm{L}, t_\mathrm{A} \mapsto \mathrm{R}, t_\mathrm{B} \mapsto \mathrm{R}\}) = 3$

10.1 Additivity

- Orthogonal α_1 & α_2: If $\forall (t \in \mathcal{T})(\alpha_1(s) = \alpha_2(t)) \vee (\alpha_2(s) = \alpha_2(t))$, where $t = \langle s, \ell, t \rangle$
- Affect α for ℓ, if $\langle \alpha(s), \ell, \alpha(t) \rangle$, where $\alpha(s) \neq \alpha(t)$
- Also ortogonal if no label affects both abstractions.
- The sum of orthogonal h^α is safe, goal-aware, admissible and consistent.

10.2 Projections & Pattern Databases

- A projection (π_P) is a special kind of abstraction
- $\pi_P : S \rightarrow S'$ is defined as$\pi_{P(s)} := s|_P$ (where $s|_{P(v)} := s(v)$ for all v)
  - I.e. we condition a state on a single variable assignment.
- The heuristic induced by π_P, we call a PDB heuristic (h^P)
- Syntatic projections ($\Pi|_P$), gives the projected planning task, by practically, removing the variables in the projection, from $\langle P, I|_P, \{o|_P, o \in O\}, \gamma|_P \rangle$



10.3 PDB Lookup

- PDBs are precomputed before search.
- Is effective done via perfect hashing.
  - $N_i := \Pi_{j=1}^{i-1} |\mathrm{dom}(v_j)|$
  - PDB-index$(s) := \Sigma_{i=1}^{k} N_i \cdot s(v_i)$

11 Merge and Shrink

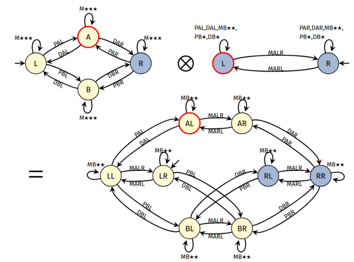
- Idea:
  - 1) Project Π to atomic projection
  - 2) Merge two of the resulting transition systems ($\mathcal{T}|_P$)
  - 3) Shrink combined $\mathcal{T}'$ by abstracting more states
  - 4) Pick the result as the first transition system and go to step (2)

Algorithm 3: Merge and Shrink

```
procedure MERGE-AND-SHRINK(Π,B)
    F ← F(Π)
    while |F| > 1 do
        "type" ← pick-merge-or-shrink(F)
        if type = merge then
            T1 ← pick(F)
            T2 ← pick(F \ {T1})
            F ← (F \ {T1, T2}) ∪ {T1 ⊗ T2}
        end
        if type = shrink then
            T ← pick(F)
            β ← pick-abstraction(B)
            F ← (F \ {T}) ∪ {T^β}
        end
    end
    return F[0]
end
```

$\mathcal{T}^{\pi_{\{\mathrm{package}\}}} \otimes \mathcal{T}^{\pi_{\{\mathrm{truck\ A}\}}}$:

$S_\otimes = S_1 \times S_2$



11.1 Factored Transition System (FTS or F)

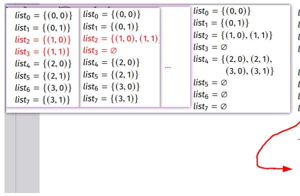
- A finite set $F = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$, where all share ℓ and cost(s).
- FTS induced by Π is $F(\Pi) = \{\mathcal{T}^{\pi_v} \mid v \in V\}$
- $\bigotimes F \sim \mathcal{T}(\Pi)$ is the transition system that induced it.

11.2 Merge Strategies

- f-preserving strategy
  - Combine nodes with identical g and h value
    - *Rational*: Preserves h and overall graph shape
  - Tie-breaking criterion, prefer merging high $g + h$
    - *Rational*: High values heuristic estimates are less likely to be explored by A*, so it can be more inprecise.

11.3 Merge and Shrink - Effective Shrink

- This is done by first converting the combined table (which is a cross product of two abstractions) into a linked list. And then...



	$s_2 = 0$	$s_2 = 1$
$s_1 = 0$	0	1
$s_1 = 1$	2	2
$s_1 = 2$	3	3
$s_1 = 3$	3	3

11.4 Merge and Shrink Lookup

- Looking up the heuristic value in a MAS system requires looking up from first single variables and then into the larger merge, i.e.

At the end, our heuristic is represented by six tables:

- three one-dimensional tables for the atomic abstractions:

$T_{package}$	L	R	A	B
	0	1	2	3

$T_{truck\ A}$	L	R
	0	1

$T_{truck\ B}$	L	R
	0	1

- two tables for the two merge and subsequent shrink steps:

$T^1_{m\&s}$	$s_2 = 0$	$s_2 = 1$
$s_1 = 0$	0	1
$s_1 = 1$	2	2
$s_1 = 2$	3	3
$s_1 = 3$	3	3

$T^2_{m\&s}$	$s_2 = 0$	$s_2 = 1$
$s_1 = 0$	1	1
$s_1 = 1$	1	0
$s_1 = 2$	2	2
$s_1 = 3$	3	3

- one table with goal distances for the final transition system:

T_h	$s = 0$	$s = 1$	$s = 2$	$s = 3$
$h(s)$	3	2	0	1

11.5 Label Reduction

11.5.1 Definition

- A label reduction $\langle \lambda : L \to L', c' : L \to \mathbb{R}^+ \rangle$, such that $\ell \in L, c'(\lambda(\ell)) \leq c(\ell)$
- The label-reduced transition system $\mathcal{T}^{\langle \lambda, c' \rangle} = \langle S, L', c', \{\langle s, \lambda(\ell), t \rangle \mid \langle s, \ell, t, \rangle \in T\}, s_0, S_\star \rangle$

11.5.2 Properties

- ℓ is alive in F if all $\mathcal{T}' \in F$ have ℓ, dead otherwise
  - ℓ locally subsumes ℓ' in $\mathcal{T}$ if for all transition $\langle s, \ell', t \rangle$ there is also $\langle s, \ell, t \rangle$
  - It also globally subsumes if this is true in all $\mathcal{T} \in F$
  - ℓ and ℓ' are locally equivilant if ℓ locally subsumes ℓ' and vice versa.
  - ℓ and ℓ' are $\mathcal{T}$-combinable if there are locally equivilant in all transition systems $\mathcal{T}' \in F \setminus \{\mathcal{T}\}$

11.5.3 Exact Label Reduction

- The label reduction is exact (No loss in information), if for all $\ell_1, \ell_2 \in L$:
  - Either ℓ_1 or ℓ_2 globally subsumes the other
  - ℓ_1 and ℓ_2 are $\mathcal{T}$-combinable for some $\mathcal{T} \in F$

12 Landmarks

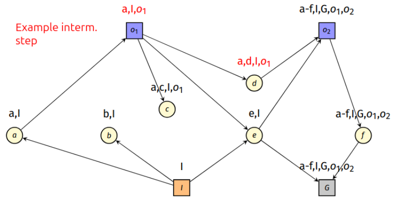
- Fact landmark: This must be visited at least once
- Disjunctive Action: One action from the set must be performed
- Network flow: Fact consumption should be balanced.

12.1 Relaxed Task Graph Landmarks

- Causal Landmark λ: for I if $\gamma \nvDash \lambda$ OR if for all plans $\langle o_1, ..., o_n \rangle$ at least one $\mathrm{pre}(o_i) \vDash \lambda$
- Causal fact landmark: Same as above, but $\lambda \coloneqq v$
- To calculate the RTG landmarks, first instantiate all nodes with all potential landmarks. Then perform this on RTG

$\mathrm{LM}(n) = \{n\} \bigcap_{n' \to n \in A} \mathrm{LM}(n')$ if $\mathrm{type}(n) = \vee$

$\mathrm{LM}(n) = \{n\} \bigcup_{n' \to n \in A} \mathrm{LM}(n')$ if $\mathrm{type}(n) = \wedge$



$LM(d) = \{d\} \cup LM(o_1)$

12.2 Minimum Hitting Set

- The minimum hitting set, the minimal cost set of operators such that a for a set of set of operators, each set contains one of the operators selected.
- This is relevant for combining disjunctive action landmarks, which is not admissible if additively combined.

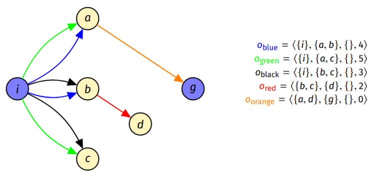
12.3 LM-Cut

12.3.1 Justification Graph

- Precondition choice function: $P : O \to V$, maps any operator in a task Π to one of it's preconditions.
- Justification graph: $\langle V, E \coloneqq \{\langle P(o), a \rangle \mid o \in O, a \in \mathrm{add}(o)\}\rangle$

Example (Precondition Choice Function)

$P(o_{blue}) = P(o_{green}) = P(o_{black}) = i, P(o_{red}) = b, P(o_{orange}) = a$

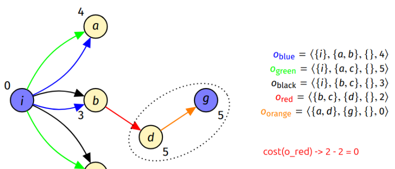


- **Cut** ($C \subseteq E$): A subset of edges, such that ALL paths $i \xrightarrow{*} g$ contains $e \in C$. Doesn't have to include all.

12.4 LM-Cut Algorithm

Initialize $h^{\text{LM-Cut}}(I) \coloneqq 0$. Then iterate:

- Compute $h^{\max}$ using a RTG. Stop if $h^{\max}(g) = 0$
- Compute Justification graph for the P/pcf that chooses the precondition with the MAXIMAL $h^{\max}$ value.
- Determine goal zone V_g (i.e. all nodes with a zero cost path to g).
- Compute the cut L that contains labels of all edges $\langle v, o, v' \rangle$ such that $v \notin V_g, v' \in V_g$, and v CAN be reached from i without traversing V_g. *It is guaranteed that* $\mathrm{cost}(L) > 0$
- Increase $h^{\text{LM-Cut}}(I)$ by $\mathrm{cost}(L)$ (i.e. the cost of the cut).
- Decrease $\mathrm{cost}(o)$ by $\mathrm{cost}(L)$ for all $o \in L$.



round	$P(o_{orange})$	$P(o_{red})$	landmark	cost
1	d	b	$\{o_{red}\}$	2
			$h^{\text{LM-cut}}(I)$	2

13 Integer and Linear Programs