

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 $T = \langle S, L, c, T, s_0, S^* \rangle$ 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
- $S^* \subseteq S$: set of goal states

1.1 Forms and Properties

1.1.1 Heuristics

- Admissible: $h(s) \leq h^*(s)$
- Consistent: $h(s) \leq c(s, s') + h(s')$
- Goal aware: $h(s \in S^*) = 0$
- 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 $\text{pre}(o)$ and $\text{eff}(o)$ are positive
 - A logical proposition is positive if \neg doesn't appear (including \leftarrow and \leftrightarrow)
 - o is flat, if $\text{eff}(o)$ is flat (i.e. contains only atomics or $(x \triangleright y)$)

- STRIPS: If all ops are STRIPS and goal follows: $\bigwedge_{u \in V} v$
- o is STRIPS if $\text{pre}(o)$ follows same form, and $\text{eff}(o)$ is atomic.

- i-g Form: STRIPS form. $\{i, g\} \subseteq V$. $I := \{i\}$.
 $\gamma := \{g\}$. $\forall (o \in O)(|\text{pre}(o)| > 0)$
 Any task can be made i-g form trivially, if already STRIPS.

- Transition Normal Form (TNF): $\forall (o \in O)(\text{vars}(\text{eff}(o)) = \text{vars}(\text{pre}(o)))$ and $\text{vars}(\gamma) = V$.
 - This can be achieved by 1) add auxiliary u to every $\text{dom}(v)$
 - 2) For each variable and value, add an operator that converts it to u for zero cost
 - 3) For all o , if a variable is in pre, but not in eff add it with the same value. If v in eff but not in pre, add $v = u$ in pre

- Algorithm is **sound** \rightarrow plans are correct, and "unsolvable" 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.
- Domaining interpretations for $\text{on}(s) \subseteq \text{on}(s')$, $s, s' \in S$

1.3 Complexity

- $P \subseteq NP \subseteq \text{PSPACE} = \text{NPSpace}$
- (PlanEx)istance $\leq (B)$ ounded (C)ost PlanEx
- (PlanEx)istance $\in \text{PSPACE}$
- True for both optimal and satisfying
- Planning is P in the number of states.

1.4 Search

- Uninformed: DFS, BFS, Iterative DFS
- Heuristic: Greedy BFS, A*, W-A*, IDA*
- Local Heuristic: Hill climbing, Sim. annealing, Beam

2 Satisfiability & Equivalence

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

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

$\varphi \models \psi$ iff $\forall I: I \models \varphi \rightarrow I \models \psi$

$\varphi \equiv \psi$ iff $\varphi \models \psi \wedge \psi \models \varphi$

3 STRIPS Regression

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

$\text{sregt}(\varphi, o) := \begin{cases} \perp & \text{if } \exists(i, j) \varphi_i = d_j \\ \text{pre}(o) \wedge (\{\varphi_1, \dots, \varphi_n\} \setminus \{a_1, \dots, a_k\}) & \text{otherwise} \end{cases}$

4 SAT Planning style-algorithm

Algorithm 1: SAT Planning

```

1:  procedure SATPLAN("Pi")
2:    for  $T$  in  $\{0, 1, 2, \dots\}$  do
3:       $\varphi \leftarrow \text{build\_sat\_formula}(\Pi, T)$ 
4:       $I \leftarrow \text{sat\_solver}(\varphi)$ 
5:      if  $I \neq \text{none}$  then
6:        return  $\text{extract\_plan}(\Pi, T, I)$ 
7:    end
8:  end
9:  end

```

4.1 SAT: Operator Selection Clauses

- o_j^i (operator chosen at step i)
- $o_1^i \vee \dots \vee o_n^i$ 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 $\neg(o_j^i \wedge o_k^i)$

4.2 Transition Clauses

Precondition:

- $\neg o^i \vee \text{pre}(o)^{i-1}$ 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 $\alpha = \text{effcond}(v, \text{eff}(o))$ $\delta = \text{effcond}(\neg v, \text{eff}(o))$

5 BDD Complexity

	Hash table	Formula	BDD
$s \in S?$	$O(h)$	$O(S)$	$O(h)$
$S := S \cup \{s\}$	$O(h)$	$O(h)$	$O(h)$
$S := S \setminus \{s\}$	$O(h)$	$O(h)$	$O(h)$
$S \cup S'$	$O(h S + h S')$	$O(1)$	$O(S S')$
$S \cap S'$	$O(h S + h S')$	$O(1)$	$O(S S')$
$S \setminus S'$	$O(h S + h S')$	$O(1)$	$O(S S')$
\bar{S}	$O(h^2)$	$O(1)$	$O(S)$
$\{s\} s(v) = T$	$O(h^2)$	$O(1)$	$O(1)$
$S = \emptyset?$	$O(1)$	co-NP-complete	$O(1)$
$S = S'?$	$O(h S)$	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

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

$t_{V(O)} = \text{pre}(o) \wedge \bigwedge_{u \in V} (\text{effcond}(v, u) \vee (u \wedge \neg \text{effcond}(\neg v, u)) \rightarrow u')$

7.2 Apply

Algorithm 2: BDD Apply

```

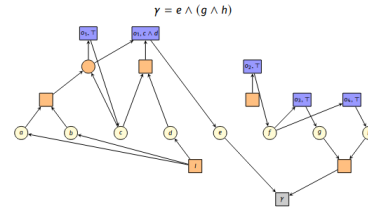
1:  procedure APPLY(reached,  $O$ )
2:     $B \leftarrow T_{V(O)}$ 
3:     $B \leftarrow \text{bdd-intersection}(B, \text{reached})$ 
4:    for  $v \in V$  do
5:       $B \leftarrow \text{bdd-forget}(B, v)$ 
6:    end
7:    for  $v \in V$  do
8:       $B \leftarrow \text{bdd-rename}(B, v', v)$ 
9:    end
10:   return  $B$ 
11: end

```

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

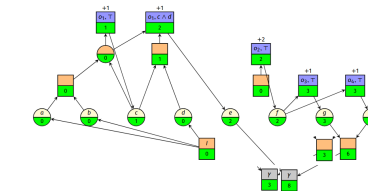
8 Relaxed Task Graph (RTG)

$o_1 = (c \vee (a \wedge b), c \wedge ((c \wedge d) \triangleright e), 1)$
 $o_2 = (T, f, 2)$
 $o_3 = (f, g, 1)$
 $o_4 = (f, h, 1)$



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

- $h^{\text{max}} \leq h^* \leq h^{\text{FF}} \leq h^{\text{add}}$
- $h^{\text{max}}(s) = \infty \leftrightarrow h^+(s) = \infty \leftrightarrow h^{\text{FF}}(s) = \infty \leftrightarrow h^{\text{add}}(s) = \infty$
- h^{max} and $h^{\text{add}} \rightarrow$ admissible and consistent
- h^+ and $h^{\text{FF}} \rightarrow$ 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 efficiently 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

- 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 $\{G_1, \dots, G_N\}$

- 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:
 - $(v_{G_i} = v \wedge \bigvee_{v' \in G_i, v' \neq v} \text{effcond}(v', e)) \triangleright v_{G_i} := \text{none}$
 - 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. $\neg(\text{effcond}(v := d, e) \wedge \text{effcond}(v := d', e))$

9.2 SAS+

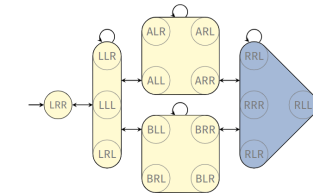
An operator of an FDR operator is a SAS+ operator if

- $\text{pre}(o)$ 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

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.



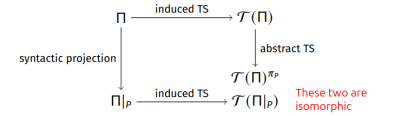
$h^\alpha((p \mapsto L, t_a \mapsto R, t_b \mapsto R)) = 3$

10.1 Additivity

- Orthogonal $\alpha_1 \wedge \alpha_2$: $\forall v(t \in \mathcal{T})(\alpha_1(s) = \alpha_2(t)) \vee (\alpha_2(s) = \alpha_2(t))$, 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 orthogonal 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)
- Syntactic 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.

- $N_i := \Pi_{j=1}^i |\text{dom}(v_j)|$
- PDB-index(s) := $\sum_{i=1}^k N_i \cdot s(v_i)$

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

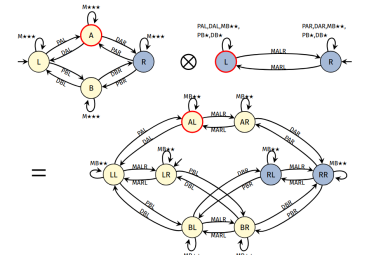
```

1:  procedure MERGE-AND-SHRINK( $\Pi, B$ )
2:     $F \leftarrow F(\Pi)$ 
3:    while  $|F| > 1$  do
4:      "type"  $\leftarrow$  pick-merge-or-shrink( $F$ )
5:      if type = merge then
6:         $\mathcal{T}_1 \leftarrow \text{pick}(F)$ 
7:         $\mathcal{T}_2 \leftarrow \text{pick}(F \setminus \{\mathcal{T}_1\})$ 
8:         $F \leftarrow (F \setminus \{\mathcal{T}_1, \mathcal{T}_2\}) \cup \{\mathcal{T}_1 \otimes \mathcal{T}_2\}$ 
9:      end
10:     if type = shrink then
11:        $\mathcal{T} \leftarrow \text{pick}(F)$ 
12:        $\beta \leftarrow \text{pick-abstraction}(B)$ 
13:        $F \leftarrow (F \setminus \{\mathcal{T}\}) \cup \{\beta(\mathcal{T})\}$ 
14:     end
15:   end
16:   return  $F[0]$ 
17: end

```

$\mathcal{T}^{\pi(\text{package})} \otimes \mathcal{T}^{\pi(\text{truck A})}$

$S_\otimes = S_1 \times S_2$



11.1 Factored Transition System (FTS or F)

- A finite set $F = \{\mathcal{T}_1, \dots, \mathcal{T}_n\}$, where all share ℓ and cost(s).
- FTS induced by Π is $F(\Pi) = \{\mathcal{T}^{\pi_*} \mid v \in V\}$
- $\otimes F \sim \mathcal{T}(\Pi)$ 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 imprecise.

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...

$list_0 = \{(0,0)\}$	$list_0 = \{(0,0)\}$	$list_0 = \{(0,0)\}$	$list_0 = \{(0,0)\}$
$list_1 = \{(0,1)\}$	$list_1 = \{(0,1)\}$	$list_1 = \{(0,1)\}$	$list_1 = \{(0,1)\}$
$list_2 = \{(1,0)\}$	$list_2 = \{(1,0), (1,1)\}$	$list_2 = \{(1,0), (1,1)\}$	$list_2 = \{(1,0), (1,1)\}$
$list_3 = \{(1,1)\}$	$list_3 = \emptyset$	$list_3 = \{(2,0), (2,1)\}$	$list_3 = \{(2,0), (2,1)\}$
$list_4 = \{(2,0)\}$	$list_4 = \{(2,0)\}$	$list_4 = \{(2,0), (2,1)\}$	$list_4 = \{(2,0), (2,1)\}$
$list_5 = \{(2,1)\}$	$list_5 = \{(2,1)\}$	$list_5 = \{(2,0), (2,1)\}$	$list_5 = \{(2,0), (2,1)\}$
$list_6 = \{(3,0)\}$	$list_6 = \{(3,0)\}$	$list_6 = \{(3,0), (3,1)\}$	$list_6 = \{(3,0), (3,1)\}$
$list_7 = \{(3,1)\}$	$list_7 = \{(3,1)\}$	$list_7 = \{(3,0), (3,1)\}$	$list_7 = \{(3,0), (3,1)\}$

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

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

$T_{m\&s}^1$	$s_2 = 0$	$s_2 = 1$	$T_{m\&s}^2$	$s_2 = 0$	$s_2 = 1$
$s_1 = 0$	0	1	$s_1 = 0$	1	1
$s_1 = 1$	2	2	$s_1 = 1$	1	0
$s_1 = 2$	3	3	$s_1 = 2$	2	2
$s_1 = 3$	3	3	$s_1 = 3$	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 $\langle \lambda : L \rightarrow L', c : L \rightarrow \mathbb{R}^+ \rangle$, such that $\ell \in L', c'(\lambda(\ell)) \leq c(\ell)$
- The label-reduced transition system $\mathcal{T}^{\langle \lambda, c \rangle} = \langle S, L', c', \{ \langle s, \lambda(\ell), t \rangle \mid \langle s, \ell, t \rangle \in T \}, s_0, S_e \rangle$

11.5.2 Properties

- ℓ is alive in F if all $T' \in F$ 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 equivalent if ℓ locally subsumes ℓ' and vice versa.
- ℓ and ℓ' are \mathcal{T} -combinable if there are locally equivalent in all transition systems $T' \in F \setminus \{\mathcal{T}\}$

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 $\gamma \models \lambda$ OR if for all plans $\langle o_1, \dots, o_n \rangle$ at least one $\text{pre}(o_i) \models \lambda$

- Causal fact landmark: Same as above, but $\lambda := v$

- To calculate the RTG landmarks, first instantiate all nodes with all potential landmarks. Then perform this on RTG

$$LM(n) = \{n\} \cap_{n' \rightarrow n \in A} LM(n') \quad \text{if type}(n) = \vee$$

$$LM(n) = \{n\} \cup_{n' \rightarrow n \in A} LM(n') \quad \text{if type}(n) = \wedge$$

