Automated Planning

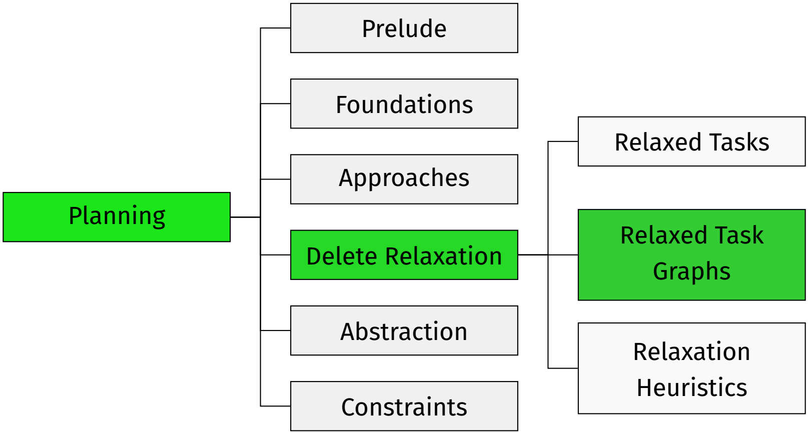
D3. Delete Relaxation: Relaxed Task Graphs

Jendrik Seipp

Linköping University

based on slides from the AI group at the University of Basel

Content of this Course



Relaxed Task Graphs

Relaxed Task Graphs

Let Π^+ be a relaxed planning task.

The relaxed task graph of Π^+, in symbols $RTG(\Pi^+)$, is an AND/OR graph that encodes

- which state variables can become true in an applicable operator sequence for Π^+,
- which operators of Π^+ can be included in an applicable operator sequence for Π^+,
- if the goal of Π^+ can be reached,
- and how these things can be achieved.

We present its definition in stages.

Note: Throughout this chapter, we assume flat operators.

Running Example

As a running example, consider the relaxed planning task $\langle V, I, \{o_1, o_2, o_3, o_4\}, \gamma\rangle$ with

$$\begin{aligned}
V &= \{a, b, c, d, e, f, g, h\} \\
I &= \{a \mapsto \mathbf{T}, b \mapsto \mathbf{T}, c \mapsto \mathbf{F}, d \mapsto \mathbf{T}, \\
&\qquad e \mapsto \mathbf{F}, f \mapsto \mathbf{F}, g \mapsto \mathbf{F}, h \mapsto \mathbf{F}\} \\
o_1 &= \langle c \vee (a \wedge b), c \wedge ((c \wedge d) \rhd e), 1\rangle \\
o_2 &= \langle \top, f, 2\rangle \\
o_3 &= \langle f, g, 1\rangle \\
o_4 &= \langle f, h, 1\rangle \\
\gamma &= e \wedge (g \wedge h)
\end{aligned}$$

Construction

Components of Relaxed Task Graphs

A relaxed task graph is an AND/OR graph:

- Nodes can be forced true and false
- If n is an AND node and all of its predecessors are forced true, then n is forced true.
- If n is an OR node and at least one of its predecessors is forced true, then n is forced true.

A relaxed task graph has four kinds of components:

- Variable nodes represent the state variables.
- The initial node represents the initial state.
- Operator subgraphs represent the preconditions and effects of operators.
- The goal subgraph represents the goal.

Variable Nodes

Let $\Pi^+ = \langle V, I, O^+, \gamma \rangle$ be a relaxed planning task.

- For each $v \in V$, $RTG(\Pi^+)$ contains an OR node n_v.
  These nodes are called variable nodes.

Variable Nodes: Example

$$V = \{a, b, c, d, e, f, g, h\}$$

Initial Node

Let $\Pi^+ = \langle V, I, O^+, \gamma \rangle$ be a relaxed planning task.

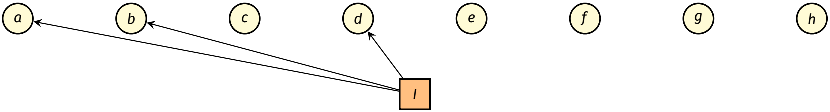
- $RTG(\Pi^+)$ contains an AND node n_I.
  This node is called the initial node.
- For all $v \in V$ with $I(v) = \mathbf{T}$, $RTG(\Pi^+)$ has an arc from n_I to n_v. These arcs are called initial state arcs.
- The initial node has no predecessor nodes.

Initial Node and Initial State Arcs: Example

$$V = \{a, b, c, d, e, f, g, h\}$$

Initial Node and Initial State Arcs: Example

$$I = \{a \mapsto \mathbf{T}, b \mapsto \mathbf{T}, c \mapsto \mathbf{F}, d \mapsto \mathbf{T}, e \mapsto \mathbf{F}, f \mapsto \mathbf{F}, g \mapsto \mathbf{F}, h \mapsto \mathbf{F}\}$$



Operator Subgraphs

Let $\Pi^+ = \langle V, I, O^+, \gamma \rangle$ be a relaxed planning task.

For each operator $o^+ \in O^+$, $RTG(\Pi^+)$ contains an operator subgraph with the following parts:

- for each formula φ that occurs as a subformula of the precondition or of some effect condition of o^+, a formula node n_φ (details follow)
- for each conditional effect $(\chi \rhd v)$ that occurs in the effect of o^+, an effect node $n^\chi_{o^+}$ (details follow); unconditional effects are treated as $(\top \rhd v)$

Formula Nodes

Formula nodes n_φ are defined as follows:

- If $\varphi = v$ for some state variable v, n_φ is the variable node n_v (so no new node is introduced).
- If $\varphi = \top$, n_φ is an AND node without incoming arcs.
- If $\varphi = \bot$, n_φ is an OR node without incoming arcs.
- If $\varphi = (\varphi_1 \land \varphi_2)$, n_φ is an AND node with incoming arcs from n_{φ_1} and n_{φ_2}.
- If $\varphi = (\varphi_1 \lor \varphi_2)$, n_φ is an OR node with incoming arcs from n_{φ_1} and n_{φ_2}.

Note: identically named nodes are identical,
so if the same formula occurs multiple times in the task,
the same node is reused.

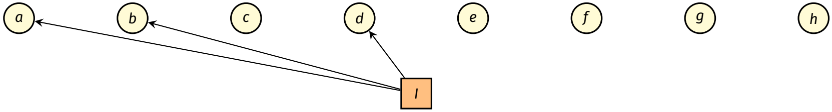
Effect Nodes

Effect nodes $n^{\chi}_{o^+}$ are defined as follows:

- $n^{\chi}_{o^+}$ is an AND node
- It has an incoming arc from the formula nodes $n_{pre(o^+)}$ (precondition arcs) and n_{χ} (effect condition arcs).
- Exception: if $\chi = \top$,
  1) there is no effect condition arc, and
  2) we omit the precondition arc if there are other incoming arcs.
  (This makes our pictures cleaner.)
- For every conditional effect $(\chi \rhd v)$ in the operator, there is an arc from $n^{\chi}_{o^+}$ (effect arcs) to variable node n_v.

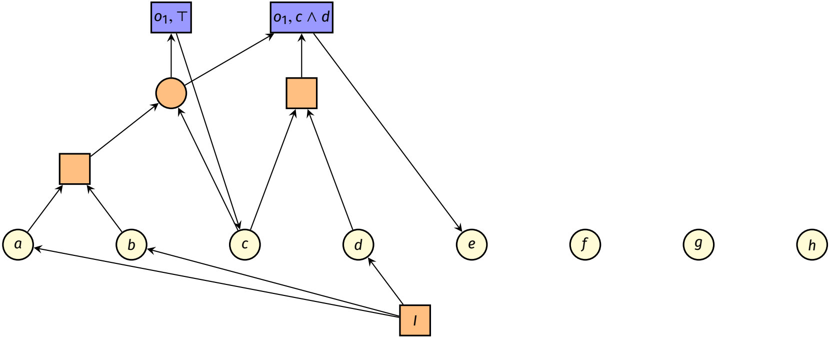
Note: identically named nodes are identical, so if the same effect condition occurs multiple times in the same operator, this only induces one node.

Operator Subgraphs: Example



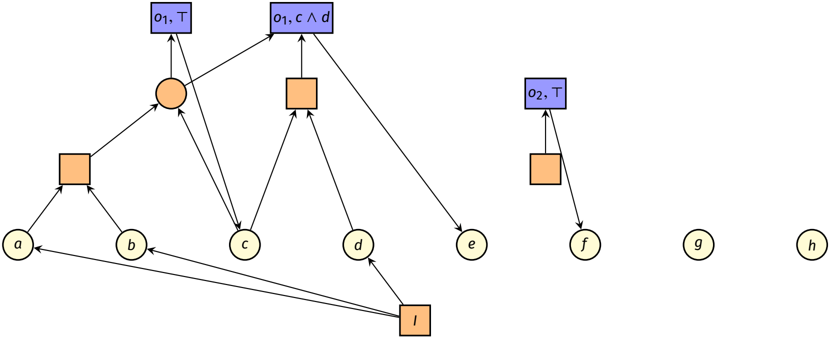
Operator Subgraphs: Example

$$o_1 = \langle c \vee (a \wedge b), c \wedge ((c \wedge d) \rhd e), 1 \rangle$$



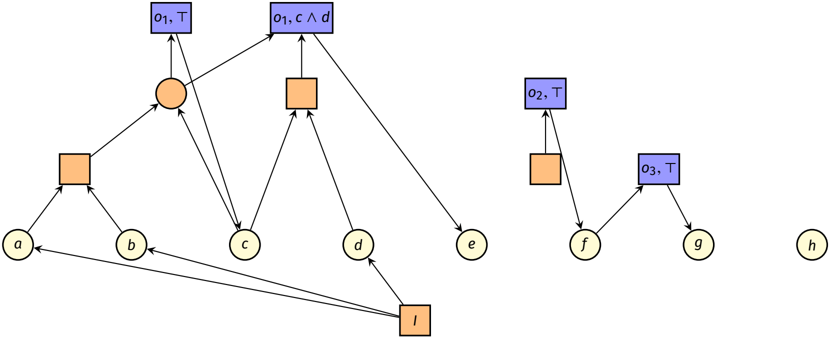
Operator Subgraphs: Example

$$o_2 = \langle \top, f, 2 \rangle$$



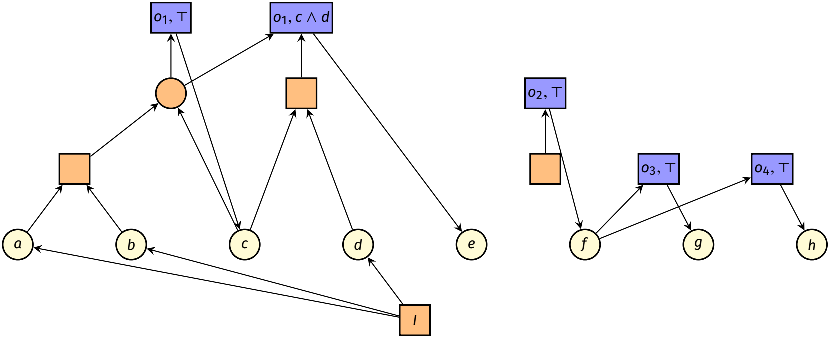
Operator Subgraphs: Example

$$o_3 = \langle f, g, 1 \rangle$$



Operator Subgraphs: Example

$$o_4 = \langle f, h, 1 \rangle$$

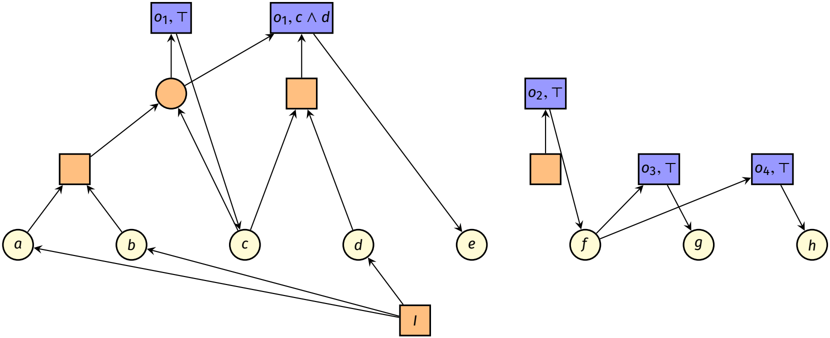


Goal Subgraph

Let $\Pi^+ = \langle V, I, O^+, \gamma \rangle$ be a relaxed planning task.

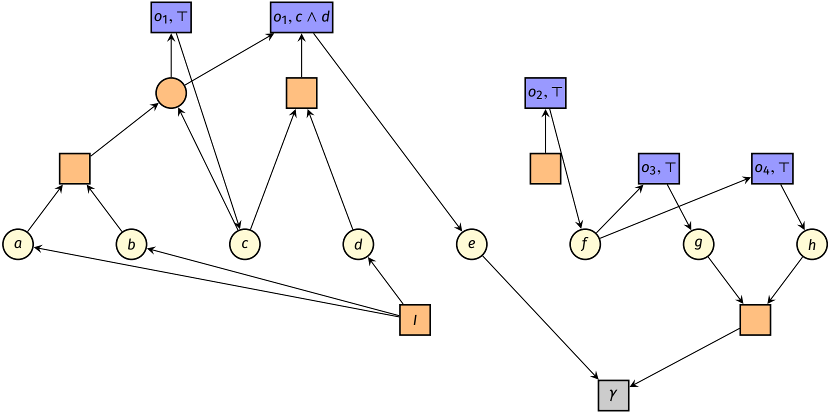
$RTG(\Pi^+)$ contains a goal subgraph, consisting of formula nodes for the goal γ and its subformulas, constructed in the same way as formula nodes for preconditions and effect conditions.

Goal Subgraph and Final Relaxed Task Graph: Example



Goal Subgraph and Final Relaxed Task Graph: Example

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



Reachability Analysis

How Can We Use Relaxed Task Graphs?

- We are now done with the definition of relaxed task graphs.
- Now we want to use them to derive information about planning tasks.
- In the following chapter, we will use them to compute heuristics for delete-relaxed planning tasks.
- Here, we start with something simpler: reachability analysis.
- Since the initial node is an AND node without predecessors, it is forced true.

Algorithm for Reachability Analysis

- reachability analysis in RTGs = computing all forced true nodes
- Here is an algorithm that achieves this:

Reachability Analysis

Associate a *reachable* attribute with each node.

for all nodes n:

    $n.reachable := \textbf{false}$

while no fixed point is reached:

    Choose a node n.

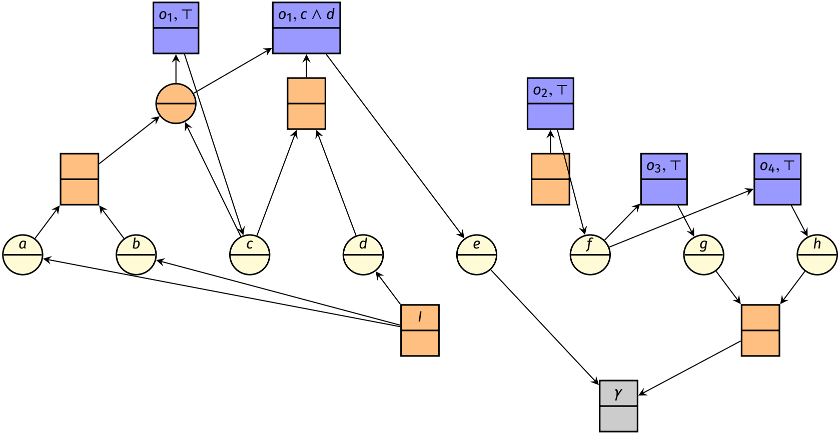
    if n is an AND node:

        $n.reachable := \bigwedge_{n' \in predecessors(n)} n'.reachable$

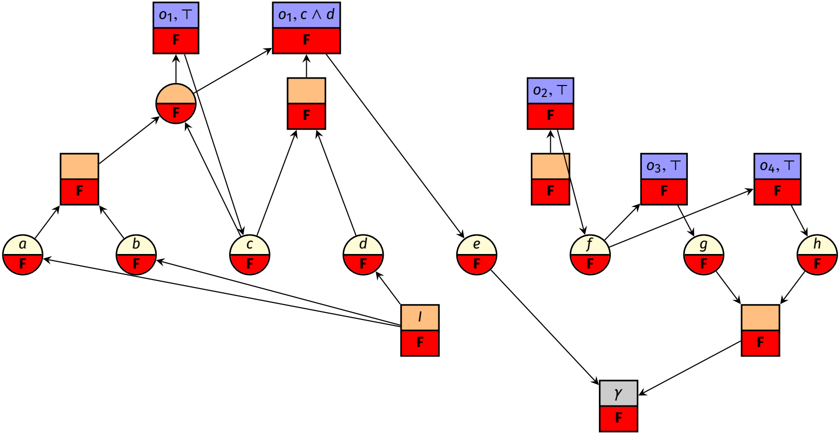
    if n is an OR node:

        $n.reachable := \bigvee_{n' \in predecessors(n)} n'.reachable$

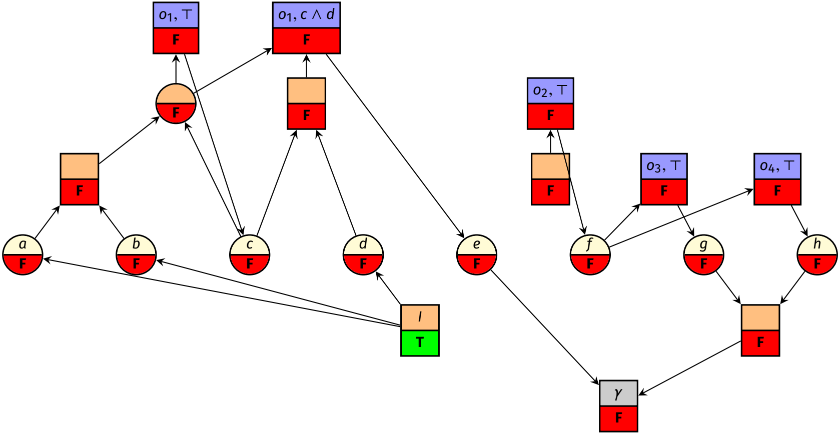
Reachability Analysis: Example



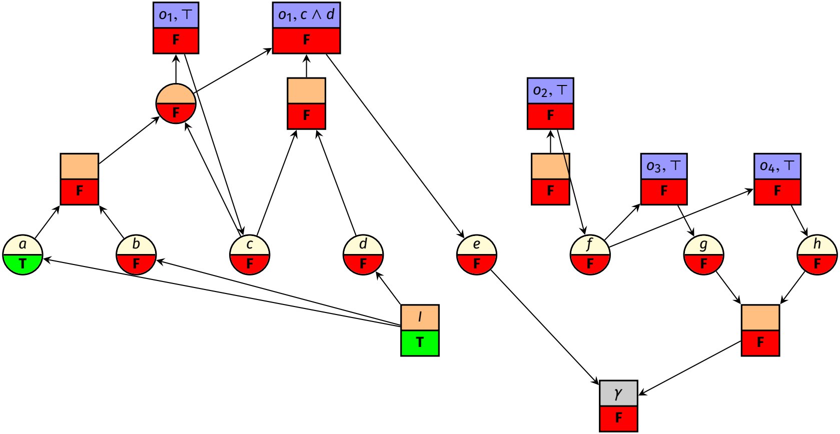
Reachability Analysis: Example



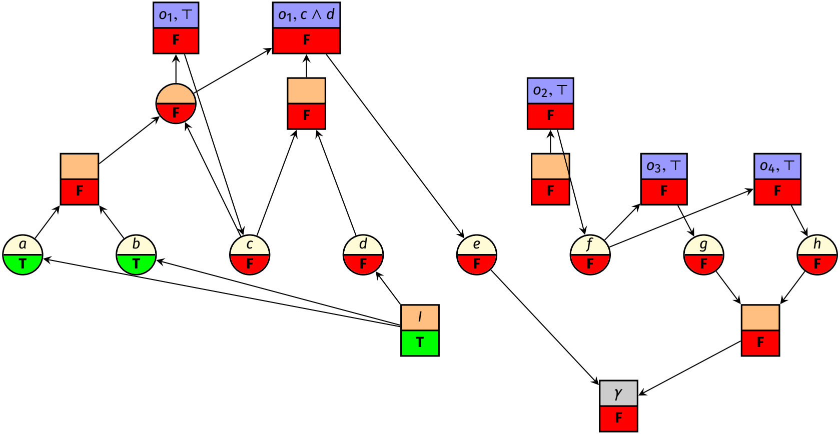
Reachability Analysis: Example



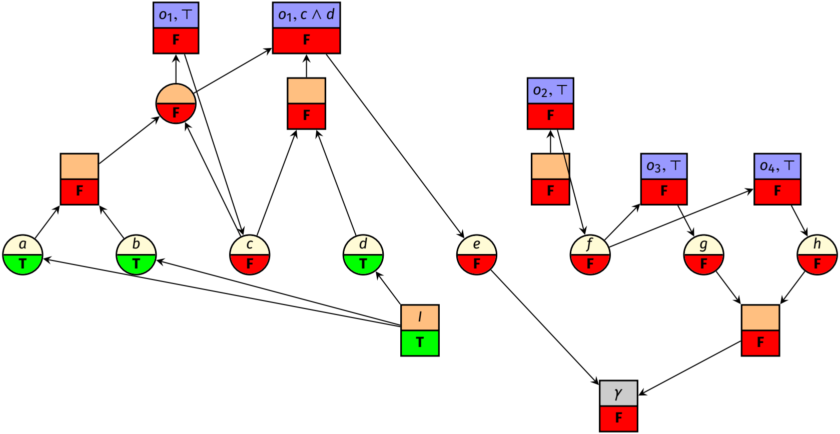
Reachability Analysis: Example



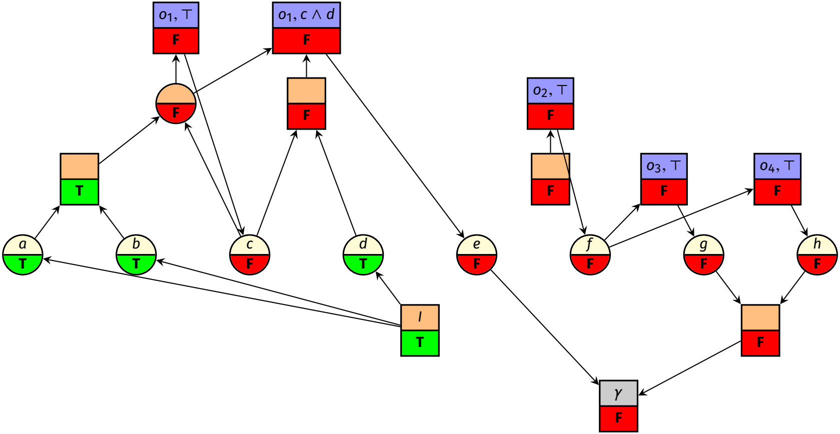
Reachability Analysis: Example



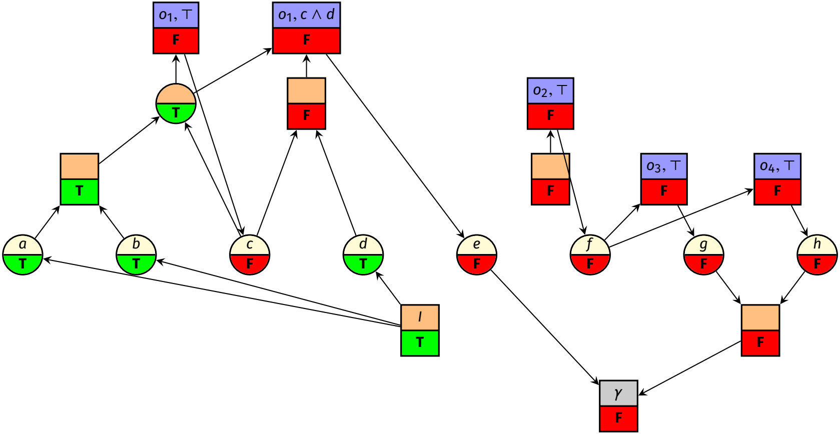
Reachability Analysis: Example



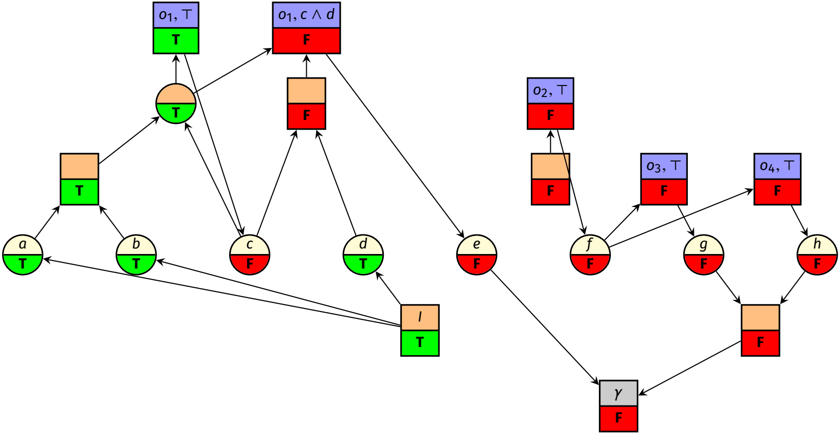
Reachability Analysis: Example



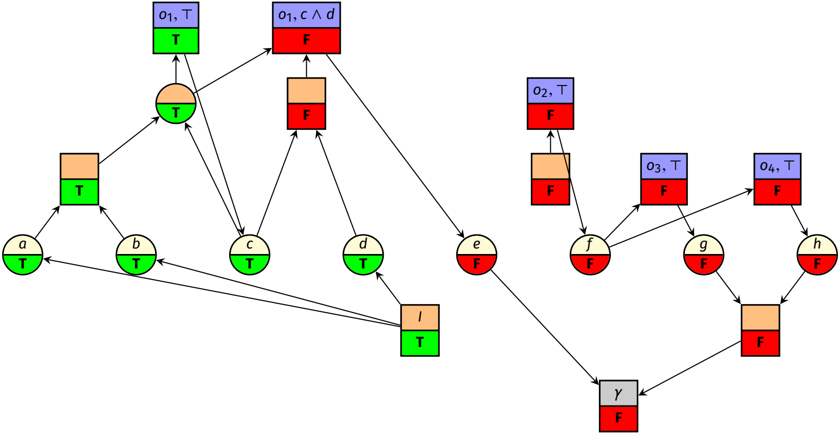
Reachability Analysis: Example



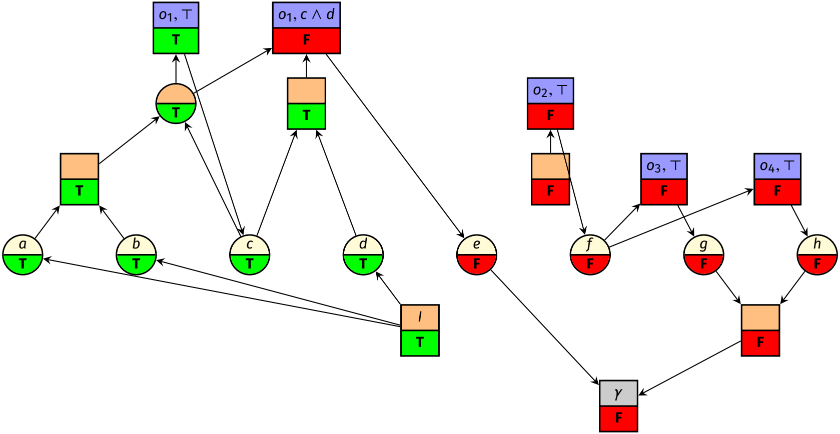
Reachability Analysis: Example



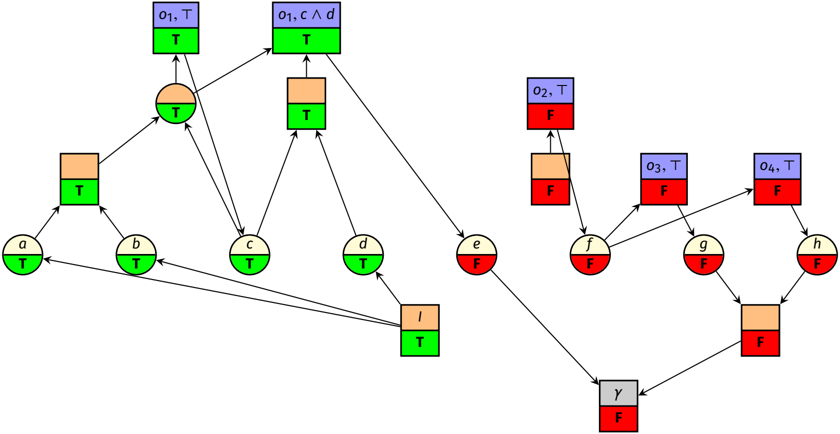
Reachability Analysis: Example



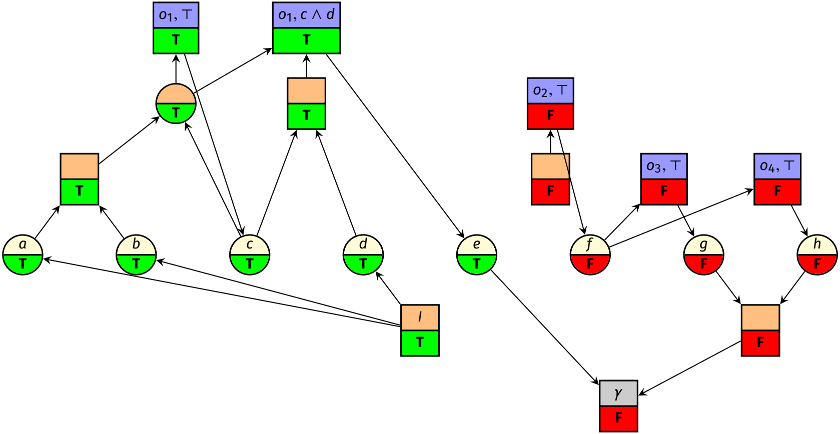
Reachability Analysis: Example



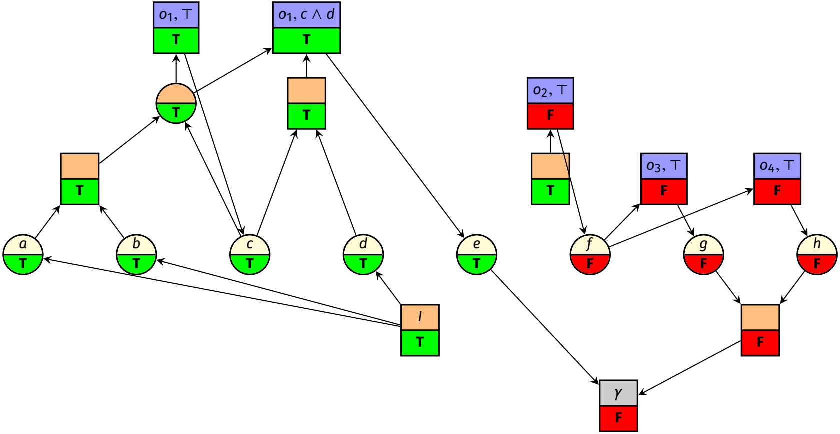
Reachability Analysis: Example



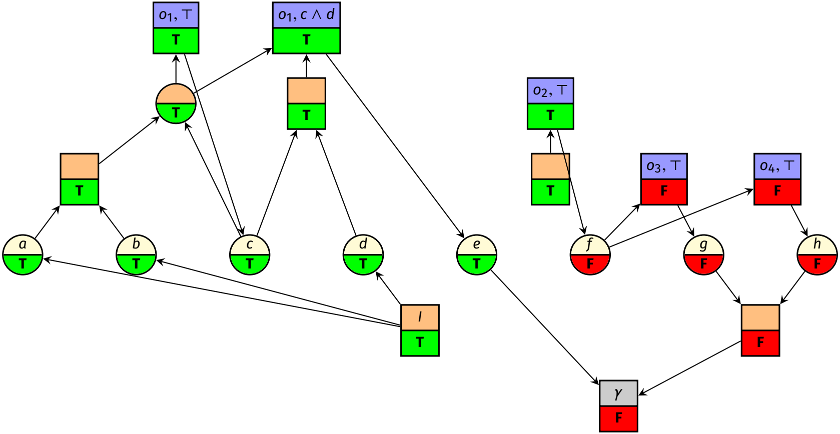
Reachability Analysis: Example



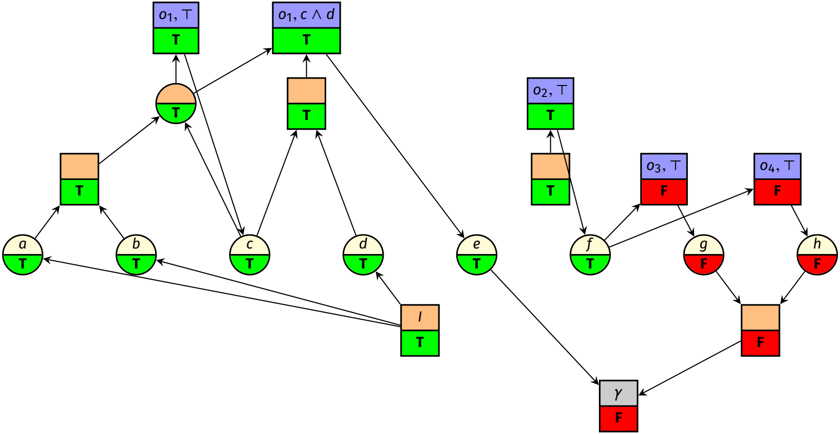
Reachability Analysis: Example



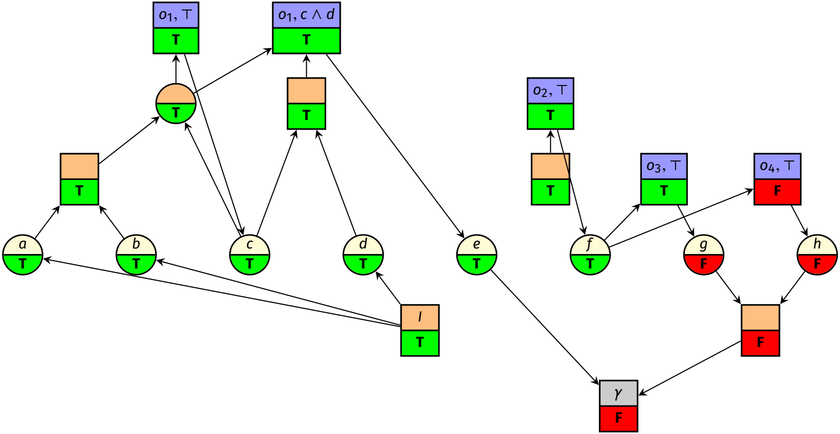
Reachability Analysis: Example



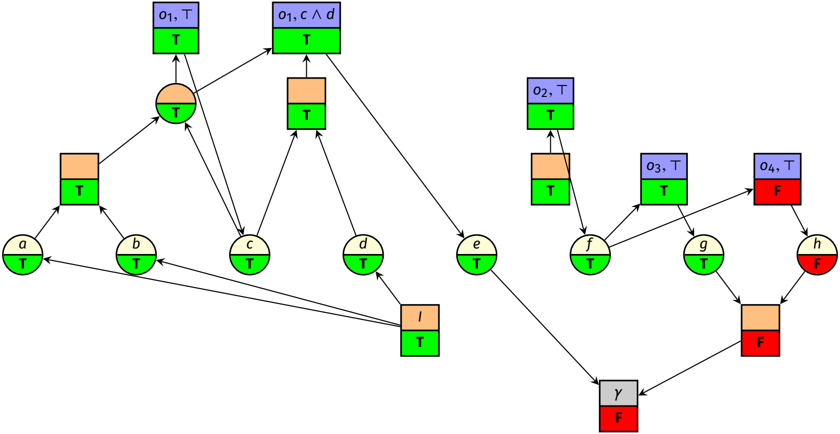
Reachability Analysis: Example



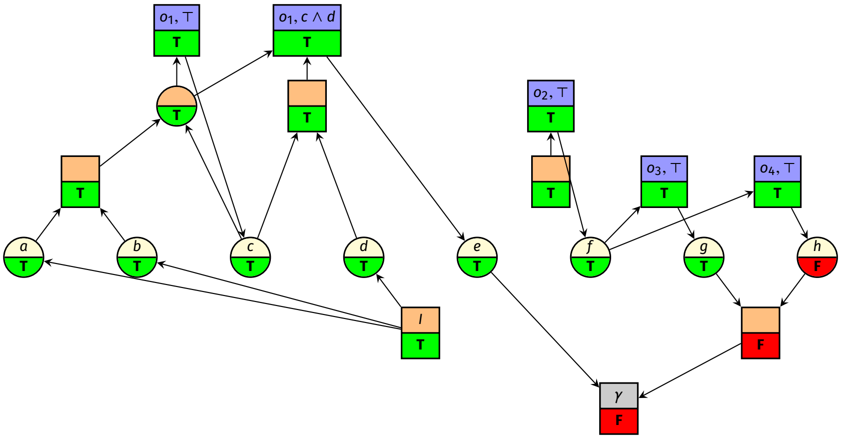
Reachability Analysis: Example



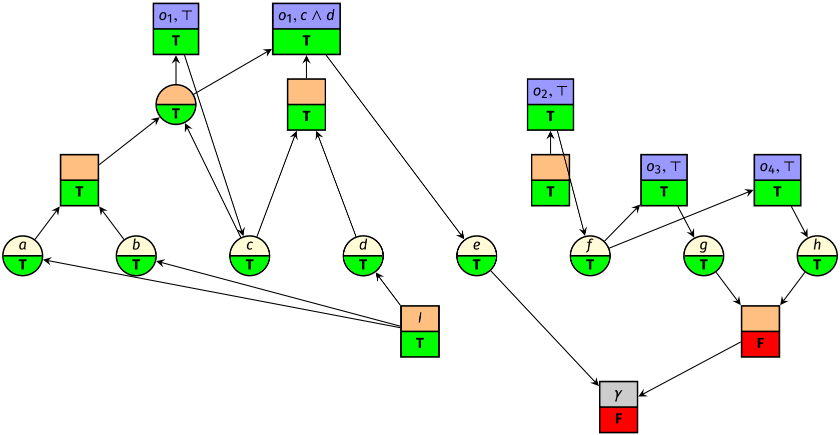
Reachability Analysis: Example



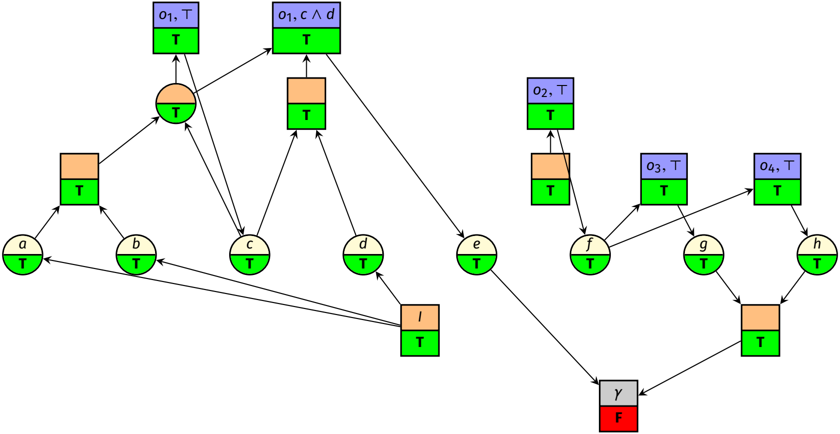
Reachability Analysis: Example



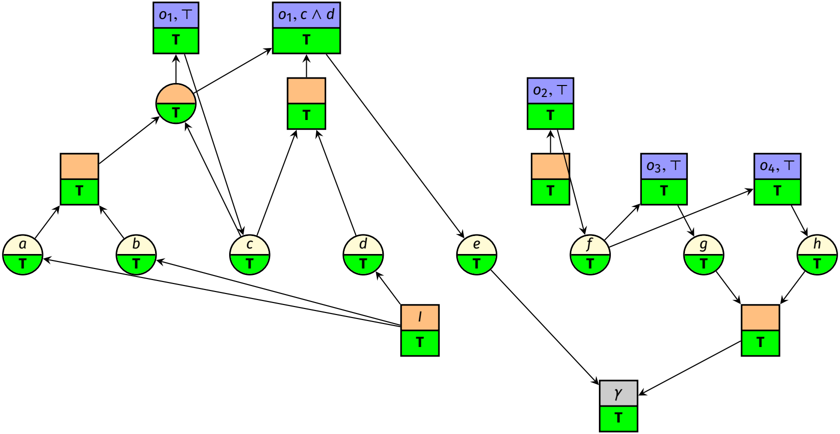
Reachability Analysis: Example



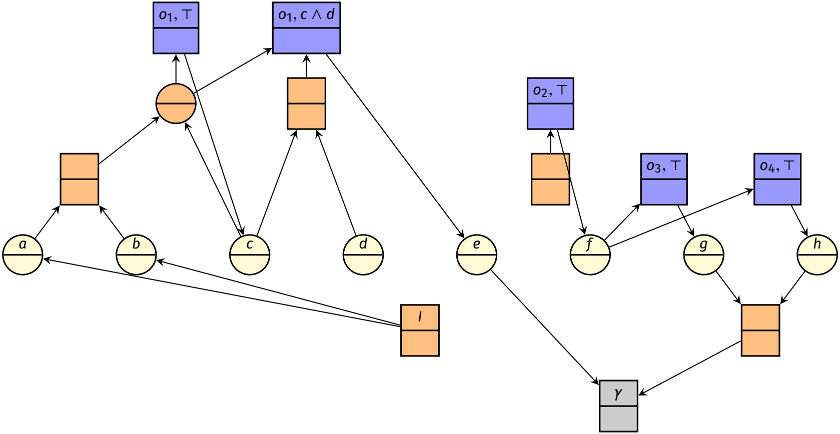
Reachability Analysis: Example



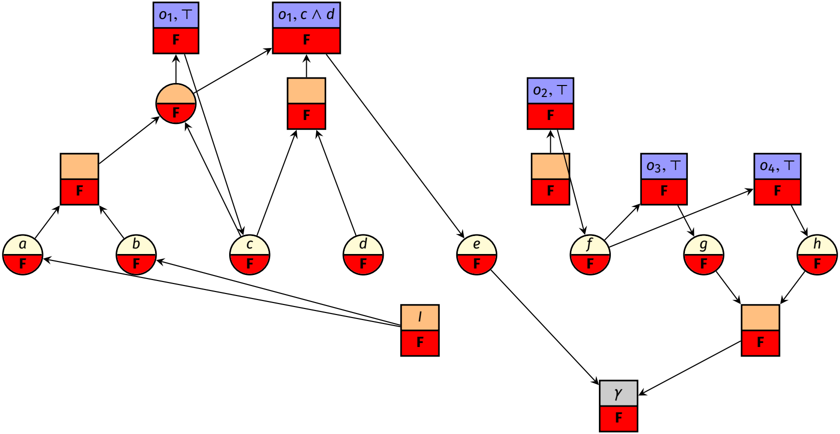
Reachability Analysis: Example



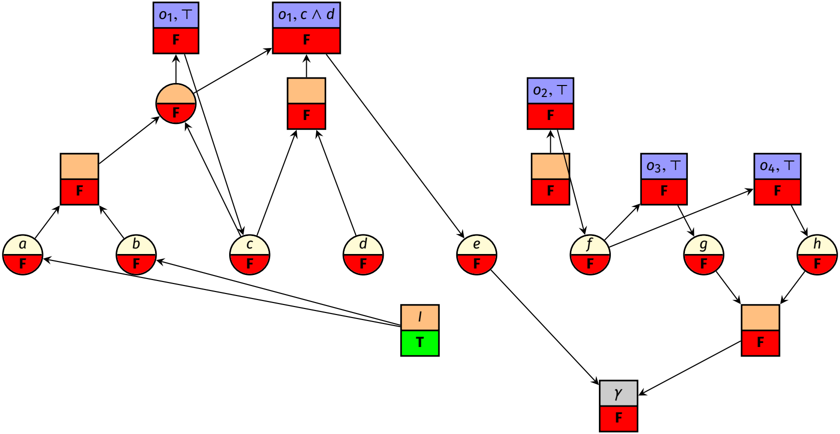
Reachability Analysis: Example with Different Initial State



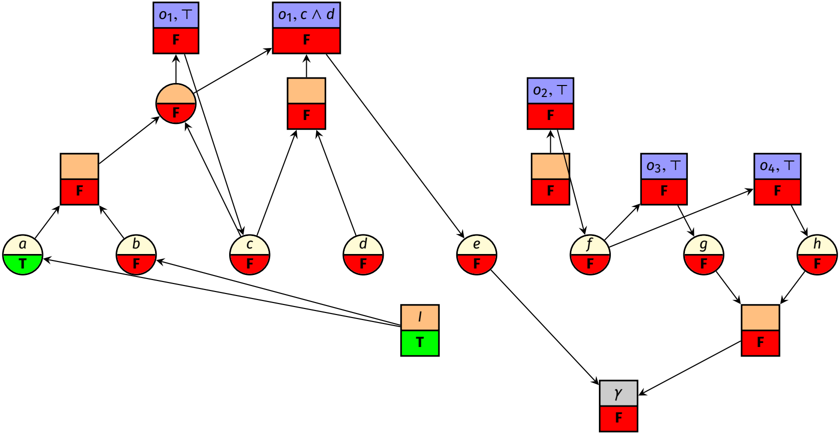
Reachability Analysis: Example with Different Initial State



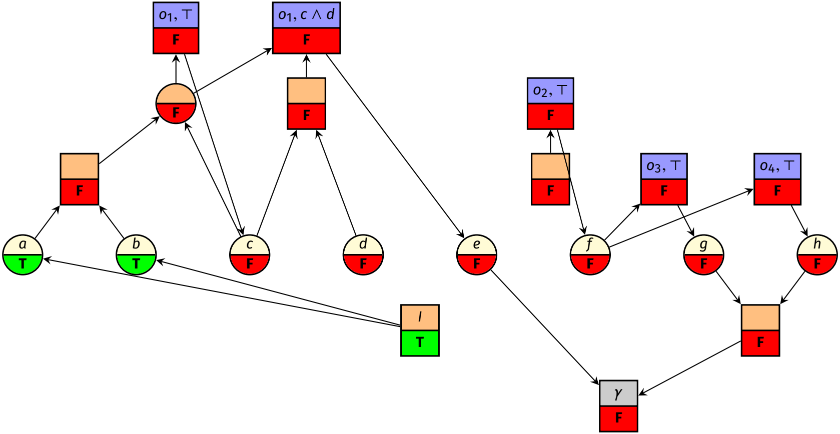
Reachability Analysis: Example with Different Initial State



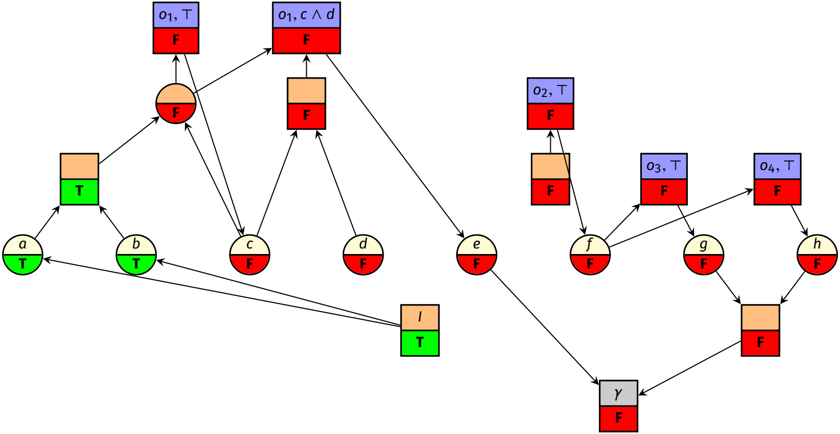
Reachability Analysis: Example with Different Initial State



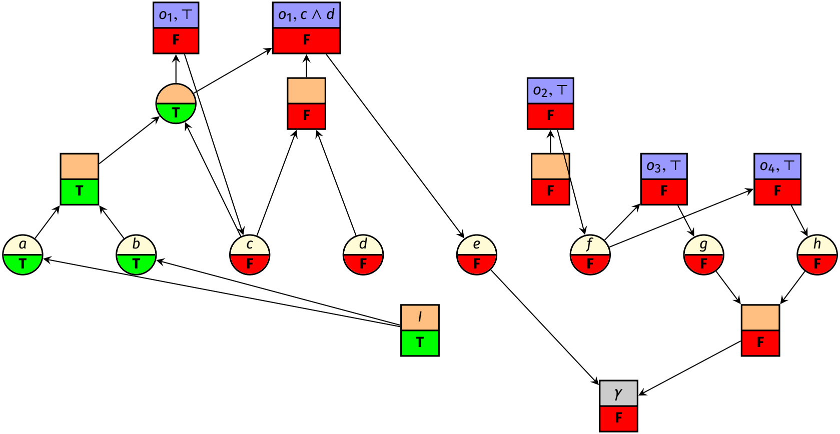
Reachability Analysis: Example with Different Initial State



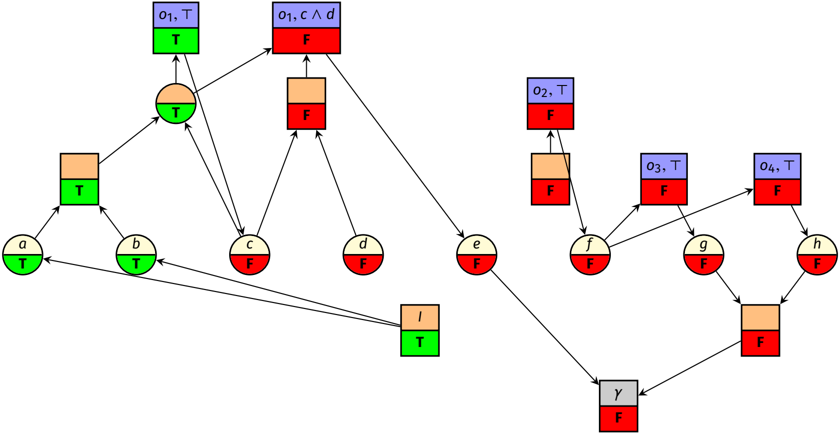
Reachability Analysis: Example with Different Initial State



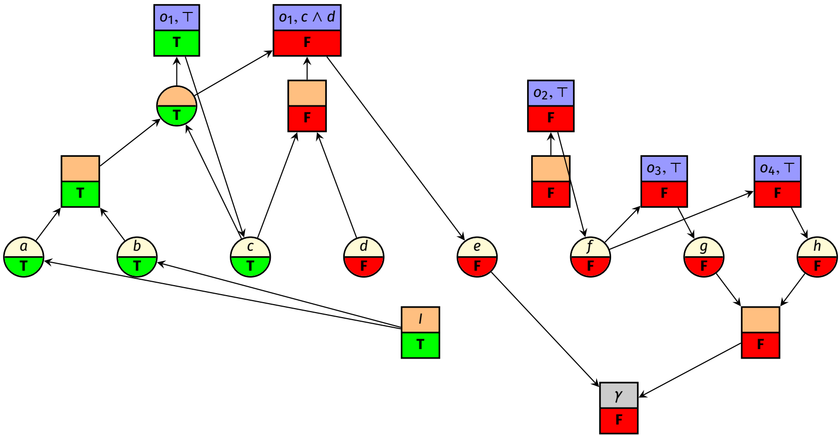
Reachability Analysis: Example with Different Initial State



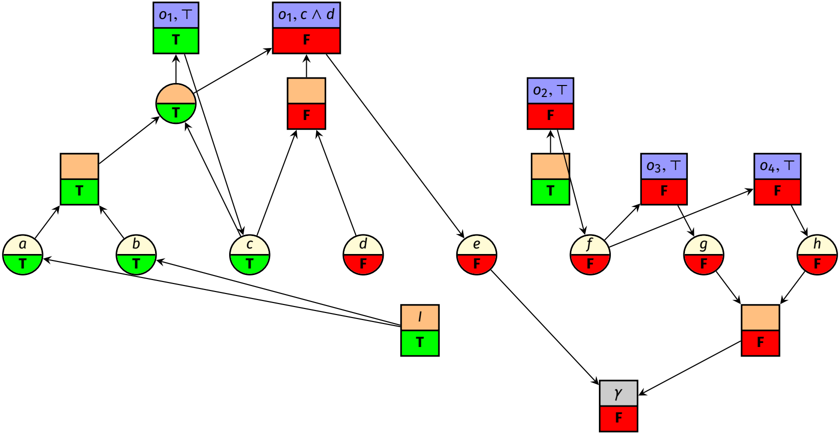
Reachability Analysis: Example with Different Initial State



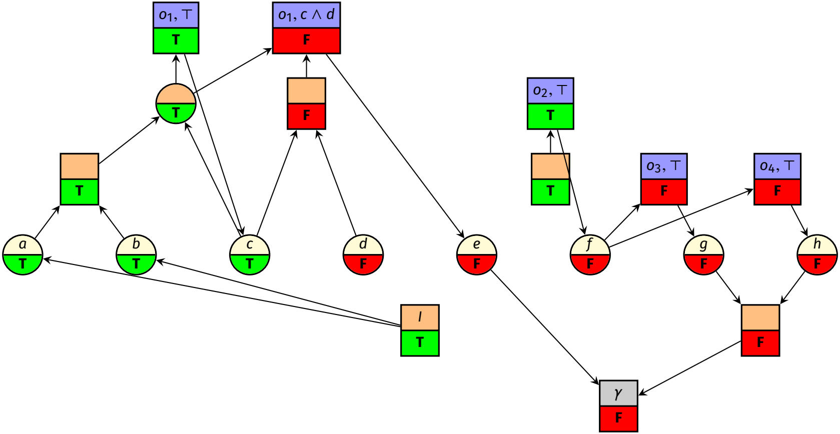
Reachability Analysis: Example with Different Initial State



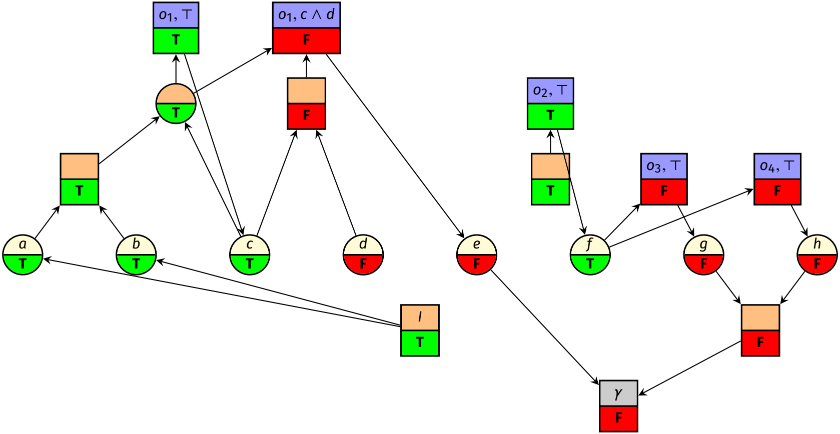
Reachability Analysis: Example with Different Initial State



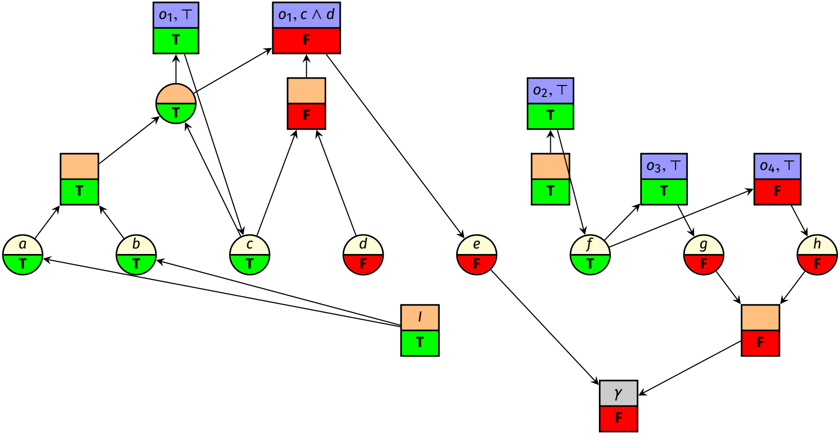
Reachability Analysis: Example with Different Initial State



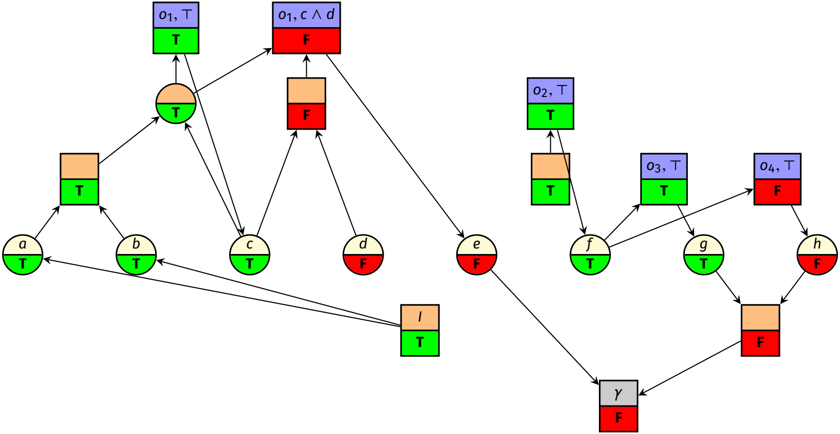
Reachability Analysis: Example with Different Initial State



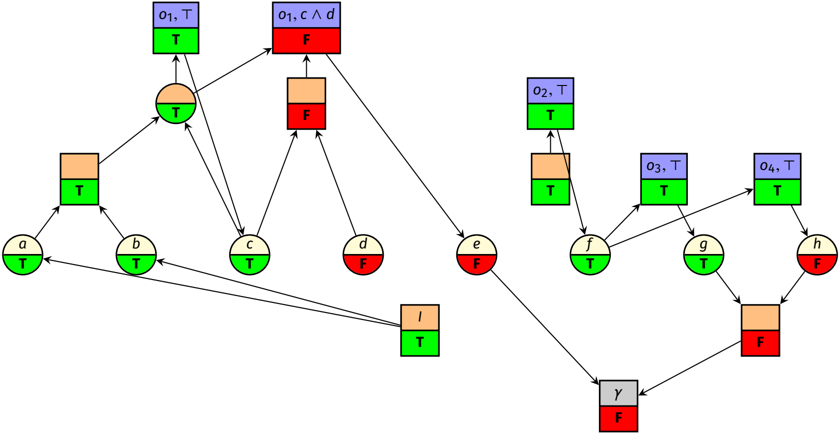
Reachability Analysis: Example with Different Initial State



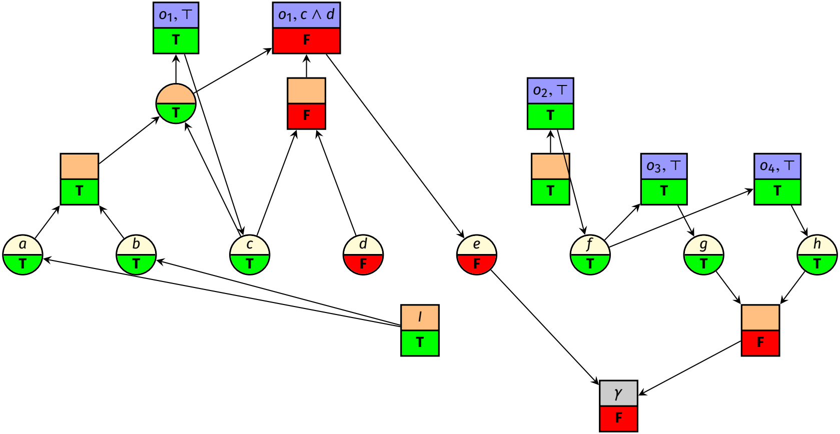
Reachability Analysis: Example with Different Initial State



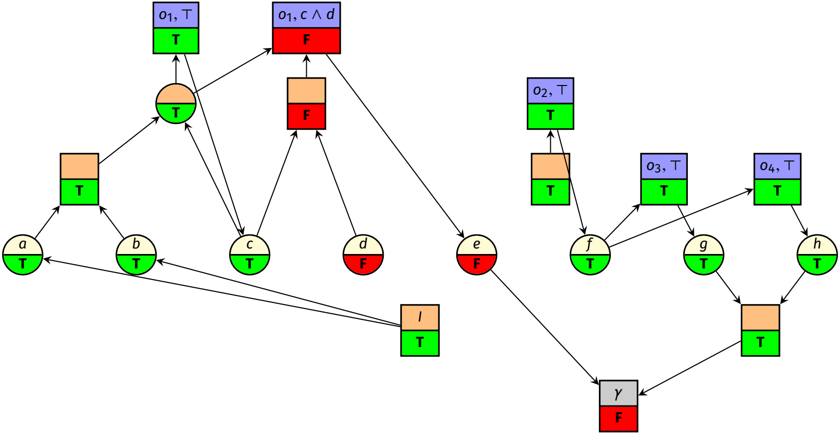
Reachability Analysis: Example with Different Initial State



Reachability Analysis: Example with Different Initial State



Reachability Analysis: Example with Different Initial State



Remarks

Relaxed Task Graphs in the Literature

Some remarks on the planning literature:

- Usually, only the STRIPS case is studied.

⇝ definitions simpler: only variable nodes and operator nodes, no formula nodes or effect nodes

- Usually, so-called relaxed planning graphs (RPGs) are studied instead of RTGs.
- These are temporally unrolled versions of RTGs, i.e., they have multiple layers ("time steps") and are acyclic.

⇝ TDDC17 course

Summary

Summary

- Relaxed task graphs (RTGs) represent (most of) the information of a relaxed planning task as an AND/OR graph.
- They consist of:
  - variable nodes
  - an initial node
  - operator subgraphs including formula nodes and effect nodes
  - a goal subgraph including formula nodes
- RTGs can be used to analyze reachability in relaxed tasks

Questions?

post feedback and ask questions anonymously at

`https://padlet.com/jendrikseipp/tddd48`