

Planning and Optimization

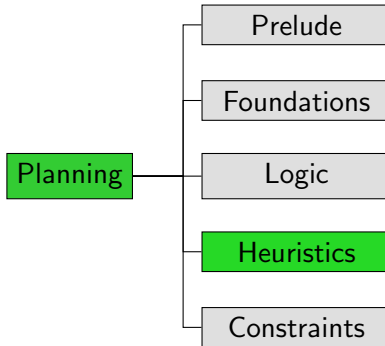
D4. Delete Relaxation: AND/OR Graphs

Malte Helmert and Gabriele Röger

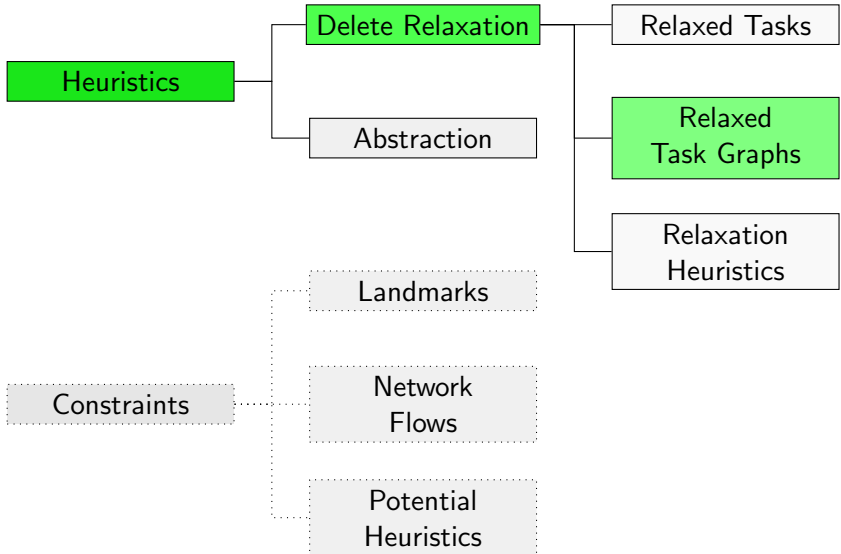
Universität Basel

October 24, 2022

Content of this Course



Content of this Course: Heuristics



AND/OR Graphs

Using Relaxations in Practice

How can we use relaxations for heuristic planning in practice?

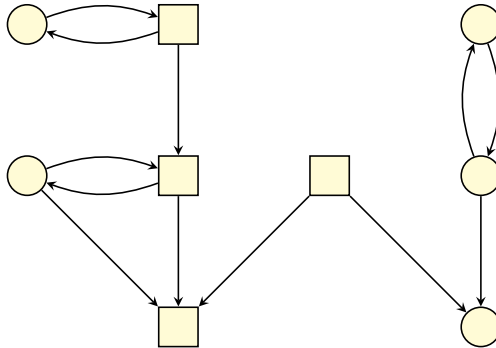
Different possibilities:

- Implement an **optimal planner** for relaxed planning tasks and use its solution costs as estimates, even though optimal relaxed planning is NP-hard.
 \rightsquigarrow **h^+ heuristic**
- Do not actually solve the relaxed planning task, but compute an approximation of its solution cost.
 \rightsquigarrow **h^{\max} heuristic, h^{add} heuristic, $h^{\text{LM-cut}}$ heuristic**
- Compute a solution for relaxed planning tasks which is not necessarily optimal, but “reasonable”.
 \rightsquigarrow **h^{FF} heuristic**

AND/OR Graphs: Motivation

- Most relaxation heuristics we will consider can be understood in terms of computations on graphical structures called **AND/OR graphs**.
- We now introduce AND/OR graphs and study some of their major properties.
- In the next chapter, we will relate AND/OR graphs to relaxed planning tasks.

AND/OR Graph Example



AND/OR Graphs

Definition (AND/OR Graph)

An **AND/OR graph** $\langle N, A, type \rangle$ is a directed graph $\langle N, A \rangle$ with a node label function $type : N \rightarrow \{\wedge, \vee\}$ partitioning nodes into

- **AND nodes** ($type(v) = \wedge$) and
- **OR nodes** ($type(v) = \vee$).

We write $succ(n)$ for the successors of node $n \in N$, i.e.,
 $succ(n) = \{n' \in N \mid \langle n, n' \rangle \in A\}$.

Note: We draw AND nodes as squares and OR nodes as circles.

AND/OR Graph Valuations

Definition (Consistent Valuations of AND/OR Graphs)

Let G be an AND/OR graph with nodes N .

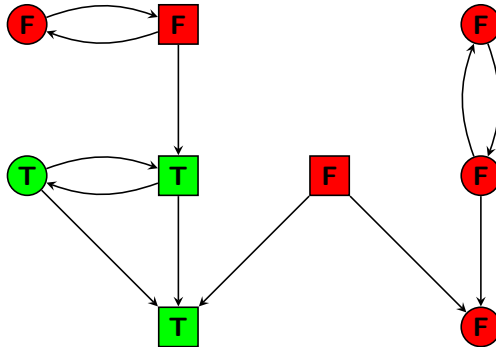
A **valuation** or **truth assignment** of G is a valuation $\alpha : N \rightarrow \{\mathbf{T}, \mathbf{F}\}$, treating the nodes as propositional variables.

We say that α is **consistent** if

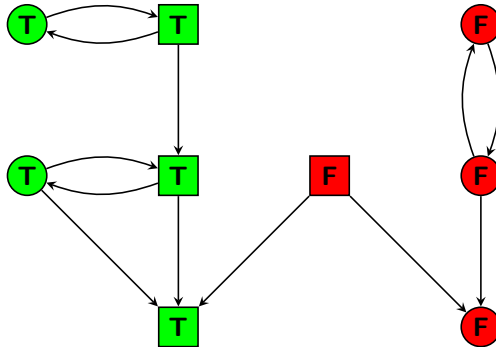
- for all AND nodes $n \in N$: $\alpha \models n$ iff $\alpha \models \bigwedge_{n' \in \text{succ}(n)} n'$.
- for all OR nodes $n \in N$: $\alpha \models n$ iff $\alpha \models \bigvee_{n' \in \text{succ}(n)} n'$.

Note that $\bigwedge_{n' \in \emptyset} n' = \top$ and $\bigvee_{n' \in \emptyset} n' = \perp$.

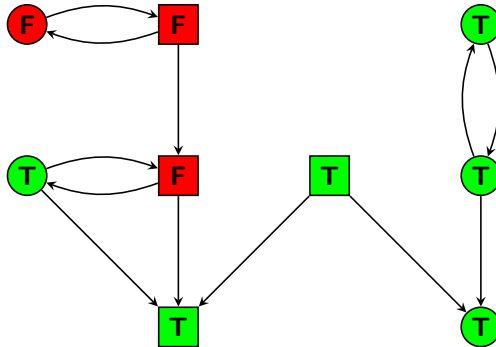
Example: A Consistent Valuation



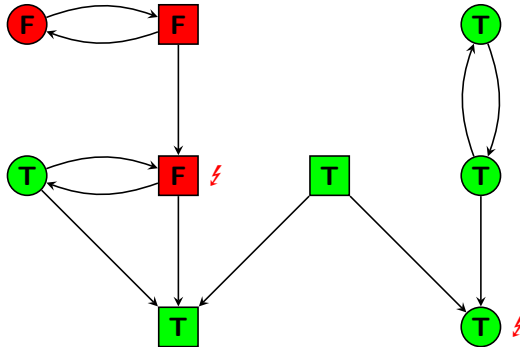
Example: Another Consistent Valuation



Example: An Inconsistent Valuation



Example: An Inconsistent Valuation



How Do We Find Consistent Valuations?

If we want to use valuations of AND/OR graphs algorithmically, a number of questions arise:

- Do consistent valuations **exist** for every AND/OR graph?
- Are they **unique**?
- If not, how are different consistent valuations **related**?
- Can consistent valuations be **computed efficiently**?

Our example shows that the answer to the second question is “no”. In the rest of this chapter, we address the remaining questions.

Forced Nodes

Forced Nodes

Definition (Forced True/False Nodes)

Let G be an AND/OR graph.

A node n of G is called **forced true**
if $\alpha(n) = \mathbf{T}$ for all consistent valuations α of G .

A node n of G is called **forced false**
if $\alpha(n) = \mathbf{F}$ for all consistent valuations α of G .

How can we efficiently determine that nodes are forced true/false?

↪ We begin by looking at some simple rules.

Rules for Forced True Nodes

Proposition (Rules for Forced True Nodes)

Let n be a node in an AND/OR graph.

Rule T-(\wedge): *If n is an AND node and **all** of its successors are forced true, then n is forced true.*

Rule T-(\vee): *If n is an OR node and **at least one** of its successors is forced true, then n is forced true.*

Rules for Forced False Nodes

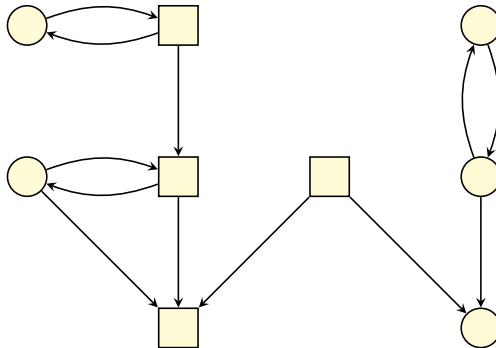
Proposition (Rules for Forced False Nodes)

Let n be a node in an AND/OR graph.

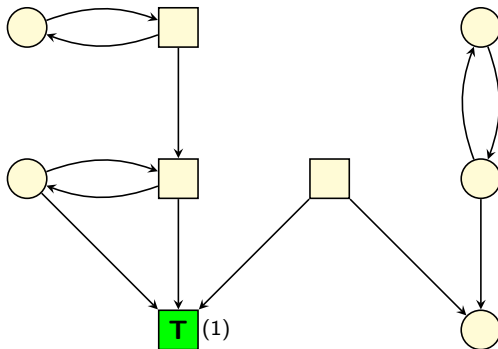
Rule F-(\wedge): *If n is an AND node and **at least one** of its successors is forced false, then n is forced false.*

Rule F-(\vee): *If n is an OR node and **all** of its successors are forced false, then n is forced false.*

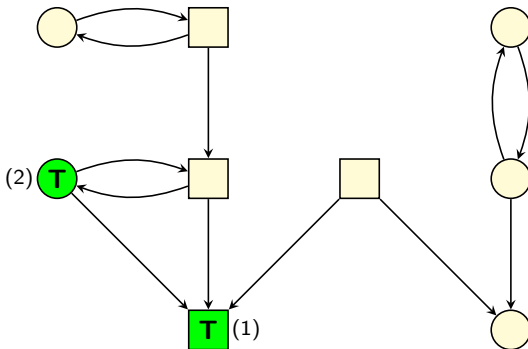
Example: Applying the Rules for Forced Nodes



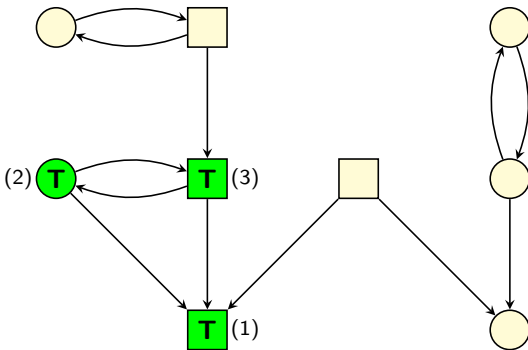
Example: Applying the Rules for Forced Nodes



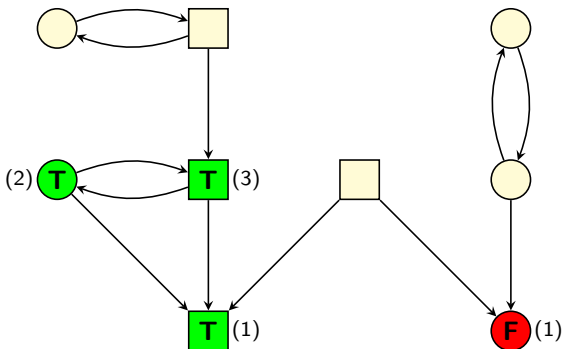
Example: Applying the Rules for Forced Nodes



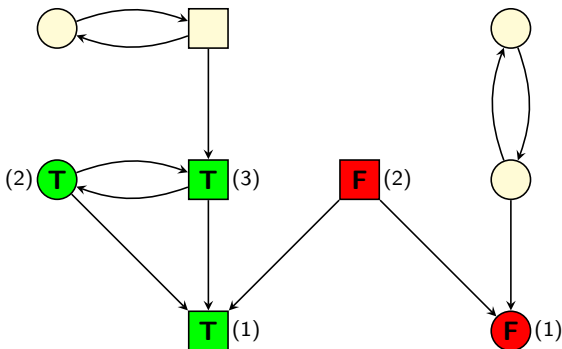
Example: Applying the Rules for Forced Nodes



Example: Applying the Rules for Forced Nodes



Example: Applying the Rules for Forced Nodes



Completeness of Rules for Forced Nodes

Theorem

*If n is a node in an AND/OR graph that is forced true, then this can be derived by a sequence of applications of Rule **T**-(\wedge) and Rule **T**-(\vee).*

Theorem

*If n is a node in an AND/OR graph that is forced false, then this can be derived by a sequence of applications of Rule **F**-(\wedge) and Rule **F**-(\vee).*

We prove the result for **forced true** nodes.

The result for forced false nodes can be proved analogously.

Completeness of Rules for Forced Nodes: Proof (1)

Proof.

- Let α be a valuation where $\alpha(n) = \mathbf{T}$ iff there exists a sequence ρ_n of applications of Rules $\mathbf{T}-(\wedge)$ and Rule $\mathbf{T}-(\vee)$ that derives that n is forced true.
- Because the rules are monotonic, there exists a sequence ρ of rule applications that derives that n is forced true for **all** $n \in on(\alpha)$. (Just concatenate all ρ_n to form ρ .)
- By the correctness of the rules, we know that all nodes reached by ρ are forced true. It remains to show that none of the nodes **not** reached by ρ is forced true.
- We prove this by showing that **α is consistent**, and hence no nodes with $\alpha(n) = \mathbf{F}$ can be forced true.

Completeness of Rules for Forced Nodes: Proof (2)

Proof (continued).

Case 1: nodes n with $\alpha(n) = \mathbf{T}$

- In this case, ρ must have reached n in one of the derivation steps. Consider this derivation step.
- If n is an AND node, ρ must have reached all successors of n in previous steps, and hence $\alpha(n') = \mathbf{T}$ for all successors n' .
- If n is an OR node, ρ must have reached at least one successor of n in a previous step, and hence $\alpha(n') = \mathbf{T}$ for at least one successor n' .
- In both cases, α is consistent for node n .

Completeness of Rules for Forced Nodes: Proof (3)

Proof (continued).

Case 2: nodes n with $\alpha(n) = \mathbf{F}$

- In this case, by definition of α no sequence of derivation steps reaches n . In particular, ρ does not reach n .
- If n is an AND node, there must exist some $n' \in \text{succ}(n)$ which ρ does not reach. Otherwise, ρ could be extended using Rule **T**-(\wedge) to reach n . Hence, $\alpha(n') = \mathbf{F}$ for some $n' \in \text{succ}(n)$.
- If n is an OR node, there cannot exist any $n' \in \text{succ}(n)$ which ρ reaches. Otherwise, ρ could be extended using Rule **T**-(\vee) to reach n . Hence, $\alpha(n') = \mathbf{F}$ for all $n' \in \text{succ}(n)$.
- In both cases, α is consistent for node n .



Remarks on Forced Nodes

Notes:

- The theorem shows that we can compute all forced nodes by applying the rules repeatedly until a fixed point is reached.
- In particular, this also shows that the order of rule application does not matter: we always end up with the same result.
- In an efficient implementation, the sets of forced nodes can be computed in linear time in the size of the AND/OR graph.
- The proof of the theorem also shows that every AND/OR graph has a consistent valuation, as we explicitly construct one in the proof.

Most/Least Conservative Valuations

Most and Least Conservative Valuation

Definition (Most and Least Conservative Valuation)

Let G be an AND/OR graph with nodes N .

The **most conservative valuation** $\alpha_{\text{mcv}}^G : N \rightarrow \{\mathbf{T}, \mathbf{F}\}$ and the **least conservative valuation** $\alpha_{\text{lcv}}^G : N \rightarrow \{\mathbf{T}, \mathbf{F}\}$ of G are defined as:

$$\alpha_{\text{mcv}}^G(n) = \begin{cases} \mathbf{T} & \text{if } n \text{ is forced true} \\ \mathbf{F} & \text{otherwise} \end{cases}$$
$$\alpha_{\text{lcv}}^G(n) = \begin{cases} \mathbf{F} & \text{if } n \text{ is forced false} \\ \mathbf{T} & \text{otherwise} \end{cases}$$

Note: α_{mcv}^G is the valuation constructed in the previous proof.

Properties of Most/Least Conservative Valuations

Theorem (Properties of Most/Least Conservative Valuations)

Let G be an AND/OR graph. Then:

- 1 α_{mcv}^G is consistent.
- 2 α_{lcv}^G is consistent.
- 3 For all consistent valuations α of G ,
 $on(\alpha_{\text{mcv}}^G) \subseteq on(\alpha) \subseteq on(\alpha_{\text{lcv}}^G)$.

Properties of MCV/LCV: Proof

Proof.

Part 1. was shown in the preceding proof. We showed that the valuation α considered in this proof is consistent and satisfies $\alpha(n) = \mathbf{T}$ iff n is forced true, which implies $\alpha = \alpha_{\text{mcv}}^G$.

The proof of Part 2. is analogous, using the rules for forced false nodes instead of forced true nodes.

Part 3 follows directly from the definitions of forced nodes, α_{mcv}^G and α_{lcv}^G . □

Properties of MCV/LCV: Consequences

This theorem answers our remaining questions about the existence, uniqueness, structure and computation of consistent valuations:

- Consistent valuations always exist and can be efficiently computed.
- All consistent valuations lie between the most and least conservative one.
- There is a unique consistent valuation iff $\alpha_{\text{mcv}}^G = \alpha_{\text{lcv}}^G$, or equivalently iff each node is forced true or forced false.

Summary

Summary

- **AND/OR graphs** are directed graphs with **AND nodes** and **OR nodes**.
- We can assign **truth values** to AND/OR graph nodes.
- Such valuations are called **consistent** if they match the intuitive meaning of “AND” and “OR”.
- Consistent valuations always exist.
- Consistent valuations can be computed efficiently.
- All consistent valuations fall between two extremes:
 - the **most conservative valuation**, where only nodes that are **forced to be true** are true
 - the **least conservative valuation**, where all nodes that are **not forced to be false** are true