

Planning and Optimization

D2. Delete Relaxation: Properties of Relaxed Planning Tasks

Malte Helmert and Gabriele Röger

Universität Basel

October 19, 2026

Planning and Optimization

October 19, 2026 — D2. Delete Relaxation: Properties of Relaxed Planning Tasks

D2.1 The Domination Lemma

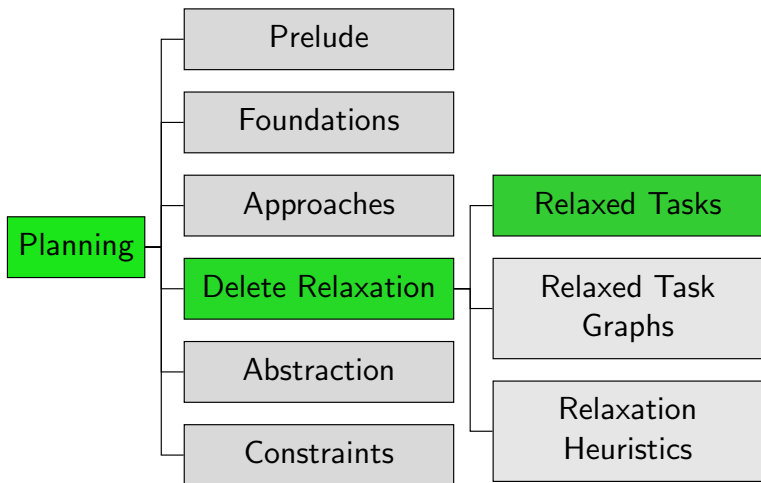
D2.2 The Relaxation Lemma

D2.3 Consequences

D2.4 Monotonicity

D2.5 Summary

Content of the Course



D2.1 The Domination Lemma

On-Set and Dominating States

Definition (On-Set)

The **on-set** of an interpretation s is the set of propositional variables that are true in s , i.e., $on(s) = s^{-1}(\{\mathbf{T}\})$.

$\rightsquigarrow$ for **states** of propositional planning tasks:
states can be viewed as **sets** of (true) state variables

Definition (Dominate)

An interpretation s' **dominates** an interpretation s if $on(s) \subseteq on(s')$.

$\rightsquigarrow$ all state variables true in s are also true in s'

Domination Lemma (1)

Lemma (Domination)

Let s and s' be interpretations of a set of propositional variables V , and let χ be a propositional formula over V which does not contain negation symbols.

If $s \models \chi$ and s' dominates s , then $s' \models \chi$.

Proof.

Proof by induction over the structure of χ .

- ▶ Base case $\chi = \top$: then $s' \models \top$.
- ▶ Base case $\chi = \perp$: then $s \not\models \perp$.

...

Domination Lemma (2)

Proof (continued).

- ▶ **Base case** $\chi = v \in V$: if $s \models v$, then $v \in on(s)$.
With $on(s) \subseteq on(s')$, we get $v \in on(s')$ and hence $s' \models v$.
- ▶ **Inductive case** $\chi = \chi_1 \wedge \chi_2$: by induction hypothesis, our claim holds for the proper subformulas χ_1 and χ_2 of χ .

$$\begin{aligned}
 s \models \chi &\implies s \models \chi_1 \wedge \chi_2 \\
 &\implies s \models \chi_1 \text{ and } s \models \chi_2 \\
 &\stackrel{\text{I.H. (twice)}}{\implies} s' \models \chi_1 \text{ and } s' \models \chi_2 \\
 &\implies s' \models \chi_1 \wedge \chi_2 \\
 &\implies s' \models \chi.
 \end{aligned}$$

- ▶ **Inductive case** $\chi = \chi_1 \vee \chi_2$: analogous



D2.2 The Relaxation Lemma

Add Sets and Delete Sets

Definition (Add Set and Delete Set for an Effect)

Consider a propositional planning task with state variables V .

Let e be an effect over V , and let s be a state over V .

The **add set** of e in s , written $addset(e, s)$,

and the **delete set** of e in s , written $delset(e, s)$,

are defined as the following sets of state variables:

$$addset(e, s) = \{v \in V \mid s \models trig(v, e)\}$$

$$delset(e, s) = \{v \in V \mid s \models trig(\neg v, e)\}$$

Note: For all states s and operators o applicable in s , we have
 $on(s[o]) = (on(s) \setminus delset(eff(o), s)) \cup addset(eff(o), s)$.

Relaxation Lemma

For this and the following chapters on delete relaxation, we assume implicitly that we are working with **propositional planning tasks in positive normal form**.

Lemma (Relaxation)

Let s be a state, and let s' be a state that dominates s .

- ❶ *If o is an operator applicable in s ,
then o^+ is applicable in s' and $s' \llbracket o^+ \rrbracket$ dominates $s \llbracket o \rrbracket$.*
- ❷ *If π is an operator sequence applicable in s ,
then π^+ is applicable in s' and $s' \llbracket \pi^+ \rrbracket$ dominates $s \llbracket \pi \rrbracket$.*
- ❸ *If additionally π leads to a goal state from state s ,
then π^+ leads to a goal state from state s' .*

Proof of Relaxation Lemma (1)

Proof.

Let V be the set of state variables.

Part 1: Because o is applicable in s , we have $s \models pre(o)$.

Because $pre(o)$ is negation-free and s' dominates s , we get $s' \models pre(o)$ from the domination lemma.

Because $pre(o^+) = pre(o)$, this shows that o^+ is applicable in s' .

...

Proof of Relaxation Lemma (2)

Proof (continued).

To prove that $s' \llbracket o^+ \rrbracket$ dominates $s \llbracket o \rrbracket$,
we first compare the relevant add sets:

$$\begin{aligned} \text{addset}(\text{eff}(o), s) &= \{v \in V \mid s \models \text{trig}(v, \text{eff}(o))\} \\ &= \{v \in V \mid s \models \text{trig}(v, \text{eff}(o^+))\} \end{aligned} \quad (1)$$

$$\begin{aligned} &\subseteq \{v \in V \mid s' \models \text{trig}(v, \text{eff}(o^+))\} \quad (2) \\ &= \text{addset}(\text{eff}(o^+), s'), \end{aligned}$$

where (1) uses $\text{trig}(v, \text{eff}(o)) \equiv \text{trig}(v, \text{eff}(o^+))$
and (2) uses the dominance lemma (note that trigger conditions
are negation-free for operators in positive normal form). ...

Proof of Relaxation Lemma (3)

Proof (continued).

We then get:

$$\begin{aligned} on(s[o]) &= (on(s) \setminus delset(eff(o), s)) \cup addset(eff(o), s) \\ &\subseteq on(s) \cup addset(eff(o), s) \\ &\subseteq on(s') \cup addset(eff(o^+), s') \\ &= on(s'[o^+]), \end{aligned}$$

and thus $s'[o^+]$ dominates $s[o]$.

This concludes the proof of Part 1.

...

Proof of Relaxation Lemma (4)

Proof (continued).

Part 2: by induction over $n = |\pi|$

Base case: $\pi = \langle \rangle$

The empty operator sequence is trivially applicable in s' , and $s'[\langle \rangle^+] = s'$ dominates $s[\langle \rangle] = s$ by prerequisite.

Inductive case: $\pi = \langle o_1, \dots, o_{n+1} \rangle$

By the induction hypothesis, $\langle o_1^+, \dots, o_n^+ \rangle$ is applicable in s' , and $t' = s'[\langle o_1^+, \dots, o_n^+ \rangle]$ dominates $t = s[\langle o_1, \dots, o_n \rangle]$.

Also, o_{n+1} is applicable in t .

Using Part 1, o_{n+1}^+ is applicable in t' and $s'[\pi^+] = t'[\langle o_{n+1}^+ \rangle]$ dominates $s[\pi] = t[\langle o_{n+1} \rangle]$.

This concludes the proof of Part 2.

...

Proof of Relaxation Lemma (5)

Proof (continued).

Part 3: Let γ be the goal formula.

From Part 2, we obtain that $t' = s'[\llbracket \pi^+ \rrbracket]$ dominates $t = s[\llbracket \pi \rrbracket]$.

By prerequisite, t is a goal state and hence $t \models \gamma$.

Because the task is in positive normal form, γ is negation-free, and hence $t' \models \gamma$ because of the domination lemma.

Therefore, t' is a goal state. □

D2.3 Consequences

Consequences of the Relaxation Lemma

- ▶ The relaxation lemma is the main technical result that we will use to study delete relaxation.
- ▶ Next, we show two further properties of delete relaxation that will be useful for us.
- ▶ They are direct consequences of the relaxation lemma.

Consequences of the Relaxation Lemma (1)

Corollary (Relaxation Preserves Plans and Leads to Dominance)

Let π be an operator sequence that is applicable in state s .

Then π^+ is applicable in s and $s[\pi^+]$ dominates $s[\pi]$.

If π is a plan for Π , then π^+ is a plan for Π^+ .

Proof.

Apply relaxation lemma with $s' = s$. □

- ~> Relaxations of plans are relaxed plans.
- ~> Delete relaxation is no harder to solve than original task.
- ~> Optimal relaxed plans are never more expensive than optimal plans for original tasks.

Consequences of the Relaxation Lemma (2)

Corollary (Relaxation Preserves Dominance)

*Let s be a state, let s' be a state that dominates s ,
and let π^+ be a relaxed operator sequence applicable in s .
Then π^+ is applicable in s' and $s'[\![\pi^+]\!]$ dominates $s[\![\pi^+]\!]$.*

Proof.

Apply relaxation lemma with π^+ for π ,
noting that $(\pi^+)^+ = \pi^+$. □

- ~> If there is a relaxed plan starting from state s ,
the same plan can be used starting from a dominating state s' .
- ~> Dominating states are always “better” in relaxed tasks.

D2.4 Monotonicity

Monotonicity of Relaxed Planning Tasks

Lemma (Monotonicity)

*Let s be a state in which relaxed operator o^+ is applicable.
Then $s[o^+]$ dominates s .*

Proof.

Since relaxed operators only have positive effects,
we have $on(s) \subseteq on(s) \cup addset(eff(o^+), s) = on(s[o^+])$. □

↪ Together with our previous results, this means that
making a transition in a relaxed planning task **never** hurts.

Finding Relaxed Plans

Using the theory we developed, we are now ready to study the problem of **finding plans** for **relaxed planning tasks**.

~> next chapter

D2.5 Summary

Summary

- ▶ With positive normal form, having more true variables is good.
- ▶ We can formalize this as **dominance** between states.
- ▶ It follows that delete relaxation is a **simplification**:
it is never harder to solve a relaxed task than the original one.
- ▶ In delete-relaxed tasks, applying an operator always takes us to a dominating state and therefore never hurts.