

Planning and Optimization

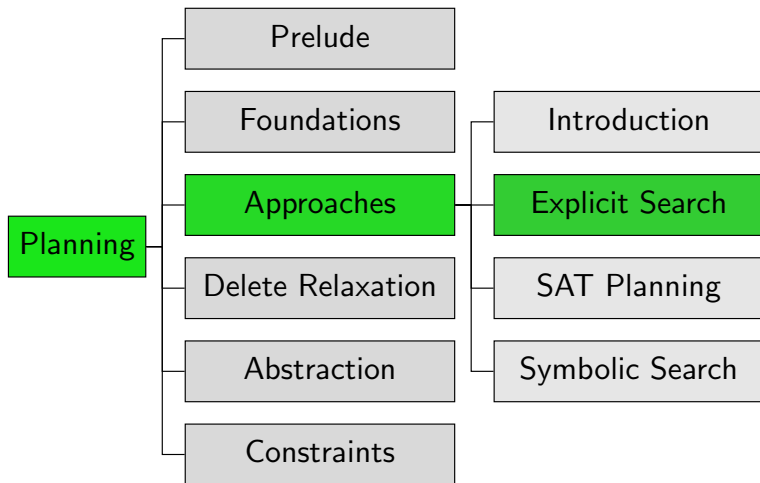
C4. General Regression

Malte Helmert and Gabriele Röger

Universität Basel

October 7, 2026

Content of the Course



Regression for General Planning Tasks

- With disjunctions and conditional effects, things become more tricky. How to regress $a \vee (b \wedge c)$ with respect to $\langle q, d \triangleright b \rangle$?
- In this chapter, we show how to regress **general sets of states** through **general operators**.
- We extensively use the idea of representing sets of states as formulas.

Regressing State Variables

Regressing State Variables: Motivation

Key question for general regression:

- Assume we are applying an operator with effect e .
- What must be true in the predecessor state for propositional state variable v to be true in the successor state?

If we can answer this question, a general definition of regression is only a small additional step.

Regressing State Variables: Key Idea

Assume we are in state s and apply effect e to obtain successor state s' .

Propositional state variable v is true in s' iff

- effect e **makes it true**, or
- it **remains true**, i.e., it is true in s and not made false by e .

Regressing a State Variable Through an Effect

Definition (Regressing a State Variable Through an Effect)

Let e be an effect of a propositional planning task,
and let v be a propositional state variable.

The **regression of v through e** , written $\text{regr}(v, e)$,
is defined as the following logical formula:

$$\text{regr}(v, e) = \text{trig}(v, e) \vee (v \wedge \neg \text{trig}(\neg v, e)).$$

Does this capture add-after-delete semantics correctly?

Regressing State Variables: Example

Example

Let $e = (b \triangleright a) \wedge (c \triangleright \neg a) \wedge b \wedge \neg d$.

v	$trig(v, e)$	$trig(\neg v, e)$	$regr(v, e)$
a	b	c	$b \vee (a \wedge \neg c)$
b	$\top$	$\perp$	$\top \vee (b \wedge \neg \perp) \equiv \top$
c	$\perp$	$\perp$	$\perp \vee (c \wedge \neg \perp) \equiv c$
d	$\perp$	$\top$	$\perp \vee (d \wedge \neg \top) \equiv \perp$

Reminder: $regr(v, e) = trig(v, e) \vee (v \wedge \neg trig(\neg v, e))$

Regressing State Variables: Correctness (1)

Lemma (Correctness of $\text{regr}(v, e)$)

Let s be a state, e be an effect and v be a state variable of a propositional planning task.

Then $s \models \text{regr}(v, e)$ iff $s[e] \models v$.

Regressing State Variables: Correctness (2)

Proof.

$(\Rightarrow)$: We know $s \models \text{regr}(v, e)$, and hence
 $s \models \text{trig}(v, e) \vee (v \wedge \neg \text{trig}(\neg v, e))$.

Do a case analysis on the two disjuncts.

Regressing State Variables: Correctness (2)

Proof.

($\Rightarrow$): We know $s \models \text{regr}(v, e)$, and hence
 $s \models \text{trig}(v, e) \vee (v \wedge \neg \text{trig}(\neg v, e))$.

Do a case analysis on the two disjuncts.

Case 1: $s \models \text{trig}(v, e)$.

Then $s[e] \models v$ by the first case in the definition of $s[e]$ (Ch. B3).

Regressing State Variables: Correctness (2)

Proof.

($\Rightarrow$): We know $s \models \text{regr}(v, e)$, and hence
 $s \models \text{trig}(v, e) \vee (v \wedge \neg \text{trig}(\neg v, e))$.

Do a case analysis on the two disjuncts.

Case 1: $s \models \text{trig}(v, e)$.

Then $s[e] \models v$ by the first case in the definition of $s[e]$ (Ch. B3).

Case 2: $s \models (v \wedge \neg \text{trig}(\neg v, e))$.

Then $s \models v$ and $s \not\models \text{trig}(\neg v, e)$.

We may additionally assume $s \not\models \text{trig}(v, e)$

because otherwise we can apply Case 1 of this proof.

Then $s[e] \models v$ by the third case in the definition of $s[e]$.

...

Regressing State Variables: Correctness (3)

Proof (continued).

($\Leftarrow$): Proof by contraposition.

We show that if $\text{regr}(v, e)$ is **false** in s , then v is **false** in $s[e]$.

Regressing State Variables: Correctness (3)

Proof (continued).

($\Leftarrow$): Proof by contraposition.

We show that if $\text{regr}(v, e)$ is **false** in s , then v is **false** in $s[e]$.

- By prerequisite, $s \not\models \text{trig}(v, e) \vee (v \wedge \neg \text{trig}(\neg v, e))$.

Regressing State Variables: Correctness (3)

Proof (continued).

($\Leftarrow$): Proof by contraposition.

We show that if $regr(v, e)$ is **false** in s , then v is **false** in $s[e]$.

- By prerequisite, $s \not\models trig(v, e) \vee (v \wedge \neg trig(\neg v, e))$.
- Hence $s \models \neg trig(v, e) \wedge (\neg v \vee trig(\neg v, e))$.

Regressing State Variables: Correctness (3)

Proof (continued).

($\Leftarrow$): Proof by contraposition.

We show that if $regr(v, e)$ is **false** in s , then v is **false** in $s[e]$.

- By prerequisite, $s \not\models trig(v, e) \vee (v \wedge \neg trig(\neg v, e))$.
- Hence $s \models \neg trig(v, e) \wedge (\neg v \vee trig(\neg v, e))$.
- From the first conjunct, we get $s \models \neg trig(v, e)$ and hence $s \not\models trig(v, e)$.

Regressing State Variables: Correctness (3)

Proof (continued).

($\Leftarrow$): Proof by contraposition.

We show that if $regr(v, e)$ is **false** in s , then v is **false** in $s[e]$.

- By prerequisite, $s \not\models trig(v, e) \vee (v \wedge \neg trig(\neg v, e))$.
- Hence $s \models \neg trig(v, e) \wedge (\neg v \vee trig(\neg v, e))$.
- From the first conjunct, we get $s \models \neg trig(v, e)$
and hence $s \not\models trig(v, e)$.
- From the second conjunct, we get $s \models \neg v \vee trig(\neg v, e)$.

Regressing State Variables: Correctness (3)

Proof (continued).

($\Leftarrow$): Proof by contraposition.

We show that if $regr(v, e)$ is **false** in s , then v is **false** in $s[e]$.

- By prerequisite, $s \not\models trig(v, e) \vee (v \wedge \neg trig(\neg v, e))$.
- Hence $s \models \neg trig(v, e) \wedge (\neg v \vee trig(\neg v, e))$.
- From the first conjunct, we get $s \models \neg trig(v, e)$ and hence $s \not\models trig(v, e)$.
- From the second conjunct, we get $s \models \neg v \vee trig(\neg v, e)$.
- **Case 1:** $s \models \neg v$. Then v is false before applying e and remains false, so $s[e] \not\models v$.

Regressing State Variables: Correctness (3)

Proof (continued).

($\Leftarrow$): Proof by contraposition.

We show that if $regr(v, e)$ is **false** in s , then v is **false** in $s[e]$.

- By prerequisite, $s \not\models trig(v, e) \vee (v \wedge \neg trig(\neg v, e))$.
- Hence $s \models \neg trig(v, e) \wedge (\neg v \vee trig(\neg v, e))$.
- From the first conjunct, we get $s \models \neg trig(v, e)$ and hence $s \not\models trig(v, e)$.
- From the second conjunct, we get $s \models \neg v \vee trig(\neg v, e)$.
- **Case 1:** $s \models \neg v$. Then v is false before applying e and remains false, so $s[e] \not\models v$.
- **Case 2:** $s \models trig(\neg v, e)$. Then v is deleted by e and not simultaneously added, so $s[e] \not\models v$.



Regressing Formulas Through Effects

Regressing Formulas Through Effects: Idea

- We can now generalize regression from state variables to general formulas over state variables.
- The basic idea is to replace **every occurrence** of every state variable v by $regr(v, e)$ as defined in the previous section.
- The following definition makes this more formal.

Regressing Formulas Through Effects: Definition

Definition (Regressing a Formula Through an Effect)

In a propositional planning task, let e be an effect, and let φ be a formula over propositional state variables.

The **regression of φ through e** , written $\text{regr}(\varphi, e)$, is defined as the following logical formula:

$$\text{regr}(\top, e) = \top$$

$$\text{regr}(\perp, e) = \perp$$

$$\text{regr}(v, e) = \text{trig}(v, e) \vee (v \wedge \neg \text{trig}(\neg v, e))$$

$$\text{regr}(\neg\psi, e) = \neg \text{regr}(\psi, e)$$

$$\text{regr}(\psi \vee \chi, e) = \text{regr}(\psi, e) \vee \text{regr}(\chi, e)$$

$$\text{regr}(\psi \wedge \chi, e) = \text{regr}(\psi, e) \wedge \text{regr}(\chi, e).$$

Regressing Formulas Through Effects: Example

Example

Let $e = (b \triangleright a) \wedge (c \triangleright \neg a) \wedge b \wedge \neg d$.

Recall:

- $\text{regr}(a, e) \equiv b \vee (a \wedge \neg c)$
- $\text{regr}(b, e) \equiv \top$
- $\text{regr}(c, e) \equiv c$
- $\text{regr}(d, e) \equiv \perp$

We get:

$$\begin{aligned}\text{regr}((a \vee d) \wedge (c \vee d), e) &\equiv ((b \vee (a \wedge \neg c)) \vee \perp) \wedge (c \vee \perp) \\ &\equiv (b \vee (a \wedge \neg c)) \wedge c \\ &\equiv b \wedge c\end{aligned}$$

Regressing Formulas Through Effects: Correctness (1)

Lemma (Correctness of $\text{regr}(\varphi, e)$)

Let φ be a logical formula, e an effect and s a state of a propositional planning task.

Then $s \models \text{regr}(\varphi, e)$ iff $s[e] \models \varphi$.

Regressing Formulas Through Effects: Correctness (2)

Proof.

The proof is by structural induction on φ .

Regressing Formulas Through Effects: Correctness (2)

Proof.

The proof is by structural induction on φ .

Induction hypothesis: $s \models \text{regr}(\psi, e)$ iff $s[e] \models \psi$
for all proper subformulas ψ of φ .

Regressing Formulas Through Effects: Correctness (2)

Proof.

The proof is by structural induction on φ .

Induction hypothesis: $s \models \text{regr}(\psi, e)$ iff $s[e] \models \psi$
for all proper subformulas ψ of φ .

Base case $\varphi = \top$:

We have $\text{regr}(\top, e) = \top$, and $s \models \top$ iff $s[e] \models \top$ is correct.

Regressing Formulas Through Effects: Correctness (2)

Proof.

The proof is by structural induction on φ .

Induction hypothesis: $s \models \text{regr}(\psi, e)$ iff $s[e] \models \psi$
for all proper subformulas ψ of φ .

Base case $\varphi = \top$:

We have $\text{regr}(\top, e) = \top$, and $s \models \top$ iff $s[e] \models \top$ is correct.

Base case $\varphi = \perp$:

We have $\text{regr}(\perp, e) = \perp$, and $s \models \perp$ iff $s[e] \models \perp$ is correct.

Regressing Formulas Through Effects: Correctness (2)

Proof.

The proof is by structural induction on φ .

Induction hypothesis: $s \models \text{regr}(\psi, e)$ iff $s[e] \models \psi$
for all proper subformulas ψ of φ .

Base case $\varphi = \top$:

We have $\text{regr}(\top, e) = \top$, and $s \models \top$ iff $s[e] \models \top$ is correct.

Base case $\varphi = \perp$:

We have $\text{regr}(\perp, e) = \perp$, and $s \models \perp$ iff $s[e] \models \perp$ is correct.

Base case $\varphi = v$:

We have $s \models \text{regr}(v, e)$ iff $s[e] \models v$ from the previous lemma. ...

Regressing Formulas Through Effects: Correctness (3)

Proof (continued).

Inductive case $\varphi = \neg\psi$:

$$\begin{aligned} s \models \text{regr}(\neg\psi, e) &\text{ iff } s \models \neg\text{regr}(\psi, e) \\ &\text{ iff } s \not\models \text{regr}(\psi, e) \\ &\text{ iff } s[e] \not\models \psi \\ &\text{ iff } s[e] \models \neg\psi \end{aligned}$$

Regressing Formulas Through Effects: Correctness (3)

Proof (continued).

Inductive case $\varphi = \neg\psi$:

$$\begin{aligned} s \models \text{regr}(\neg\psi, e) &\text{ iff } s \models \neg\text{regr}(\psi, e) \\ &\text{ iff } s \not\models \text{regr}(\psi, e) \\ &\text{ iff } s[e] \not\models \psi \\ &\text{ iff } s[e] \models \neg\psi \end{aligned}$$

Inductive case $\varphi = \psi \vee \chi$:

$$\begin{aligned} s \models \text{regr}(\psi \vee \chi, e) &\text{ iff } s \models \text{regr}(\psi, e) \vee \text{regr}(\chi, e) \\ &\text{ iff } s \models \text{regr}(\psi, e) \text{ or } s \models \text{regr}(\chi, e) \\ &\text{ iff } s[e] \models \psi \text{ or } s[e] \models \chi \\ &\text{ iff } s[e] \models \psi \vee \chi \end{aligned}$$

Regressing Formulas Through Effects: Correctness (3)

Proof (continued).

Inductive case $\varphi = \neg\psi$:

$$\begin{aligned} s \models \text{regr}(\neg\psi, e) &\text{ iff } s \models \neg\text{regr}(\psi, e) \\ &\text{ iff } s \not\models \text{regr}(\psi, e) \\ &\text{ iff } s[e] \not\models \psi \\ &\text{ iff } s[e] \models \neg\psi \end{aligned}$$

Inductive case $\varphi = \psi \vee \chi$:

$$\begin{aligned} s \models \text{regr}(\psi \vee \chi, e) &\text{ iff } s \models \text{regr}(\psi, e) \vee \text{regr}(\chi, e) \\ &\text{ iff } s \models \text{regr}(\psi, e) \text{ or } s \models \text{regr}(\chi, e) \\ &\text{ iff } s[e] \models \psi \text{ or } s[e] \models \chi \\ &\text{ iff } s[e] \models \psi \vee \chi \end{aligned}$$

Inductive case $\varphi = \psi \wedge \chi$:

Like previous case, replacing “ $\vee$ ” by “ $\wedge$ ”
and replacing “or” by “and”.



Regressing Formulas Through Operators

Regressing Formulas Through Operators: Idea

- We can now regress arbitrary formulas through arbitrary effects.
- The last missing piece is a definition of regression through **operators**, describing exactly in which states s applying a given operator o leads to a state satisfying a given formula φ .
- There are two requirements:
 - The operator o must be **applicable** in the state s .
 - The **resulting state** $s[o]$ must **satisfy** φ .

Regressing Formulas Through Operators: Definition

Definition (Regressing a Formula Through an Operator)

In a propositional planning task, let o be an operator, and let φ be a formula over state variables.

The **regression of φ through o** , written $\text{regr}(\varphi, o)$, is defined as the following logical formula:

$$\text{regr}(\varphi, o) = \text{pre}(o) \wedge \text{regr}(\varphi, \text{eff}(o)).$$

Regressing Formulas Through Operators: Correctness (1)

Theorem (Correctness of $\text{regr}(\varphi, o)$)

Let φ be a logical formula, o an operator and s a state of a propositional planning task.

Then $s \models \text{regr}(\varphi, o)$ iff o is applicable in s and $s[o] \models \varphi$.

Regressing Formulas Through Operators: Correctness (2)

Reminder: $\text{regr}(\varphi, o) = \text{pre}(o) \wedge \text{regr}(\varphi, \text{eff}(o))$

Proof.

Case 1: $s \models \text{pre}(o)$.

Then o is applicable in s and the statement we must prove simplifies to: $s \models \text{regr}(\varphi, e)$ iff $s[e] \models \varphi$, where $e = \text{eff}(o)$.

This was proved in the previous lemma.

Regressing Formulas Through Operators: Correctness (2)

Reminder: $\text{regr}(\varphi, o) = \text{pre}(o) \wedge \text{regr}(\varphi, \text{eff}(o))$

Proof.

Case 1: $s \models \text{pre}(o)$.

Then o is applicable in s and the statement we must prove simplifies to: $s \models \text{regr}(\varphi, e)$ iff $s \llbracket e \rrbracket \models \varphi$, where $e = \text{eff}(o)$. This was proved in the previous lemma.

Case 2: $s \not\models \text{pre}(o)$.

Then $s \not\models \text{regr}(\varphi, o)$ and o is not applicable in s .

Hence both statements are false and therefore equivalent. □

Regression Examples (1)

Examples: compute regression and simplify to DNF

- $\text{regr}(b, \langle a, b \rangle)$
 $\equiv a \wedge (\top \vee (b \wedge \neg \perp))$
 $\equiv a$
- $\text{regr}(b \wedge c \wedge d, \langle a, b \rangle)$
 $\equiv a \wedge (\top \vee (b \wedge \neg \perp)) \wedge (\perp \vee (c \wedge \neg \perp)) \wedge (\perp \vee (d \wedge \neg \perp))$
 $\equiv a \wedge c \wedge d$
- $\text{regr}(b \wedge \neg c, \langle a, b \wedge c \rangle)$
 $\equiv a \wedge (\top \vee (b \wedge \neg \perp)) \wedge \neg(\top \vee (c \wedge \neg \perp))$
 $\equiv a \wedge \top \wedge \perp$
 $\equiv \perp$

Regression Examples (2)

Examples: compute regression and simplify to DNF

- $\text{regr}(b, \langle a, c \triangleright b \rangle)$
 - $\equiv a \wedge (c \vee (b \wedge \neg \perp))$
 - $\equiv a \wedge (c \vee b)$
 - $\equiv (a \wedge c) \vee (a \wedge b)$
- $\text{regr}(b, \langle a, (c \triangleright b) \wedge ((d \wedge \neg c) \triangleright \neg b) \rangle)$
 - $\equiv a \wedge (c \vee (b \wedge \neg(d \wedge \neg c)))$
 - $\equiv a \wedge (c \vee (b \wedge (\neg d \vee c)))$
 - $\equiv a \wedge (c \vee (b \wedge \neg d) \vee (b \wedge c))$
 - $\equiv a \wedge (c \vee (b \wedge \neg d))$
 - $\equiv (a \wedge c) \vee (a \wedge b \wedge \neg d)$

Summary

Summary

- Regressing a **propositional state variable** through an (arbitrary) operator must consider two cases:
 - state variables **made true** (by add effects)
 - state variables **remaining true** (by absence of delete effects)
- Regression of propositional state variables can be generalized to arbitrary formulas φ by replacing each occurrence of a state variable in φ by its regression.
- **Regressing a formula φ** through an **operator** involves regressing φ through the effect and enforcing the precondition.