

Planning and Optimization

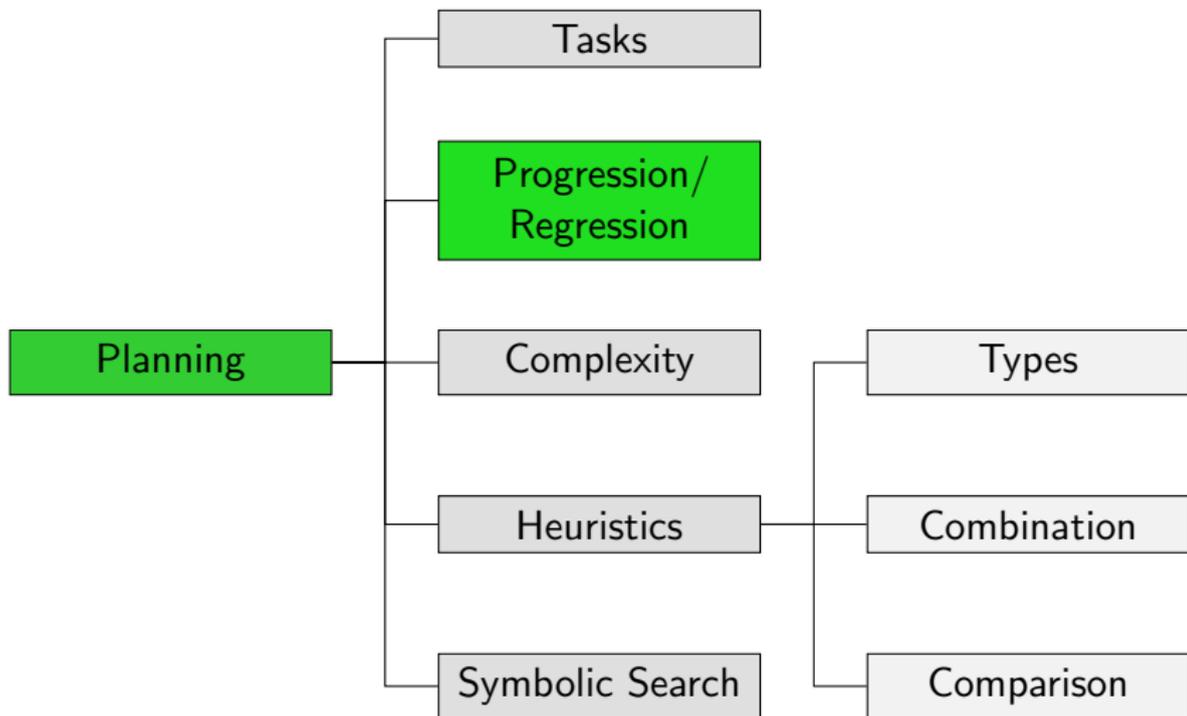
B4. General Regression, Part II

Malte Helmert and Gabriele Röger

Universität Basel

October 16, 2017

Content of this Course



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.

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: $s \models \text{regr}(\varphi, \text{eff}(o))$ iff $s[[o]] \models \varphi$.

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: $s \models \text{regr}(\varphi, \text{eff}(o))$ iff $s[[o]] \models \varphi$.

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

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

Practical Issues

Emptiness and Subsumption Testing

The following two tests are useful when performing regression searches to avoid exploring unpromising branches:

- Test that $\text{regr}(\varphi, o)$ does not represent the empty set (which would mean that search is in a dead end).
For example, $\text{regr}(p, \langle a, \neg p \rangle) \equiv a \wedge (\perp \vee (p \wedge \neg \top)) \equiv \perp$.
- Test that $\text{regr}(\varphi, o)$ does not represent a subset of φ (which would mean that the resulting search state is worse than φ and can be pruned).
For example, $\text{regr}(a, \langle b, c \rangle) \equiv a \wedge b$.

Both of these problems are **NP-complete**.

Formula Growth

The formula $\text{regr}(\text{regr}(\dots \text{regr}(\varphi, o_n) \dots, o_2), o_1)$ may have size $O(|\varphi| |o_1| |o_2| \dots |o_{n-1}| |o_n|)$, i.e., the product of the sizes of φ and the operators.

\rightsquigarrow worst-case **exponential** size $\Omega(|\varphi|^n)$

Logical Simplifications

- $\perp \wedge \varphi \equiv \perp$, $\top \wedge \varphi \equiv \varphi$, $\perp \vee \varphi \equiv \varphi$, $\top \vee \varphi \equiv \top$
- $a \vee \varphi \equiv a \vee \varphi[\perp/a]$, $\neg a \vee \varphi \equiv \neg a \vee \varphi[\top/a]$,
 $a \wedge \varphi \equiv a \wedge \varphi[\top/a]$, $\neg a \wedge \varphi \equiv \neg a \wedge \varphi[\perp/a]$
- idempotence, absorption, commutativity, associativity, ...

Restricting Formula Growth in Search Trees

Problem very big formulas obtained by regression

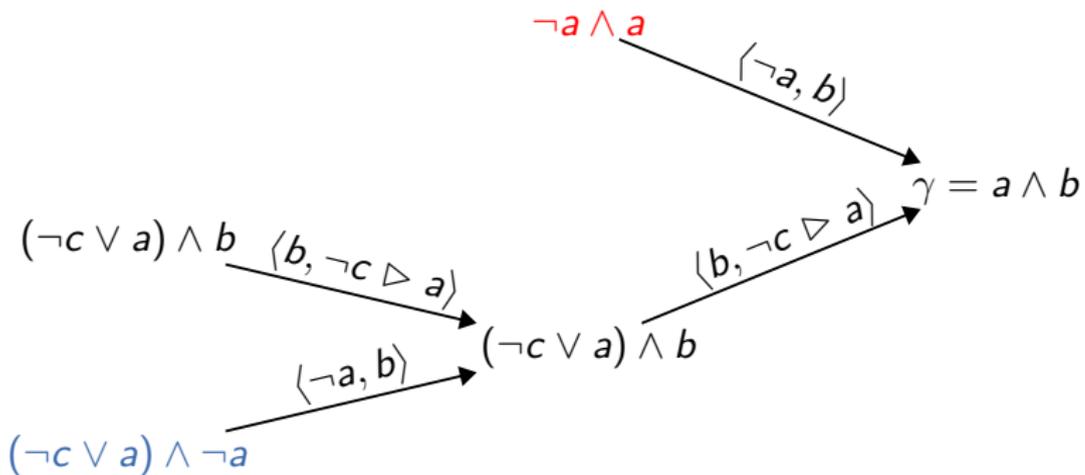
Cause **disjunctivity** in the (NNF) formulas
(formulas **without disjunctions** easily convertible
to monomials $\ell_1 \wedge \dots \wedge \ell_n$ where ℓ_i are literals
and n is at most the number of state variables)

Idea split disjunctive formulas when generating search trees

Unrestricted Regression: Search Tree Example

Unrestricted regression: do not treat disjunctions specially

Goal $\gamma = a \wedge b$, initial state $I = \{a \mapsto \mathbf{F}, b \mapsto \mathbf{F}, c \mapsto \mathbf{F}\}$.



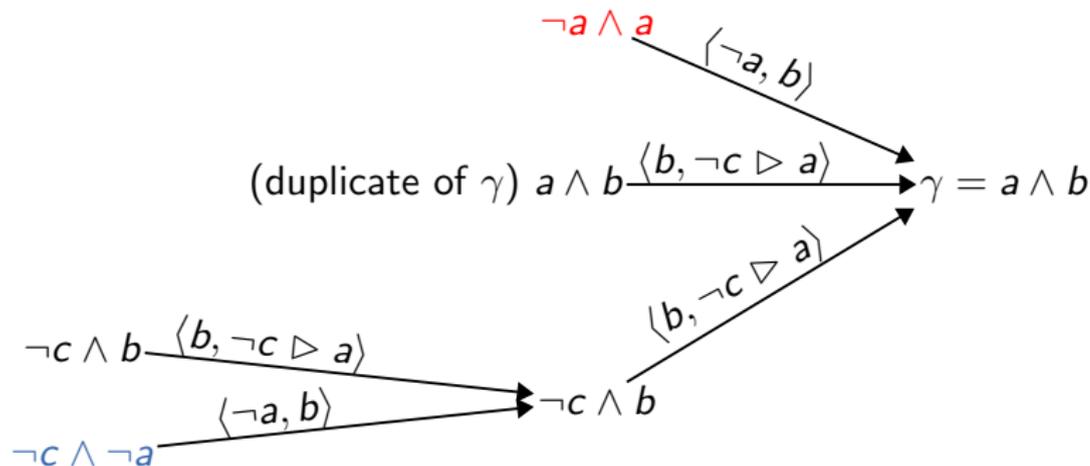
Full Splitting: Search Tree Example

Full splitting: always split all disjunctive formulas

Goal $\gamma = a \wedge b$, initial state $I = \{a \mapsto \mathbf{F}, b \mapsto \mathbf{F}, c \mapsto \mathbf{F}\}$.

$(\neg c \vee a) \wedge b$ in DNF: $(\neg c \wedge b) \vee (a \wedge b)$

\rightsquigarrow split into $\neg c \wedge b$ and $a \wedge b$



General Splitting Strategies

Alternatives:

- 1 Do nothing (**unrestricted regression**).
- 2 Always eliminate all disjunctivity (**full splitting**).
- 3 Reduce disjunctivity if formula becomes too big.

Discussion:

- **With unrestricted regression** formulas may have **sizes that are exponential** in the number of state variables.
- **With full splitting** search tree can be **exponentially bigger** than without splitting.
- The third option lies between these two extremes.

Summary

Summary

- Regressing a formula φ through an operator involves regressing φ through the effect and enforcing the precondition.
- When applying regression in practice, additional considerations come into play, including:
 - emptiness testing to prune dead-end search states
 - subsumption testing to prune dominated search states
 - logical simplifications and splitting to restrict formula growth