Planning and Optimization

B4. General Regression, Part II

Malte Helmert and Gabriele Röger

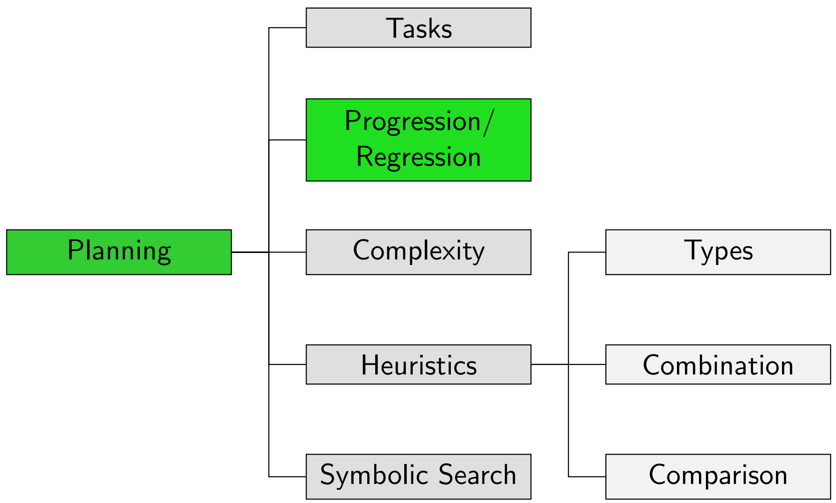
Universität Basel

October 16, 2017

Planning and Optimization

October 16, 2017 — B4. General Regression, Part II

Content of this Course



B4.1 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 $regr(\varphi, o)$, is defined as the following logical formula:

$$regr(\varphi, o) = pre(o) \land regr(\varphi, eff(o)).$$

Regressing Formulas Through Operators: Correctness (1)

Theorem (Correctness of $regr(\varphi, o)$)

Let φ be a logical formula, o an operator and s a state.

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

Regressing Formulas Through Operators: Correctness (2)

Reminder: $regr(\varphi, o) = pre(o) \land regr(\varphi, eff(o))$

Proof.

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

Then o is applicable in s and the statement we must prove simplifies to: $s \models regr(\varphi, eff(o))$ iff $s[\![o]\!] \models \varphi$.
This was proved in the previous lemma.

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

Then $s \not\models regr(\varphi, o)$ 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

- $regr(b, \langle a, b\rangle)$
  $\equiv a \land (\top \lor (b \land \neg\bot))$
  $\equiv a$
- $regr(b \land c \land d, \langle a, b\rangle)$
  $\equiv a \land (\top \lor (b \land \neg\bot)) \land (\bot \lor (c \land \neg\bot)) \land (\bot \lor (d \land \neg\bot))$
  $\equiv a \land c \land d$
- $regr(b \land \neg c, \langle a, b \land c\rangle)$
  $\equiv a \land (\top \lor (b \land \neg\bot)) \land \neg(\top \lor (c \land \neg\bot))$
  $\equiv a \land \top \land \bot$
  $\equiv \bot$

Regression Examples (2)

Examples: compute regression and simplify to DNF

- $regr(b, \langle a, c \rhd b\rangle)$
  $\equiv a \land (c \lor (b \land \neg\bot))$
  $\equiv a \land (c \lor b)$
  $\equiv (a \land c) \lor (a \land b)$
- $regr(b, \langle a, (c \rhd b) \land ((d \land \neg c) \rhd \neg b)\rangle)$
  $\equiv a \land (c \lor (b \land \neg(d \land \neg c)))$
  $\equiv a \land (c \lor (b \land (\neg d \lor c)))$
  $\equiv a \land (c \lor (b \land \neg d) \lor (b \land c))$
  $\equiv a \land (c \lor (b \land \neg d))$
  $\equiv (a \land c) \lor (a \land b \land \neg d)$

B4.2 Practical Issues

Emptiness and Subsumption Testing

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

- Test that $regr(\varphi, o)$ does not represent the empty set (which would mean that search is in a dead end).
  For example, $regr(p, \langle a, \neg p\rangle) \equiv a \land (\bot \lor (p \land \neg\top)) \equiv \bot$.
- Test that $regr(\varphi, o)$ 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, $regr(a, \langle b, c\rangle) \equiv a \land b$.

Both of these problems are NP-complete.

Formula Growth

The formula $regr(regr(\dots regr(\varphi, o_n) \dots, o_2), o_1)$ 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.
$\leadsto$ worst-case exponential size $\Omega(|\varphi|^n)$

Logical Simplifications

- $\bot \land \varphi \equiv \bot$, $\top \land \varphi \equiv \varphi$, $\bot \lor \varphi \equiv \varphi$, $\top \lor \varphi \equiv \top$
- $a \lor \varphi \equiv a \lor \varphi[\bot/a]$, $\neg a \lor \varphi \equiv \neg a \lor \varphi[\top/a]$,
  $a \land \varphi \equiv a \land \varphi[\top/a]$, $\neg a \land \varphi \equiv \neg a \land \varphi[\bot/a]$
- idempotence, absorption, commutativity, associativity, ...

Restricting Formula Growth in Search Trees

Problem very big formulas obtained by regression

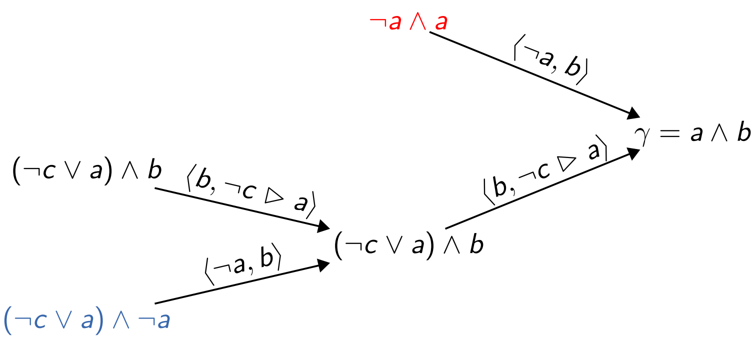
Cause disjunctivity in the (NNF) formulas
(formulas without disjunctions easily convertible to monomials $\ell_1 \land \dots \land \ell_n$ where ℓ_i are literals and n is at most the number of state variables)

Idea split disjunctive formulas when generating search trees

Unrestricted Regression: Search Tree Example

Unrestricted regression: do not treat disjunctions specially

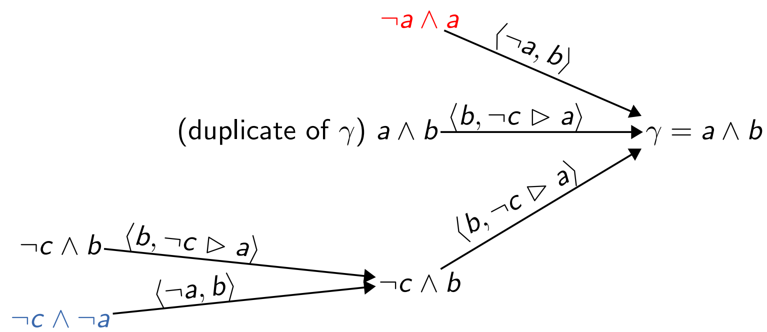
Goal $\gamma = a \land b$, initial state $I = \{a \mapsto \mathbf{F}, b \mapsto \mathbf{F}, c \mapsto \mathbf{F}\}$.



Full Splitting: Search Tree Example

Full splitting: always split all disjunctive formulas

Goal $\gamma = a \land b$, initial state $I = \{a \mapsto \mathbf{F}, b \mapsto \mathbf{F}, c \mapsto \mathbf{F}\}$.
$(\neg c \lor a) \land b$ in DNF: $(\neg c \land b) \lor (a \land b)$
$\leadsto$ split into $\neg c \land b$ and $a \land b$



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.

B4.3 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