

Planning and Optimization

B4. Practical Issues of Regression Search

Malte Helmert and Gabriele Röger

Universität Basel

Planning and Optimization

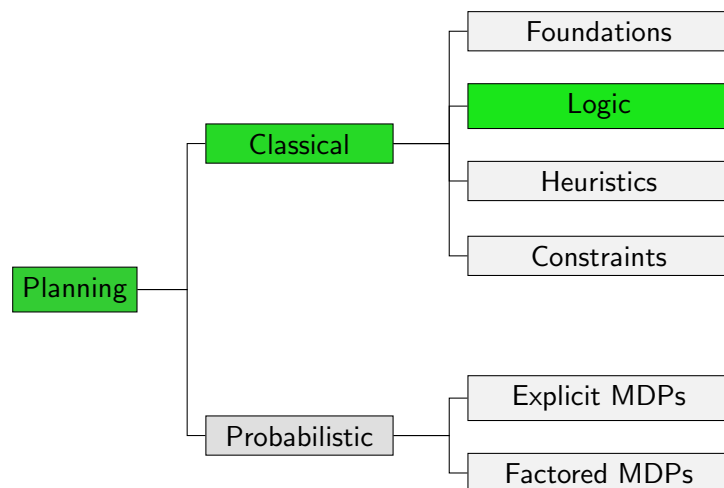
— B4. Practical Issues of Regression Search

B4.1 Unpromising Branches

B4.2 Formula Growth

B4.3 Summary

Content of this Course



Regression Search

regression search

- ▶ backward search from goal to initial state
- ▶ formulas represent sets of states
- ▶ regression computes possible predecessor states for a set of states and an operator

B4.1 Unpromising Branches

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**.

B4.2 Formula Growth

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

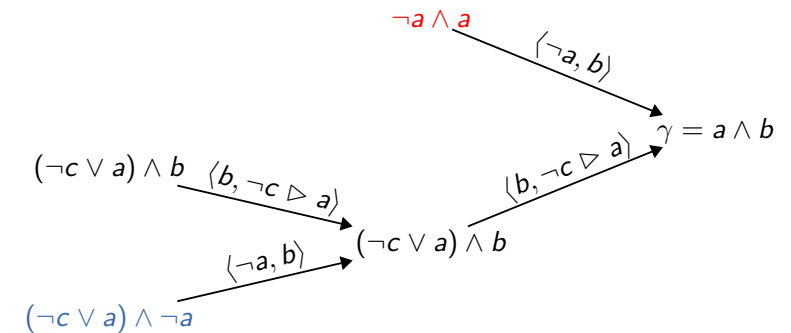
Cause **disjunctivity** in the (NNF) formulas
(formulas **without disjunctions** easily convertible to conjunctions $l_1 \wedge \dots \wedge l_n$ where l_i are literals and n is at most the number of state variables)

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 $I = \{a \mapsto \mathbf{F}, b \mapsto \mathbf{F}, c \mapsto \mathbf{F}\}$.



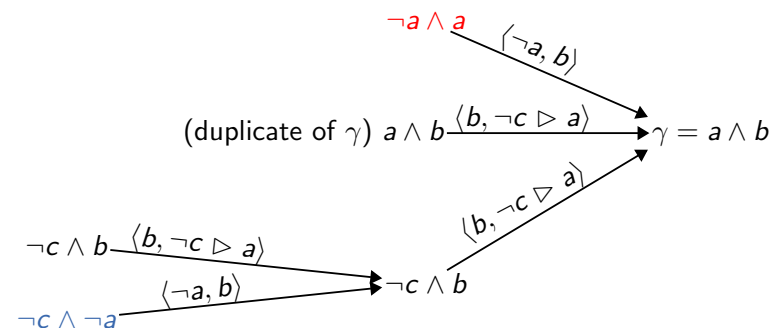
Full Splitting: Search Tree Example

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.

B4.3 Summary

Summary

When applying regression in practice, we need to consider

- ▶ **emptiness testing** to prune dead-end search states
- ▶ **subsumption testing** to prune dominated search states
- ▶ **logical simplifications** and **splitting** to restrict formula growth