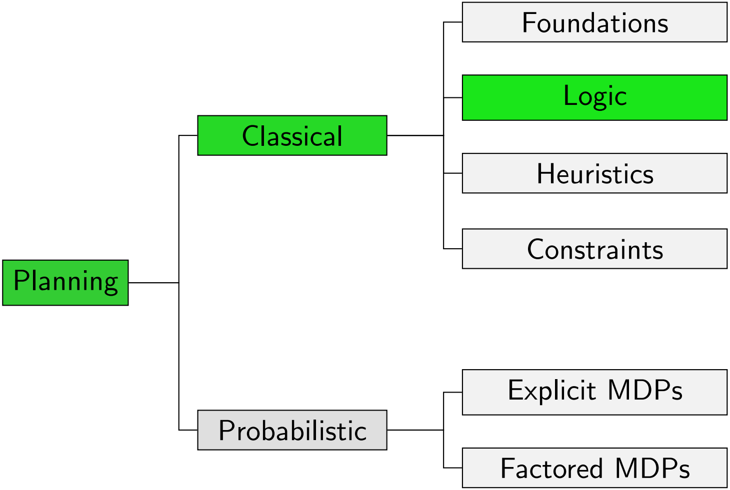
Planning and Optimization

B4. Practical Issues of Regression Search

Malte Helmert and Gabriele Röger

Universität Basel

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

Unpromising Branches

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, $regr(p, \langle a, \neg p \rangle) \equiv a \wedge (\bot \vee (p \wedge \neg \top)) \equiv \bot$.
- 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, $regr(a, \langle b, c \rangle) \equiv a \wedge b$.

Both of these problems are NP-complete.

Formula Growth

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.

$\rightsquigarrow$ 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

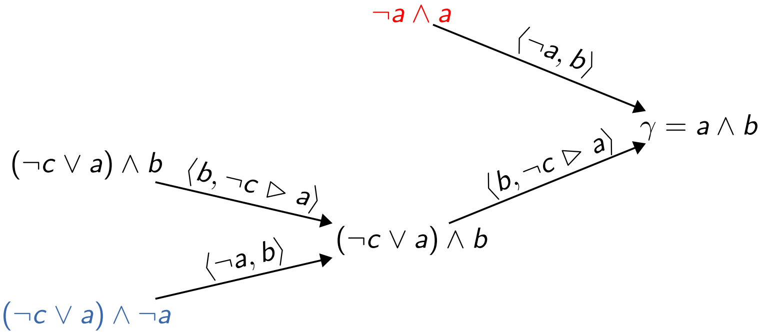
Cause disjunctivity in the (NNF) formulas
(formulas without disjunctions easily convertible to conjunctions $\ell_1 \wedge \cdots \wedge \ell_n$ where ℓ_i are literals and n is at most the number of state variables)

Idea split disjunctive formulas when generating search trees

Unrestricted Regression: Search Tree Example

Unrestricted regression: do not treat disjunctions specially

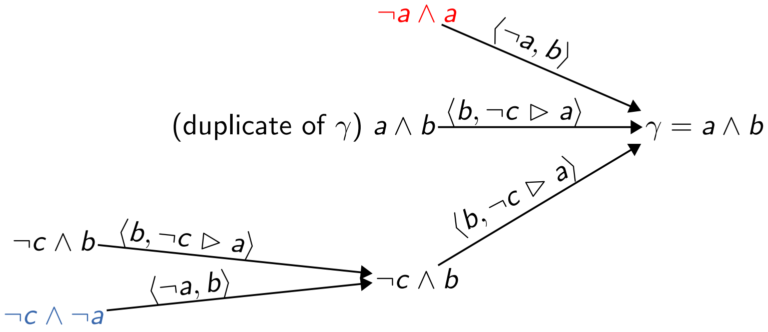
Goal $\gamma = a \wedge b$, initial state $I = \{a \mapsto \mathbf{F}, b \mapsto \mathbf{F}, c \mapsto \mathbf{F}\}$.



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

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