Planning and Optimization
B4. Equivalent Operators and Normal Forms

Malte Helmert and Gabriele Röger

Universität Basel

September 30, 2024

Planning and Optimization
September 30, 2024 — B4. Equivalent Operators and Normal Forms
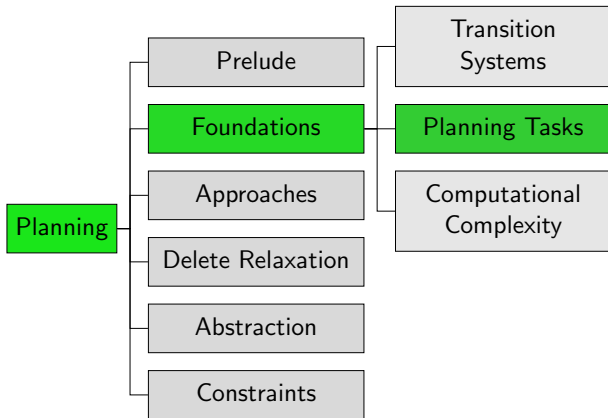
B4.1 Reminder & Motivation

B4.2 Equivalence Transformations

B4.3 Conflict-Free Operators

B4.4 Flat Effects

B4.5 Summary

Content of the Course

B4.1 Reminder & Motivation

Reminder: Syntax of Effects

Definition (Effect)

Effects over propositional state variables V
are inductively defined as follows:

► $\top$ is an effect (empty effect).

► If $v \in V$ is a propositional state variable,
  then v and $\neg v$ are effects (atomic effect).

► If e and e' are effects, then $(e \wedge e')$ is an effect
  (conjunctive effect).

► If χ is a formula over V and e is an effect,
  then $(\chi \rhd e)$ is an effect (conditional effect).

Arbitrary nesting of conjunctive and conditional effects,
e.g. $c \wedge (a \rhd (\neg b \wedge (c \rhd (b \wedge \neg d \wedge \neg a)))) \wedge (\neg b \rhd \neg a)$
⤳ Can we make our life easier?

Reminder: Semantics of Effects

▶ *effcond*(ℓ, e): condition that must be true in the current state
   for the effect e to trigger the atomic effect ℓ

▶ add-after-delete semantics:
   if an operator with effect e is applied in state s
   and we have both $s \models$ *effcond*(v, e) and $s \models$ *effcond*($\neg v$, e),
   then $s'(v) = \mathbf{T}$ in the resulting state s'.

This is a very subtle detail.
⤳ Can we make our life easier?

Motivation

Similarly to normal forms in propositional logic (DNF, CNF, NNF),
we can define normal forms for effects, operators
and planning tasks.

Among other things, we consider normal forms that avoid
complicated nesting and subtleties of conflicts.

This is useful because algorithms (and proofs) then only
need to deal with effects, operators and tasks in normal form.

Content of the Course
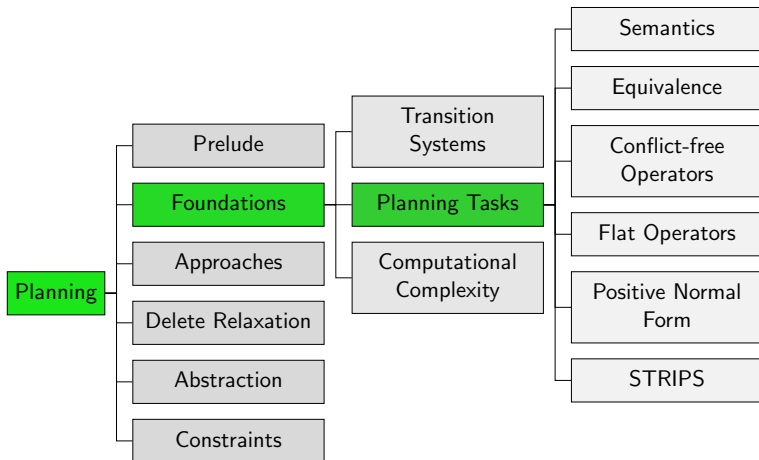
Notation: Applying Operator Sequences
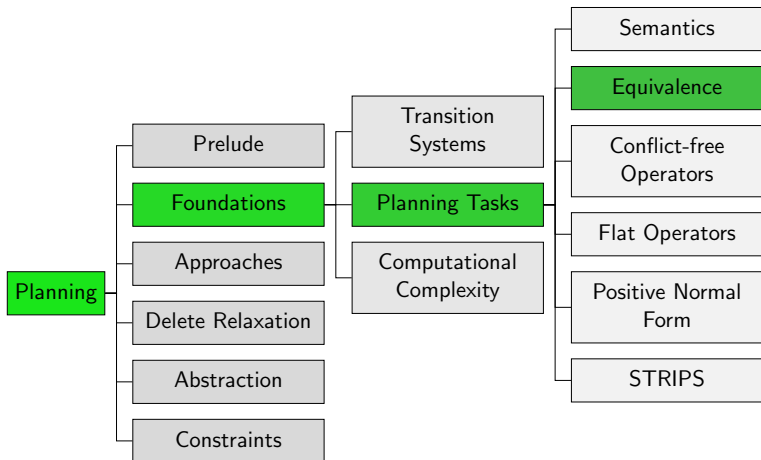
Existing notation:

▶ We already write $s[\![o]\!]$ for the resulting state
after applying operator o in state s.

New extended notation:

▶ For a sequence $\pi = \langle o_1, \ldots, o_n \rangle$ of operators
that are consecutively applicable in s,
we write $s[\![\pi]\!]$ for $s[\![o_1]\!][\![o_2]\!] \ldots [\![o_n]\!]$.

▶ This includes the case of an empty operator sequence:
$s[\![\langle\rangle]\!] = s$

B4.2 Equivalence Transformations

Content of the Course

Equivalence of Operators and Effects: Definition

Definition (Equivalent Effects)

Two effects e and e' over state variables V are equivalent,
written $e \equiv e'$, if $s[\![e]\!] = s[\![e']\!]$ for all states s.

Definition (Equivalent Operators)

Two operators o and o' over state variables V are equivalent,
written $o \equiv o'$, if $cost(o) = cost(o')$ and for all states s, s' over V,
o induces the transition $s \xrightarrow{o} s'$ iff o' induces the transition $s \xrightarrow{o'} s'$.

Equivalence of Operators and Effects: Theorem

Theorem
Let o and o' be operators with $pre(o) \equiv pre(o')$, $eff(o) \equiv eff(o')$ and $cost(o) = cost(o')$. Then $o \equiv o'$.

Note: The converse is not true. (Why not?)

Equivalence Transformations for Effects

$$e \wedge e' \;\equiv\; e' \wedge e \tag{1}$$

$$(e \wedge e') \wedge e'' \;\equiv\; e \wedge (e' \wedge e'') \tag{2}$$

$$\top \wedge e \;\equiv\; e \tag{3}$$

$$\chi \rhd e \;\equiv\; \chi' \rhd e \qquad \text{if } \chi \equiv \chi' \tag{4}$$

$$\top \rhd e \;\equiv\; e \tag{5}$$

$$\bot \rhd e \;\equiv\; \top \tag{6}$$

$$\chi \rhd (\chi' \rhd e) \;\equiv\; (\chi \wedge \chi') \rhd e \tag{7}$$

$$\chi \rhd (e \wedge e') \;\equiv\; (\chi \rhd e) \wedge (\chi \rhd e') \tag{8}$$

$$(\chi \rhd e) \wedge (\chi' \rhd e) \;\equiv\; (\chi \vee \chi') \rhd e \tag{9}$$

B4.3 Conflict-Free Operators

Content of the Course
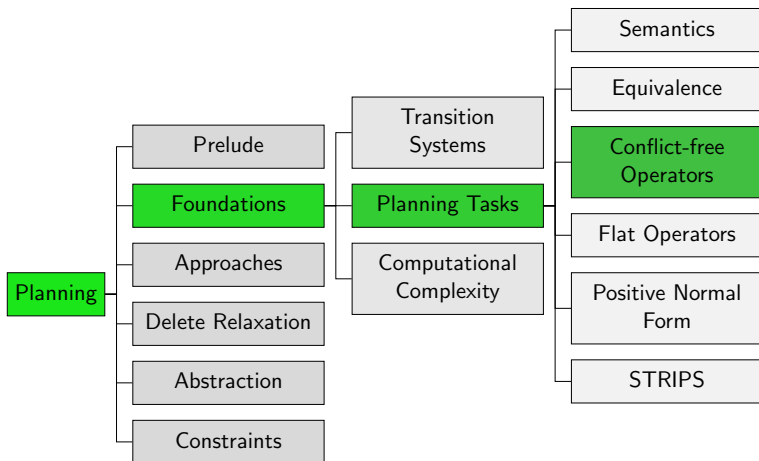
Conflict-Freeness: Motivation

- ▶ The add-after-delete semantics makes effects like $(a \rhd c) \wedge (b \rhd \neg c)$ somewhat unintuitive to interpret.
- ⤳ What happens in states where $a \wedge b$ is true?
- ▶ It would be nicer if *effcond*(ℓ, e) always were the condition under which the atomic effect ℓ actually materializes (because of add-after-delete, it is not)
- ⤳ introduce normal form where "complicated case" never arises

Conflict-Free Effects and Operators

> **Definition (Conflict-Free)**
>
> An effect e over propositional state variables V
> is called conflict-free if $effcond(v, e) \land effcond(\neg v, e)$
> is unsatisfiable for all $v \in V$.
>
> An operator o is called conflict-free if $eff(o)$ is conflict-free.

Making Operators Conflict-Free

▶ In general, testing whether an operator is conflict-free
  is a coNP-complete problem. (Why?)

▶ However, we do not necessarily need such a test.
  Instead, we can produce an equivalent conflict-free operator
  in polynomial time.

▶ Algorithm: given operator o, replace all atomic effects
  of the form $\neg v$ by $(\neg \mathit{effcond}(v, \mathit{eff}(o))) \rhd \neg v$.
  The resulting operator o' is conflict-free and $o \equiv o'$. (Why?)

B4.4 Flat Effects

Content of the Course
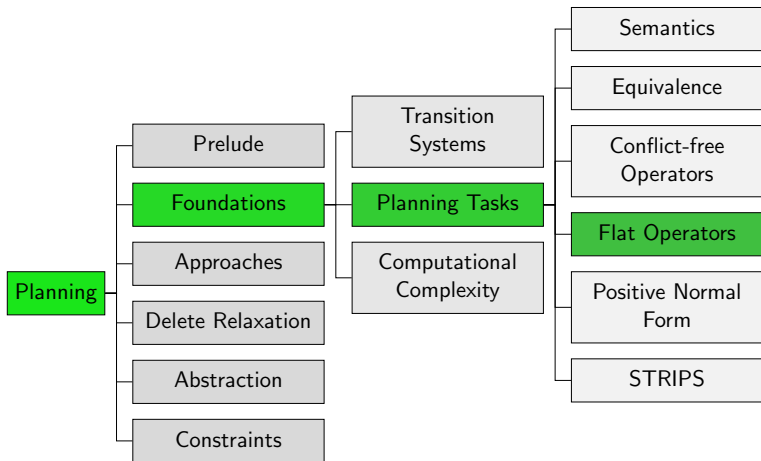
Flat Effects: Motivation

▶ CNF and DNF limit the nesting of connectives
   in propositional logic.

▶ For example, a CNF formula is
   ▶ a conjunction of 0 or more subformulas,
   ▶ each of which is a disjunction of 0 or more subformulas,
   ▶ each of which is a literal.

▶ Similarly, we can define a normal form that limits
   the nesting of effects.

▶ This is useful because we then do not have to consider
   arbitrarily structured effects, e.g., when representing them
   in a planning algorithm.

Flat Effect

> ### Definition (Flat Effect)
> An effect is **simple** if it is either an atomic effect
> or of the form $(\chi \rhd e)$, where e is an atomic effect.
>
> An effect e is **flat** if it is a conjunction of 0 or more simple effects,
> and none of these simple effects include the same atomic effect.
>
> An operator o is **flat** if $eff(o)$ is flat.

Notes: analogously to CNF, we consider

- ▶ a single simple effect as a conjunction of 1 simple effect
- ▶ the empty effect as a conjunction of 0 simple effects

Flat Effect: Example

Example

Consider the effect

$$c \wedge (a \triangleright (\neg b \wedge (c \triangleright (b \wedge \neg d \wedge \neg a)))) \wedge (\neg b \triangleright \neg a)$$

An equivalent flat (and conflict-free) effect is

$$c \wedge$$
$$((a \wedge \neg c) \triangleright \neg b) \wedge$$
$$((a \wedge c) \triangleright b) \wedge$$
$$((a \wedge c) \triangleright \neg d) \wedge$$
$$((\neg b \vee (a \wedge c)) \triangleright \neg a)$$

Note: if we want, we can write c as $(\top \triangleright c)$ to make the structure even more uniform, with each simple effect having a condition.

Producing Flat Operators

Theorem
For every operator, an equivalent flat operator and an equivalent flat, conflict-free operator can be computed in polynomial time.

Producing Flat Operators: Proof

> ### Proof Sketch.
> Replace the effect e over variables V by
>
> $$\bigwedge_{v \in V}(\mathit{effcond}(v, e) \rhd v)$$
> $$\wedge \bigwedge_{v \in V}(\mathit{effcond}(\neg v, e) \rhd \neg v),$$
>
> which is an equivalent flat effect.
>
> To additionally obtain conflict-freeness, use
>
> $$\bigwedge_{v \in V}(\mathit{effcond}(v, e) \rhd v)$$
> $$\wedge \bigwedge_{v \in V}((\mathit{effcond}(\neg v, e) \wedge \neg \mathit{effcond}(v, e)) \rhd \neg v)$$
>
> instead.

(Conjuncts of the form $(\chi \rhd e)$ where $\chi \equiv \bot$
can be omitted to simplify the effect.)

B4.5 Summary

Summary

- ▶ **Equivalences** can be used to simplify operators and effects.
- ▶ In **conflict-free** operators, the "complicated case" of operator semantics does not arise.
- ▶ For **flat** operators, the only permitted nesting is atomic effects within conditional effects within conjunctive effects, and all atomic effects must be distinct.
- ▶ For flat, conflict-free operators, it is easy to determine the **condition** under which a given **literal** is **made true** by applying the operator in a given state.
- ▶ Every operator can be **transformed** into an equivalent **flat and conflict-free** one in **polynomial time**.