Planning and Optimization
A5. Equivalent Operators and Normal Forms

Malte Helmert and Thomas Keller

Universität Basel

September 25, 2019

Planning and Optimization
September 25, 2019 — A5. Equivalent Operators and Normal Forms

A5.1 Reminder & Motivation

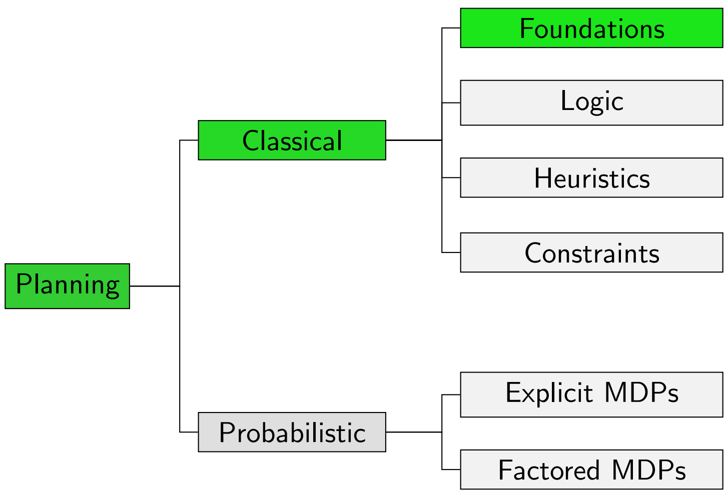
A5.2 Equivalence Transformations

A5.3 Conflict-Free Operators

A5.4 Flat Effects

A5.5 Summary

Content of this Course



A5.1 Reminder & Motivation

Reminder: Syntax of Effects

Definition (Effect)

Effects over state variables V are inductively defined as follows:

- If $v \in V$ is a propositional state variable, then v and $\neg v$ are effects (atomic effect).
- If $v \in V$ is a finite-domain state variable and $d \in \mathrm{dom}(v)$, then $v := d$ is an effect (atomic effect).
- If $e_1, \dots, e_n$ are effects, then $(e_1 \wedge \dots \wedge e_n)$ is an effect (conjunctive effect). The special case with $n = 0$ is the empty effect $\top$.
- 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!

$\rightsquigarrow$ Can we make our life easier?

Reminder: Semantics of Effects

- $\mathit{effcond}(e, e')$: condition that must be true in the current state for the effect e' to trigger the atomic effect e
- add-after-delete semantics (propositional tasks): if an operator with effect e is applied in state s and we have both $s \models \mathit{effcond}(v, e)$ and $s \models \mathit{effcond}(\neg v, e)$, then $s'(v) = \mathbf{T}$ in the resulting state s'.
- consistency semantics (finite-domain tasks): applying an operator with effect e in a state s where both $s \models \mathit{effcond}(v := d, e)$ and $s \models \mathit{effcond}(v := d', e)$ for values $d \neq d'$ is forbidden

  $\rightsquigarrow$ tested via the consistency condition $\mathit{consist}(e)$

These are very subtle details!

$\rightsquigarrow$ 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.

Notation: Applying Effects and Operator Sequences

Existing notation:

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

New extended notation:

- If we want to consider an effect e without a precondition, we write $s[\![e]\!]$ for $s[\![\langle \top, e \rangle]\!]$.
- For a sequence $\pi = \langle o_1, \dots, o_n \rangle$ of operators that are consecutively applicable in s, we write $s[\![\pi]\!]$ for $s[\![o_1]\!][\![o_2]\!] \dots [\![o_n]\!]$.

A5.2 Equivalence Transformations

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.

For consistency semantics, this includes the requirement that $s[\![e]\!]$ is defined iff $s[\![e']\!]$ is.

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

$$\begin{aligned}
e_1 \wedge e_2 &\equiv e_2 \wedge e_1 && (1)\\
(e_1 \wedge \dots \wedge e_n) \wedge (e'_1 \wedge \dots \wedge e'_m) &\equiv e_1 \wedge \dots \wedge e_n \wedge e'_1 \wedge \dots \wedge e'_m && (2)\\
\top \wedge e &\equiv e && (3)\\
\chi \rhd e &\equiv \chi' \rhd e \qquad \text{if } \chi \equiv \chi' && (4)\\
\top \rhd e &\equiv e && (5)\\
\bot \rhd e &\equiv \top && (6)\\
\chi_1 \rhd (\chi_2 \rhd e) &\equiv (\chi_1 \wedge \chi_2) \rhd e && (7)\\
\chi \rhd (e_1 \wedge \dots \wedge e_n) &\equiv (\chi \rhd e_1) \wedge \dots \wedge (\chi \rhd e_n) && (8)\\
(\chi_1 \rhd e) \wedge (\chi_2 \rhd e) &\equiv (\chi_1 \vee \chi_2) \rhd e && (9)
\end{aligned}$$

A5.3 Conflict-Free Operators

Conflict-Freeness: Motivation

- The add-after-delete semantics makes effects like $(a \rhd c) \wedge (b \rhd \neg c)$ somewhat unintuitive to interpret.
- $\rightsquigarrow$ What happens in states where $a \wedge b$ is true?
- Similarly, it may be unintuitive that an effect like $(u = \mathrm{a} \rhd w := \mathrm{a}) \wedge (v = \mathrm{b} \rhd w := \mathrm{b})$ introduces an applicability condition "through the back door"
- It would be nicer if
  - $effcond(e, e')$ always were the condition under which the atomic effect e actually materializes (because of add-after-delete, it is not)
  - $pre(o)$ always fully described the applicability of o (because of the consistency condition, is does not)
- $\rightsquigarrow$ introduce normal form where "complicated cases" never arise

Conflict-Free Effects and Operators

Definition (Conflict-Free)

An effect e over propositional state variables V is called conflict-free if $effcond(v, e) \wedge effcond(\neg v, e)$ is unsatisfiable for all $v \in V$.

An effect e over finite-domain state variables V is called conflict-free if $effcond(v := d, e) \wedge effcond(v := d', e)$ is unsatisfiable for all $v \in V$ and $d, d' \in \mathrm{dom}(v)$ with $d \neq d'$.

An operator o is called conflict-free if $eff(o)$ is conflict-free.

Note: This fixes both of our issues.
In particular, observe that $consist(o) \equiv \top$ for conflict-free o.

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 $\neg v$ by $(\neg effcond(v, eff(o)) \rhd \neg v)$
  - replace all atomic effects $v := d$ by $(consist(o) \rhd v := d)$
  - replace $pre(o)$ with $pre(o) \wedge consist(o)$ in the FDR case

  The resulting operator o' is conflict-free and $o \equiv o'$. (Why?)

A5.4 Flat Effects

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 e is flat if it is:

- a conjunctive effect
- whose conjuncts are conditional effects
- whose subeffects are atomic effects, and
- no atomic effect occurs in e multiple times.

An operator o is flat if $\mathit{eff}(o)$ is flat.

Note: non-conjunctive effects can be considered as conjunctive effects with 1 conjunct

Flat Effect: Example

Example

Consider the effect

$$c \land (a \rhd (\neg b \land (c \rhd (b \land \neg d \land \neg a)))) \land (\neg b \rhd \neg a)$$

An equivalent flat (and conflict-free) effect is

$$\begin{aligned} &(\top \rhd c) \land \\ &((a \land \neg c) \rhd \neg b) \land \\ &((a \land c) \rhd b) \land \\ &((a \land c) \rhd \neg d) \land \\ &((\neg b \lor (a \land c)) \rhd \neg a) \end{aligned}$$

Note: for simplicity, we often write $(\top \rhd e)$ as e, i.e., omit trivial effect conditions. We still consider such effects to be flat.

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.

Let E be the set of atomic effects over variables V.
Every effect e' over variables V is equivalent to $\bigwedge_{e \in E}(\mathit{effcond}(e, e') \rhd e)$, which is a flat effect.
(Conjuncts of the form $(\chi \rhd e)$ where $\chi \equiv \bot$ can be omitted to simplify the effect.)

To compute a flat operator equivalent to operator o, replace $\mathit{eff}(o)$ by an equivalent flat effect.

To compute an equivalent conflict-free and flat operator, first compute a conflict-free operator o' equivalent to o, then replace $\mathit{eff}(o')$ by an equivalent flat effect.
(Why not do these in the opposite order?)

A5.5 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.