

Planning and Optimization

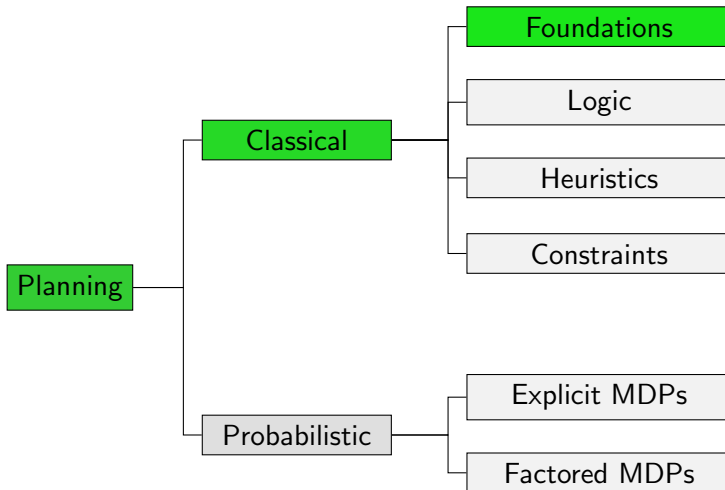
A6. Positive Normal Form, STRIPS and SAS⁺

Malte Helmert and Thomas Keller

Universität Basel

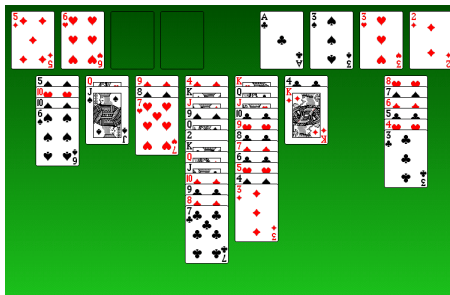
September 25, 2019

Content of this Course



Motivation

Example: Freecell



Example (Good and Bad Effects)

If we move $K\heartsuit$ to a free tableau position, the **good effect** is that $4\clubsuit$ is now accessible. The **bad effect** is that we lose one free tableau position.

What is a Good or Bad Effect?

Question: Which operator effects are good, and which are bad?

Difficult to answer in general, because it depends on context:

- Locking our door is **good** if we want to keep burglars out.
- Locking our door is **bad** if we want to enter.

We now consider a reformulation of propositional planning tasks that makes the distinction between good and bad effects obvious.

Positive Normal Form

Positive Formulas, Operators and Tasks

Definition (Positive Formula)

A logical formula φ is **positive** if no negation symbols appear in φ .

Note: This includes the negation symbols implied by $\rightarrow$ and $\leftrightarrow$.

Definition (Positive Operator)

An operator o is **positive** if $pre(o)$ and all effect conditions in $eff(o)$ are positive.

Definition (Positive Propositional Planning Task)

A propositional planning task $\langle V, I, O, \gamma \rangle$ is **positive** if all operators in O and the goal γ are positive.

Positive Normal Form

Definition (Positive Normal Form)

A propositional planning task is in **positive normal form** if it is positive and all operator effects are flat.

Positive Normal Form: Existence

Theorem (Positive Normal Form)

For every propositional planning task Π , there is an equivalent propositional planning task Π' in positive normal form. Moreover, Π' can be computed from Π in polynomial time.

Note: Equivalence here means that the transition systems induced by Π and Π' , **restricted to the reachable states**, are isomorphic.

We prove the theorem by describing a suitable algorithm.
(However, we do not prove its correctness or complexity.)

Positive Normal Form: Algorithm

Transformation of $\langle V, I, O, \gamma \rangle$ to Positive Normal Form

Replace all operators with equivalent conflict-free operators.

Convert all conditions to negation normal form (NNF).

while any condition contains a negative literal $\neg v$:

Let v be a variable which occurs negatively in a condition.

$V := V \cup \{\hat{v}\}$ for some new propositional state variable $\hat{v}$

$$I(\hat{v}) := \begin{cases} \mathbf{F} & \text{if } I(v) = \mathbf{T} \\ \mathbf{T} & \text{if } I(v) = \mathbf{F} \end{cases}$$

Replace the effect v by $(v \wedge \neg \hat{v})$ in all operators $o \in O$.

Replace the effect $\neg v$ by $(\neg v \wedge \hat{v})$ in all operators $o \in O$.

Replace $\neg v$ by $\hat{v}$ in all conditions.

Convert all operators $o \in O$ to flat operators.

Here, **all conditions** refers to all operator preconditions, operator effect conditions and the goal.

Positive Normal Form: Example

Example (Transformation to Positive Normal Form)

$$V = \{home, uni, lecture, bike, bike-locked\}$$

$$I = \{home \mapsto \mathbf{T}, bike \mapsto \mathbf{T}, bike-locked \mapsto \mathbf{T}, \\ uni \mapsto \mathbf{F}, lecture \mapsto \mathbf{F}\}$$

$$O = \{\langle home \wedge bike \wedge \neg bike-locked, \neg home \wedge uni \rangle, \\ \langle bike \wedge bike-locked, \neg bike-locked \rangle, \\ \langle bike \wedge \neg bike-locked, bike-locked \rangle, \\ \langle uni, lecture \wedge ((bike \wedge \neg bike-locked) \triangleright \neg bike) \rangle\}$$

$$\gamma = lecture \wedge bike$$

Positive Normal Form: Example

Example (Transformation to Positive Normal Form)

$$V = \{home, uni, lecture, bike, bike-locked\}$$

$$I = \{home \mapsto \mathbf{T}, bike \mapsto \mathbf{T}, bike-locked \mapsto \mathbf{T}, \\ uni \mapsto \mathbf{F}, lecture \mapsto \mathbf{F}\}$$

$$O = \{\langle home \wedge bike \wedge \neg bike-locked, \neg home \wedge uni \rangle, \\ \langle bike \wedge bike-locked, \neg bike-locked \rangle, \\ \langle bike \wedge \neg bike-locked, bike-locked \rangle, \\ \langle uni, lecture \wedge ((bike \wedge \neg bike-locked) \triangleright \neg bike) \rangle\}$$

$$\gamma = lecture \wedge bike$$

Identify state variable v occurring negatively in conditions.

Positive Normal Form: Example

Example (Transformation to Positive Normal Form)

$$V = \{home, uni, lecture, bike, bike\text{-}locked, \textcolor{red}{bike\text{-}unlocked}\}$$

$$I = \{home \mapsto \mathbf{T}, bike \mapsto \mathbf{T}, bike\text{-}locked \mapsto \mathbf{T}, \\ uni \mapsto \mathbf{F}, lecture \mapsto \mathbf{F}, \textcolor{red}{bike\text{-}unlocked} \mapsto \mathbf{F}\}$$

$$O = \{\langle home \wedge bike \wedge \neg bike\text{-}locked, \neg home \wedge uni \rangle, \\ \langle bike \wedge bike\text{-}locked, \neg bike\text{-}locked \rangle, \\ \langle bike \wedge \neg bike\text{-}locked, bike\text{-}locked \rangle, \\ \langle uni, lecture \wedge ((bike \wedge \neg bike\text{-}locked) \triangleright \neg bike) \rangle\}$$

$$\gamma = lecture \wedge bike$$

Introduce new variable $\hat{v}$ with complementary initial value.

Positive Normal Form: Example

Example (Transformation to Positive Normal Form)

$$V = \{home, uni, lecture, bike, bike-locked, bike-unlocked\}$$

$$I = \{home \mapsto \mathbf{T}, bike \mapsto \mathbf{T}, bike-locked \mapsto \mathbf{T}, \\ uni \mapsto \mathbf{F}, lecture \mapsto \mathbf{F}, bike-unlocked \mapsto \mathbf{F}\}$$

$$O = \{\langle home \wedge bike \wedge \neg bike-locked, \neg home \wedge uni \rangle, \\ \langle bike \wedge bike-locked, \neg bike-locked \rangle, \\ \langle bike \wedge \neg bike-locked, bike-locked \rangle, \\ \langle uni, lecture \wedge ((bike \wedge \neg bike-locked) \triangleright \neg bike) \rangle\}$$

$$\gamma = lecture \wedge bike$$

Identify effects on variable v .

Positive Normal Form: Example

Example (Transformation to Positive Normal Form)

$$V = \{home, uni, lecture, bike, bike-locked, bike-unlocked\}$$

$$I = \{home \mapsto \mathbf{T}, bike \mapsto \mathbf{T}, bike-locked \mapsto \mathbf{T},$$

$$uni \mapsto \mathbf{F}, lecture \mapsto \mathbf{F}, bike-unlocked \mapsto \mathbf{F}\}$$

$$O = \{\langle home \wedge bike \wedge \neg bike-locked, \neg home \wedge uni \rangle,$$

$$\langle bike \wedge bike-locked, \neg bike-locked \wedge bike-unlocked \rangle,$$

$$\langle bike \wedge \neg bike-locked, bike-locked \wedge \neg bike-unlocked \rangle,$$

$$\langle uni, lecture \wedge ((bike \wedge \neg bike-locked) \triangleright \neg bike) \rangle\}$$

$$\gamma = lecture \wedge bike$$

Introduce complementary effects for $\hat{v}$.

Positive Normal Form: Example

Example (Transformation to Positive Normal Form)

$$V = \{home, uni, lecture, bike, bike-locked, bike-unlocked\}$$

$$I = \{home \mapsto \mathbf{T}, bike \mapsto \mathbf{T}, bike-locked \mapsto \mathbf{T},$$

$$uni \mapsto \mathbf{F}, lecture \mapsto \mathbf{F}, bike-unlocked \mapsto \mathbf{F}\}$$

$$O = \{\langle home \wedge bike \wedge \neg bike-locked, \neg home \wedge uni \rangle,$$

$$\langle bike \wedge bike-locked, \neg bike-locked \wedge bike-unlocked \rangle,$$

$$\langle bike \wedge \neg bike-locked, bike-locked \wedge \neg bike-unlocked \rangle,$$

$$\langle uni, lecture \wedge ((bike \wedge \neg bike-locked) \triangleright \neg bike) \rangle\}$$

$$\gamma = lecture \wedge bike$$

Identify negative conditions for v .

Positive Normal Form: Example

Example (Transformation to Positive Normal Form)

$$V = \{home, uni, lecture, bike, bike\text{-}locked, bike\text{-}unlocked\}$$

$$I = \{home \mapsto \mathbf{T}, bike \mapsto \mathbf{T}, bike\text{-}locked \mapsto \mathbf{T}, \\ uni \mapsto \mathbf{F}, lecture \mapsto \mathbf{F}, bike\text{-}unlocked \mapsto \mathbf{F}\}$$

$$O = \{\langle home \wedge bike \wedge \textcolor{red}{bike\text{-}unlocked}, \neg home \wedge uni \rangle, \\ \langle bike \wedge bike\text{-}locked, \neg bike\text{-}locked \wedge bike\text{-}unlocked \rangle, \\ \langle bike \wedge \textcolor{red}{bike\text{-}unlocked}, bike\text{-}locked \wedge \neg bike\text{-}unlocked \rangle, \\ \langle uni, lecture \wedge ((bike \wedge \textcolor{red}{bike\text{-}unlocked}) \triangleright \neg bike) \rangle\}$$

$$\gamma = lecture \wedge bike$$

Replace by positive condition $\hat{v}$.

Positive Normal Form: Example

Example (Transformation to Positive Normal Form)

$$V = \{home, uni, lecture, bike, bike-locked, bike-unlocked\}$$

$$I = \{home \mapsto \mathbf{T}, bike \mapsto \mathbf{T}, bike-locked \mapsto \mathbf{T},$$

$$uni \mapsto \mathbf{F}, lecture \mapsto \mathbf{F}, bike-unlocked \mapsto \mathbf{F}\}$$

$$O = \{\langle home \wedge bike \wedge bike-unlocked, \neg home \wedge uni \rangle,$$

$$\langle bike \wedge bike-locked, \neg bike-locked \wedge bike-unlocked \rangle,$$

$$\langle bike \wedge bike-unlocked, bike-locked \wedge \neg bike-unlocked \rangle,$$

$$\langle uni, lecture \wedge ((bike \wedge bike-unlocked) \triangleright \neg bike) \rangle\}$$

$$\gamma = lecture \wedge bike$$

Why Positive Normal Form is Interesting

In positive normal form, good and bad effects are easy to distinguish:

- Effects that make state variables true
(**add effects**) are good.
- Effects that make state variables false
(**delete effects**) are bad.

This is particularly useful for planning algorithms based on **delete relaxation**, which we will study later in this course.

STRIPS

STRIPS Operators and Planning Tasks

Definition (STRIPS Operator)

An operator o of a prop. planning task is a **STRIPS operator** if

- $pre(o)$ is a conjunction of state variables, and
- $eff(o)$ is a conflict-free conjunction of atomic effects.

Definition (STRIPS Planning Task)

A propositional planning task $\langle V, O, I, \gamma \rangle$ is a **STRIPS planning task** if all operators $o \in O$ are STRIPS operators and γ is a conjunction of state variables.

Note: STRIPS operators are conflict-free and flat.

STRIPS is a special case of positive normal form.

STRIPS Operators: Remarks

- Every STRIPS operator is of the form

$$\langle v_1 \wedge \cdots \wedge v_n, \ell_1 \wedge \cdots \wedge \ell_m \rangle$$

where v_i are state variables and ℓ_j are atomic effects.

- Often, STRIPS operators o are described via three **sets of state variables**:
 - the **preconditions** (state variables occurring in $pre(o)$)
 - the **add effects** (state variables occurring positively in $eff(o)$)
 - the **delete effects** (state variables occurring negatively in $eff(o)$)
- Definitions of STRIPS in the literature often do **not** require conflict-freeness. But it is easy to achieve and makes many things simpler.
- There exists a variant called **STRIPS with negation** where negative literals are also allowed in conditions.

Why STRIPS is Interesting

- STRIPS is **particularly simple**, yet expressive enough to capture general planning tasks.
- In particular, STRIPS planning is **no easier** than planning in general (as we will see next week).
- Many algorithms in the planning literature are **only presented for STRIPS planning tasks** (generalization is often, but not always, obvious).

STRIPS

STanford Research Institute Problem Solver
(Fikes & Nilsson, 1971)

Transformation to STRIPS

- Not every operator is equivalent to a STRIPS operator.
- However, each operator can be transformed into a **set** of STRIPS operators whose “combination” is equivalent to the original operator. (How?)
- However, this transformation may exponentially increase the number of operators. There are planning tasks for which such a blow-up is unavoidable.
- There are polynomial transformations of propositional planning tasks to STRIPS, but these do not lead to isomorphic transition systems (auxiliary states are needed). (They are, however, equivalent in a weaker sense.)

SAS⁺

SAS⁺ Operators and Planning Tasks

Definition (SAS⁺ Operator)

An operator o of an FDR planning task is a **SAS⁺ operator** if

- $pre(o)$ is a satisfiable conjunction of atoms, and
- $eff(o)$ is a conflict-free conjunction of atomic effects.

Definition (SAS⁺ Planning Task)

An FDR planning task $\langle V, O, I, \gamma \rangle$ is a **SAS⁺ planning task** if all operators $o \in O$ are SAS⁺ operators and γ is a satisfiable conjunction of atoms.

Note: SAS⁺ operators are conflict-free and flat.

SAS⁺ Operators: Remarks

- Every SAS⁺ operator is of the form

$$\langle v_1 = d_1 \wedge \dots \wedge v_n = d_n, \quad v'_1 := d'_1 \wedge \dots \wedge v'_m := d'_m \rangle$$

where all v_i are distinct and all v'_j are distinct.

- Often, SAS⁺ operators o are described via two **sets of partial assignments**:
 - the **preconditions** $\{v_1 \mapsto d_1, \dots, v_n \mapsto d_n\}$
 - the **effects** $\{v'_1 \mapsto d'_1, \dots, v'_m \mapsto d'_m\}$

SAS⁺ vs. STRIPS

- SAS⁺ is an analogue of STRIPS planning tasks for FDR, but there is no special role of “positive” conditions.
- Apart from this difference, all comments for STRIPS apply analogously.
- If all variable domains are binary, SAS⁺ is essentially STRIPS with negation.

SAS⁺

Derives from SAS = Simplified Action Structures
(Bäckström & Klein, 1991)

Summary

Summary

- A **positive** task allows distinguishing good and bad effects.
The notion of positive tasks only exists for **propositional** tasks.
- A positive task with flat operators is in **positive normal form**.
- **STRIPS** is even more restrictive than positive normal form, forbidding complex preconditions and conditional effects.
- Both forms are expressive enough to capture general propositional planning tasks.
- Transformation to positive normal form is possible with polynomial size increase.
- Isomorphic transformations of propositional planning tasks to STRIPS can increase the number of operators exponentially; non-isomorphic polynomial transformations exist.
- **SAS⁺** is the analogue of STRIPS for FDR planning tasks.