Computational Complexity and Expressive Power in Classical AI Planning

Part III: Compilability and Expressiveness

Bernhard Nebel

Albert-Ludwigs-Universität Freiburg
Department of Computer Science

Propositional STRIPS and Variants

- In the following we will only consider **propositional STRIPS** and some variants of it.
- No variables, i.e. no parameters in operator schemata, i.e., they are already operators.
- Set of propositional atoms denoted by Σ (not $\mathcal{S}$), i.e., **planning task**:

$$\Pi = \langle \Sigma, \mathrm{O}, \mathrm{I}, \mathrm{G} \rangle.$$

- Often we refer to **domain structures** $\Xi = \langle \Sigma, \mathrm{O} \rangle$.

Disjunctive Preconditions: Trivial or Essential?

- Kambhampati et al [ECP 97] and Gazen & Knoblock [ECP 97]: Disjunctive preconditions are trivial – since they can be translated to basic STRIPS (DNF-preconditions)
- Bäckström [AIJ 95]: Disjunctive preconditions are probably essential – since they can not easily be translated to basic STRIPS (CNF-preconditions)
- Anderson et al [AIPS 98]: "[D]isjunctive preconditions . . . are . . . essential prerequisites for handling conditional effects" $\leadsto$ conditional effects imply disjunctive preconditions (?) (General Boolean preconditions)

More "Expressive Power"

STRIPS : plain STRIPS (we will not consider)

STRIPS$_N$: negative literals in preconditions

STRIPS$_{Bd}$: precondition in disjunctive normal form

STRIPS$_{Bc}$: precondition in conjunctive normal form

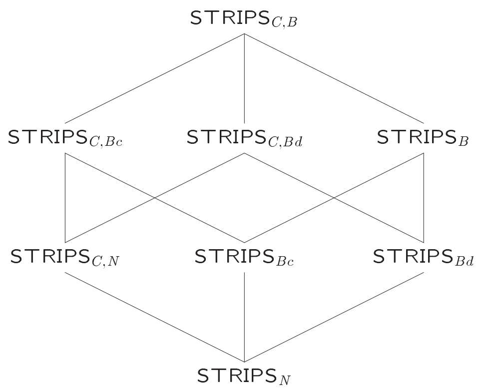
STRIPS$_B$: Boolean expressions as preconditions

STRIPS$_C$: conditional effects (push-accelerator)

STRIPS$_{C,N}$: conditional effects & negative literals

. . .

Ordering Planning Formalisms Partially



Computational Complexity ...

Theorem. PLANEX is PSPACE-complete for STRIPS_N, $\text{STRIPS}_{C,B}$, and for all formalisms "between" the two.

Proof. PSPACE-hardness follows from Bylander's theorem (we showed in the previous lecture). Membership for $\text{STRIPS}_{C,B}$ follows with the same same argument as in the proof of Bylanders theorem.

[Membership in NPSPACE because one can successively guess operators and compute the resulting state (needing only poly. space) – until a goal state is non-deterministically reached (verification in poly time)]

Measuring Expressive Power

Consider **mappings** between planning problems in different formalisms

- that **preserve**
  - solution existence
  - plan size linearly or polynomially etc.
  - the exact plan size
  - the plan "structure"
  - the solutions/plans themselves
- that **are limited**
  - in the *size* of the result (poly. size)
  - in the *computational resources* (poly. time)
- that **transform**
  - entire planning instances
  - domain structure and states in isolation

Method 1: Polynomial Transformation

- **preserving**
  - $\boxed{\text{solution existence}}$
  - plan size linearly or polynomially etc.
  - the exact plan size
  - the plan "structure"
  - the solutions/plans themselves
- **limiting**
  - in the *size* of the result (poly. size)
  - $\boxed{\text{in the } \textit{computational resources} \text{ (poly. time)}}$
- **transforming**
  - $\boxed{\text{entire planning instances}}$
  - domain structure and states in isolation

$\rightsquigarrow$ all formalisms have the **same expressiveness** (?)

Method 2: Bäckström's ESP-reductions

- **preserving**
  - solution existence
  - plan size linearly or polynomially etc.
  - the exact plan size
  - the plan "structure"
  - the solutions/plans themselves
- **limiting**
  - in the *size* of the result (poly. size)
  - in the *computational resources* (poly. time)
- **transforming**
  - entire planning instances
  - domain structure and states in isolation

$\rightsquigarrow$ Expressiveness is independent of the **computational resources** needed to compute the mapping

Method 3: Polysize Mappings

- **preserving**
  - solution existence
  - plan size linearly or polynomially etc.
  - the exact plan size
  - the plan "structure"
  - the solutions/plans themselves
- **limiting**
  - in the *size* of the result (poly. size)
  - in the *computational resources* (poly. time)
- **transforming**
  - entire planning instances
  - domain structure and states in isolation

$\rightsquigarrow$ All formalisms are **trivially equivalent** (because planning is in PSPACE)

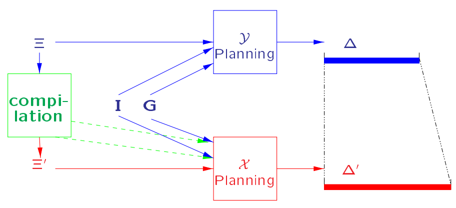
Method 4: Modular & Polysize Mappings

- **preserving**
  - solution existence
  - plan size linearly or polynomially etc.
  - the exact plan size
  - the plan "structure"
  - the solutions/plans themselves
- **limiting**
  - in the *size* of the result (poly. size)
  - in the *computational resources* (poly. time)
- **transforming**
  - entire planning instances
  - domain structure and states in isolation

$\rightsquigarrow$ When measuring the expressiveness of **planning formalisms**, domain structures should be considered independently from states

The Right Method: Compilation Schemes (Simplified)

- Transform **domain structure** $\Xi = \langle \Sigma, \mathrm{O} \rangle$ (with polynomial blowup) to Ξ' preserving solution existence
- Only trivial changes to **states** (independent of operator set)
- Resulting **plans** Δ' should not grow too much (additive constant, linear growth, polynomial growth)



$\rightsquigarrow$ Similar to **knowledge compilation**, with operators as the *fixed part* and initial states & goals as the *varying part*

Compilability

$\mathcal{Y} \preceq \mathcal{X}$ ($\mathcal{Y}$ is compilable to $\mathcal{X}$)
iff
there exists a compilation scheme from $\mathcal{Y}$ to $\mathcal{X}$.

$\mathcal{Y} \preceq^1 \mathcal{X}$: preserving plan size **exactly** (modulo additive constants)

$\mathcal{Y} \preceq^c \mathcal{X}$: preserving plan size **linearly** (in $|\Delta|$)

$\mathcal{Y} \preceq^p \mathcal{X}$: preserving plan size **polynomially** (in $|\Delta|$ and $|\Xi|$)

$\mathcal{Y} \preceq^x_p \mathcal{X}$: **polynomial-time** compilability

Proposition. For all x, y, the relations $\preceq^x_y$ are transitive and reflexive.

A (Trivial) Positive Result: STRIPS$_{Bd}$ $\preceq^1_p$ STRIPS$_N$

DNF preconditions can be **"compiled away."**

Assume operator o with postcondition $post(o)$ and precondition

$$pre(o) = L_1 \vee \ldots \vee L_k$$

with L_i being a conjunction of literals. Create k operators o_i:

$$pre(o_i) = L_i, post(o_i) = post(o).$$

1. compilation is solution-preserving,
2. Ξ' is only polynomially larger than Ξ,
3. compilation can be computed in polynomial time,
4. resulting plans do not grow at all.

$\leadsto$ STRIPS$_d$ $\preceq^1_p$ STRIPS$_n$

Another Positive Result: STRIPS$_{C,Bc}$ $\preceq^c_p$ STRIPS$_{C,N}$

CNF preconditions can be **"compiled away"** – provided we have already conditional effects.

- Evaluate the truth value of all disjunctions appearing in operators by using a **special evaluation operator** with conditional effects.
- Alternate between executing original operators and evaluation operators

$\leadsto$ Operator sets grow only **polynomially**

$\leadsto$ Plans are **double as long** as the original plans

$\leadsto$ **Anderson et al's conjecture** holds in a weak version

A First Negative Results: Conditional Effects Cannot be Compiled into Boolean Preconditions

Consider the STRIPS$_{C,N}$-Operator o:

$$pre(o) = \top,\ post(o) = \{p_1 \Rightarrow \neg p_1, \neg p_1 \Rightarrow p_1, \ldots, p_k \Rightarrow \neg p_k, \neg p_k \Rightarrow p_k\},$$

which "inverts" a given state. Let this be the only operator in Ξ. For all $(\mathbf{I}, \mathbf{G})$, $\mathbf{G} = \Sigma - \mathbf{I}$, there exists a STRIPS$_{C,N}$ one-step plan.

Assume there exists a compilation preserving plan size linearly leading to a STRIPS$_B$ domain structure Ξ'. There are exponentially many $(\mathbf{I}, \Sigma - \mathbf{I})$ pairs, but only polynomially many different c-step plans for Ξ'. Some STRIPS$_B$ plan Δ is used for different initial states (for large enough k).

$\leadsto$ Since Δ is unconditional, it must lead to the same goal.
$\leadsto$ The transformation is **not solution preserving**!

$\leadsto$ **Conditional effects** cannot be compiled away

Another Negative Result: STRIPS$_{Bc}$ $\not\preceq^c$ STRIPS$_N$

k-**FISEX**: Planning problem with fixed plan length k and varying initial state. Does there exists an initial state leading to a successful k-step plan?

1-FISEX is NP-complete for STRIPS$_{Bc}$ (= SAT).

k-FISEX is polynomial for STRIPS$_N$ (regression analysis)

$\boxed{\leadsto \text{STRIPS}_{Bc} \not\preceq^c_p \text{STRIPS}_N \text{ (if P}\neq\text{NP)}}$

Using a technique first used by Kautz & Selman, one can show that even arbitrary compilations can be ruled out – provided the polynomial hierarchy does not collapse.

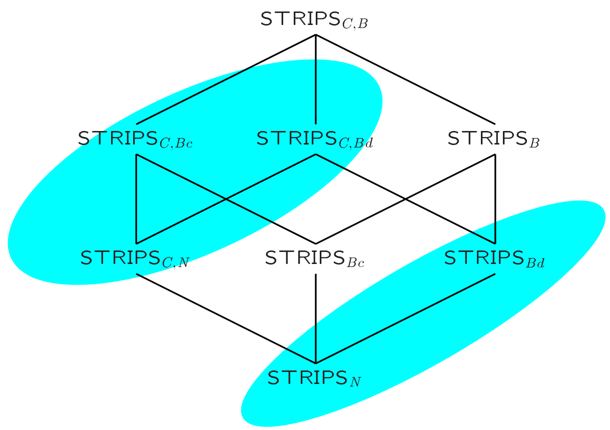
$\leadsto$ **Bäckström's conjecture holds** in the compilation framework.

A Final Negative Result: Boolean Preconditions Cannot be Compiled Away Even in the presence of Conditional Effects

- Boolean preconditions have the power of **families of Boolean circuits with logarithmic (because Boolean formula have) depth**
- Conditional effects can simulate only **families of circuits with fixed depth**
- The parity function can be expressed in the first framework (NC^1) while it cannot be expressed in the second (AC^0).

$\leadsto$ The negative result follows unconditionally!

General Results for Compilability Preserving Plan Size Linearly



All other potential positive results have been ruled out by our 3 negative results and transitivity.

Summary

- Compilation schemes seem to be the right method to measure the *relative* expressive power of planning formalisms
- Either we get a positive result preserving plan size **linearly** with a **polynomial-time compilation**
- or we get an **impossibility result**

$\rightarrow$ **Results are relevant for building planning systems**

$\leadsto$ **CNF preconditions** do not add much when we have already conditional effects