Motivation: Why Analyzing the Expressive Power?

- **Expressive power** is the motivation for designing new planning languages.
- Often there is the question: *Syntactic sugar* or *essential feature*?

→ **Compiling away** or change planning algorithm?

→ If a feature can be compiled away, then it is apparently only *syntactic sugar*.

→ Sometimes, however, a compilation can lead to much larger planning domain descriptions or to much longer plans.

→ This means the planning algorithm will probably choke, i.e., it cannot be considered as a *compilation*.

Example: DNF Preconditions

- Assume we have **DNF preconditions** in STRIPS operators.
- This can be compiled away as follows.
- **Split** each operator with a DNF precondition $c_1 \lor \ldots \lor c_n$ into $n$ operators with the same effects and $c_i$ as preconditions.

→ If there exists a plan for the original planning task there is one for the new planning task and vice versa.

→ The planning task has almost the same size.

→ The shortest plans have the same size.
Example: Conditional effects

► Can we compile away conditional effects to STRIPS?
► Example operator: \(\langle a, b \triangleright d \land \neg c \triangleright e \rangle\)
► Can be translated into four operators:
\(\langle a \land b \land c, d \rangle, \langle a \land b \land \neg c, d \land e \rangle, \ldots\)
► Plan existence and plan size are identical
► Exponential blowup of domain description!
➔ Can this be avoided?

Propositional STRIPS and Variants

Propositional STRIPS and Variants

► In the following we will only consider propositional STRIPS and some variants of it.
► Planning task:
\(T = \langle A, I, O, G \rangle.\)

More “Expressive Power”

STRIPS\(_N\) : plain strips with negative literals
STRIPS\(_{Ed}\) : precondition in disjunctive normal form
STRIPS\(_{Bc}\) : precondition in conjunctive normal form
STRIPS\(_{B}\) : Boolean expressions as preconditions
STRIPS\(_C\) : conditional effects
STRIPS\(_{C,N}\) : conditional effects & negative literals

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” \(\sim\) conditional effects imply disjunctive preconditions (?) (General Boolean preconditions)
Ordering Planning Formalisms Partially

\[ \text{STRIPS}_{B,C} \]
\[ \text{STRIPS}_{B_c,C} \quad \text{STRIPS}_{B,d,C} \quad \text{STRIPS}_B \]
\[ \text{STRIPS}_{N,C} \quad \text{STRIPS}_{B_c} \quad \text{STRIPS}_{B_d} \]
\[ \text{STRIPS}_N \]

Computational Complexity

Theorem
PLANEX is \textit{PSPACE}-complete for STRIPS\(_N\), STRIPS\(_{C,B}\), and for all formalisms “between” the two.

Proof.
Follows from theorems proved in the previous lecture.

Measuring Expressive Power

Consider \textit{mappings} between planning problems in different formalisms

- \textit{preserving}
  - solution existence
  - plan size linearly or polynomially etc.
  - the exact plan size
  - the plan “structure”
  - the solutions/plans themselves

- \textit{limiting}
  - in the size of the result (poly. size)
  - in the \textit{computational resources} (poly. time)

- \textit{transforming}
  - \textit{entire planning instances}
  - domain structure and states in isolation

\[ \text{all formalisms have the \textit{same expressiveness}} \quad (?) \]
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

⇝ However, **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

⇝ All formalisms are **trivially equivalent** (because planning is PSPACE-complete for all propositional STRIPS formalisms)

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

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

The Right Method: Compilation Schemes (Simplified)

- **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

⇝ Similar to **knowledge compilation**, with operators as the **fixed part** and initial states & goals as the **varying part**
Compilability

\( Y \preceq X \) (\( Y \) is compilable to \( X \))

iff there exists a compilation scheme from \( Y \) to \( X \).

\( Y \preceq^1 X \): preserving plan size exactly (modulo additive constants)
\( Y \preceq^c X \): preserving plan size linearly (in \(|\pi|\))
\( Y \preceq^p X \): preserving plan size polynomially (in \(|\pi|\) and \(|D|\))
\( Y \preceq^x_p X \): polynomial-time compilability

Theorem
For all \( x, y \), the relations \( \preceq^x_y \) are transitive and reflexive.

Positive Results

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

DNF preconditions can be “compiled away.”
Assume operator \( o = \langle c, e \rangle \) and

\[ c = L_1 \lor \ldots \lor L_k \]

with \( L_i \) being a conjunction of literals. Create \( k \) operators \( o_i = \langle L_i, e \rangle \)

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

\( \Rightarrow \) \( \text{STRIPS}_{Bd} \preceq^1_p \text{STRIPS}_N \)

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

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 that make new “clause atoms” true
- Alternate between executing original operators (clauses replaced by new atoms) and evaluation operators

\( \Rightarrow \) Operator sets grow only polynomially
\( \Rightarrow \) Plans are double as long as the original plans

\( \Rightarrow \) Anderson et al’s conjecture holds in a weak version
A First Negative Result: Conditional Effects Cannot be Compiled into Boolean Preconditions

Consider domain $D$ with only one (STRIPS$_{C,B}$) operator $\circ$:

$$(\top, (p_1 \leftarrow \neg p_1) \land (\neg p_1 \leftarrow p_1) \land \ldots \land (p_k \leftarrow \neg p_k) \land (\neg p_k \leftarrow p_k)),$$

which "inverts" a given state. For all $(I, G)$ with

$$G = \bigwedge \{ \neg v \mid v \in A, I \models v \} \land \bigwedge \{ v \mid v \in A, I \not\models v \},$$

there exists a STRIPS$_{C,B}$ one-step plan. Assume there exists a compilation preserving plan size linearly leading to a STRIPS$_B$ domain structure $D'$.

Some STRIPS$_{C,B}$ plan $\pi$ is used for different initial states $I_1, I_2$ (for large enough $k$). Let $v$ be a variable with $I_1(v) \neq I_2(v)$.

$\Rightarrow$ In one case, $v$ must be set by $\pi$, in the other case, it must be cleared.

$\Rightarrow$ This is not possible in an unconditional plan.

$\Rightarrow$ The transformation is not solution preserving!

$\Rightarrow$ Conditional effects cannot be compiled away (if plan size can grow only linearly)

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 exist 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)

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

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. The proof method uses non-uniform complexity classes such as $P/poly$.

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

$\Rightarrow$ The negative result follows unconditionally!
The classes $AC^k$

- The classes $NC^k$ are defined with a fixed fan-in
- If we have *unbounded fan-in*, we get the classes $AC^k$
  - gate types: NOT, $n$-ary AND, $n$-ary OR for all $n \geq 2$
- Obviously: $NC^k \subseteq AC^k$
- Possible to show: $AC^{k-1} \subseteq NC^k$
- The *parity language* is in $NC^1$, but not in $AC^0$!

Accepting languages with families of domain structures with fixed goals

- We will view *families of domain structures* with fixed goals and fixed size plans as "machines" that accept languages
- Consider families of poly-sized domain structures in $STRIPS_B$ and use one-step plans for acceptance.
- Obviously, this is the same as using Boolean formulae
  $\Rightarrow$ All languages in $NC^1$ can be accepted in this way

Simulating $STRIPS_{C,N}$ $c$-Step Plans with $AC^0$ circuits (1)

- Represent each operator and then chain the actions together ($O(|O|^c)$ different plans):

```
\[\begin{array}{c}
\land \\
\neg \\
\vdots \\
\neg \\
\lor \\
\neg \\
\vdots \\
\neg \\
p_1^c \\
p_2^c \\
\cdots \\
p_n^c \\
p_1^0 \\
p_2^0 \\
\cdots \\
p_n^0 \\
F_1 \\
\cdots \\
F_m \\
\end{array}\]
```

Simulating $STRIPS_{C,N}$ $c$-Step Plans with $AC^0$ circuits (2)

- For each single action (precondition testing (a), conditional effects (b), and the computation of effects (c)
Theorem
STRIPS\(_B \not\leq^c \) STRIPS\(_{C,N}\).

Proof.
Assuming STRIPS\(_B \leq^c \) STRIPS\(_{C,N}\) has the consequence that the underlying compilation scheme could be used to compile a NC\(^1\) circuit family into an AC\(^0\) circuit family, which is impossible in the general case.

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