What is the Expressive Power of Disjunctive Preconditions?

Bernhard Nebel
Institut für Informatik, Albert-Ludwigs-Universität
Freiburg, Germany

Institut für Informatik – Technical Report 118

March 3, 1999

Abstract

While there seems to be a general consensus about the expressive power of a number of language features in planning formalisms, one can find many different statements about the expressive power of disjunctive preconditions. Using the "compilability framework," we show that preconditions in conjunctive normal form add to the expressive power of propositional STRIPS, which confirms a conjecture by Bäckström. Further, we show that preconditions in conjunctive normal form do not add any expressive power once we have conditional effects.

Contents

1 Introduction

There seems to be a general consensus about the *expressive power* of a number of language features in planning formalisms. For example, everybody seems to agree that adding negative preconditions does not add very much to the expressive power of basic STRIPS, while conditional effects are considered as a significant increase in expressive power [Anderson *et al.*, 1998; Gazen and Knoblock, 1997; Kambhampati *et al.*, 1997; Koehler *et al.*, 1997].

For *disjunctive preconditions*, however, the picture is much less clear. One can find many different statements about the expressive power of disjunctive preconditions. For instance, Anderson *et al.* [1998] write that "disjunctive preconditions ... are ... essential prerequisites for handling conditional effects." This statement could be understood as stating that implementing disjunctive preconditions would make it easier to handle conditional effects. However, it also could be read as stating that once we are able to handle conditional effects, disjunctive preconditions come for free. Bäckström [1995] conjectures that "disjunctive preconditions most likely increase the expressive power [of STRIPS]." Kambhampati *et al.* [1997] and Gazen and Knoblock [1997], finally, believe that disjunctive preconditions are easy to add to basic STRIPS.

On top of these different conjectures there is no consensus on what the term *disjunctive precondition* means. Bäckström [1995] interprets this as preconditions in conjunctive normal form (CNF). Gazen and Knoblock [1997] interpret it as preconditions in disjunctive normal form (DNF), and Anderson *et al.* [1998] mean general Boolean formulae. This means that instead of asking for the expressive power of disjunctive preconditions (relative to basic STRIPS or STRIPS with conditional effects), we should ask for the expressive power of CNF-, DNF-, and general Boolean preconditions.

In order to address this problem we use the *compilability framework* [Nebel, 1998], which is inspired by the work of Gazen and Knoblock [1997] and by the *knowledge compilation* framework [Cadoli and Donini, 1999]. The main idea behind this approach is that a language feature does not increase the expressiveness if planning operators containing this language feature can be translated into operators that do not contain this feature without blowing up the operator description too much and without enlarging the resulting plans too much.

Using this approach, we show that Bäckström's conjecture that CNF preconditions are more expressive than basic STRIPS is indeed correct. We also prove that CNF preconditions do not add to the expressive power if we have already conditional effects – proving a weak version of Anderson's *et al.* [1998] conjecture. However, general boolean formulae as preconditions are incomparable to conditional effects as follows from earlier results [Nebel, 1998].

Although these results are purely theoretical, they also have some significance for designing planning algorithms. Provided we can prove that a language feature can be "compiled away" easily, i.e., that it can be regarded as "syntactic sugar," a planning algorithm for the original language can be easily extended to deal with the new feature. Conversely, if we can prove that a language feature cannot be compiled away, we most probably will have significant problems when we want to extend the planning algorithm for the original language to deal with the new language feature.

The rest of the paper is structured as follows. The next section introduces general terminology and definitions. In Section 3, we introduce the notion of compilability between planning formalisms. Using this framework, we analyze the relationship between STRIPS with disjunctive preconditions and basic STRIPS in Section 4. We show that DNF preconditions can be compiled away and that Bäckström's [1995] conjecture holds in the compilability framework. In Section 5 we show that CNF preconditions can be compiled away if we have conditional effects. Finally, in Section 6 we summarize and discuss the results.

2 Propositional Planning Formalisms with Disjunctive Preconditions and Conditional Effects

Let Σ be a finite set of **propositional atoms** and $\hat{\Sigma}$ be the set consisting of the constants $\top$ (denoting truth) and $\bot$ (denoting falsity) as well as atoms and negated atoms, i.e., the **literals**, over Σ. The set of all **Boolean formulae** over Σ is denoted by $\mathbf{B}_\Sigma$. The set of **formulae in conjunctive normal form (CNF)** over Σ is denoted by $\mathbf{C}_\Sigma$ and the set of **formulae in disjunctive normal form (DNF)** over Σ by $\mathbf{D}_\Sigma$. Finally, by $\mathbf{L}_\Sigma$ we refer to the set of **formulae that are conjunctions of literals** over Σ, and the set of **formulae that are conjunctions of atoms** is denoted by $\mathbf{A}_\Sigma$. In general, we will use the symbol $\mathcal{L}_\Sigma$ to refer to a possibly restricted language over Σ.

Given a set of literals $L \subseteq \hat{\Sigma}$, by $pos(L)$ we refer to the **positive literals** in L, by $neg(L)$ we refer to the **negative literals** in L, and $\neg L$ denotes the **element-wise negation** of the literals in L. **Operators** are pairs $o = \langle pre, post \rangle$. We use the notation $pre(o)$ and $post(o)$ to refer to the first and second part of an operator o, respectively. The **precondition** pre is an element of $\mathcal{L}_\Sigma$. The set $post$, the set of **postconditions**, consists of **conditional effects**, each having the form $\varphi \Rightarrow L$, where $\varphi \in \mathcal{L}_\Sigma$ is called **effect condition** and the elements of $L \subseteq \hat{\Sigma}$ are called **effects**. If all postconditions of an operator have the form $\top \Rightarrow L$, then we say that the operator is **unconditional** and we write the postconditions as a set of literals containing all *effects*.

A **state** S is a *truth-assignment* for the atoms in Σ, which is represented by the set of atoms that are true in this state. By $\bot$ we represent the **illegal state**. Given a state S and an operator o, we define the **active effects** $A(S, o)$ as follows:

$$A(S,o) \quad = \quad \bigcup \{L \mid (\varphi \Rightarrow L) \in post(o), S \models \varphi\}.$$

Using this function, the result of executing operator o in state S can be specified as:

$$R(S,o) = \begin{cases} S - neg(A(S,o)) \cup pos(A(S,o)) & \text{if } S \neq \bot \text{ and} \\ & \quad S \models pre(o) \text{ and} \\ & \quad A(S,o) \not\models \bot, \\ \bot & \text{otherwise} \end{cases}$$

A **planning instance** is now a tuple $\Pi = \langle \Xi, \mathbf{I}, \mathbf{G} \rangle$, where

- $\Xi = \langle \Sigma, \mathbf{O} \rangle$ is the **domain structure** consisting of a finite set of propositional atoms Σ and a finite set of operators $\mathbf{O}$,
- $\mathbf{I} \subseteq \Sigma$ is the **initial state**, and
- $\mathbf{G} \subseteq \widehat{\Sigma}$ is the **goal specification**.

When we talk about the **size of an instance**, symbolically $||\Pi||$, in the following, we mean the size of a (reasonable) encoding of the instance.

Let Δ be an element of $\mathbf{O}^\star$, i.e., a finite sequence of operators, which is called **plan**. Then $||\Delta||$ denotes the size of the plan, i.e., the number of operators in Δ. We say that Δ is a ***c*-step plan** if $||\Delta|| \leq c$. The result of applying Δ to a state S is recursively defined as follows:

$$\begin{aligned} Res(S, \langle\rangle) &= S, \\ Res(S, \langle o_1, o_2, \ldots, o_n \rangle) &= Res(R(S, o_1), \langle o_2, \ldots, o_n \rangle). \end{aligned}$$

A sequence of operators Δ is said to be a **plan for** Π or a **solution of** Π iff

1. $Res(\mathbf{I}, \Delta) \neq \bot$ and
2. $Res(\mathbf{I}, \Delta) \models \mathbf{G}$.

The most general planning language we consider in this paper is $\text{STRIPS}_{\mathcal{C},\mathbf{B}}$, which permits conditional effects and general Boolean formulae in preconditions and effect conditions. Without the index $\mathcal{C}$, we refer to planning languages without conditional effects, i.e., all conditional effects have the form $\top \Rightarrow L$. If we have $\mathbf{C}$, $\mathbf{D}$ or $\mathbf{L}$ instead of $\mathbf{B}$, we refer to languages that permit only for CNF or DNF formulae or conjunctions of literals in preconditions and effect conditions, respectively. The language STRIPS (without any index), finally, is identical to basic STRIPS, i.e., it requires that all preconditions are conjunctions of atoms and all effects are unconditional. In this paper, however, we assume that all formulae may contain literals, i.e., the least expressive language we consider is $\text{STRIPS}_{\mathbf{L}}$.[1]

[1]The reason for leaving out basic STRIPS is that it has already be shown that STRIPS and $\text{STRIPS}_{\mathbf{L}}$ as well as $\text{STRIPS}_{\mathcal{C}}$ and $\text{STRIPS}_{\mathcal{C},\mathbf{L}}$ are equivalent with respect to expressiveness [Nebel, 1998, Theorem 6 and 9]. Furthermore, ignoring the case $\mathcal{L}_\Sigma = \mathbf{A}_\Sigma$ simplifies some of the technical problems when specifying compilation schemes.

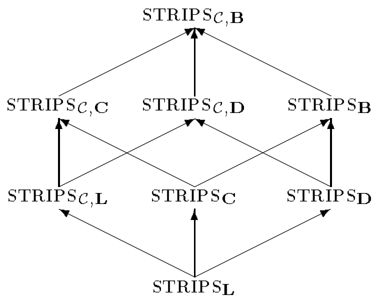

Figure 1: Partial order induced by syntactic restrictions

In Figure 1, the partial order induced by the syntactic restrictions is shown. In the sequel we say that $\mathcal{X}$ **is a specialization** of $\mathcal{Y}$, written $\mathcal{X} \sqsubseteq \mathcal{Y}$, iff $\mathcal{X}$ is identical to $\mathcal{Y}$ or below $\mathcal{Y}$ in the diagram depicting the partial order.

While one would expect that planning in $\text{STRIPS}_{\mathbf{L}}$ is much easier than planning in $\text{STRIPS}_{\mathcal{C},\mathbf{B}}$, it turns out that this is not the case, provided one takes a computational complexity perspective. The plan existence problem in all the formalisms we consider is PSPACE-complete [Nebel, 1998, Theorem 3].

3 Compilation Schemes

We will consider a planning formalism $\mathcal{X}$ *as expressive as* another formalism $\mathcal{Y}$ if planning domains and plans formulated in formalism $\mathcal{Y}$ are *concisely expressible* in $\mathcal{X}$. We formalize this intuition by making use of what we call *compilation schemes*, which are *solution preserving mappings* with *polynomially sized results* from $\mathcal{Y}$ domain structures to $\mathcal{X}$ domain structures. While we restrict the size of the result of a compilation scheme, we do not require any bounds on the computational resources for the compilation. In fact, for measuring the *expressibility*, it is irrelevant whether the mapping is polynomial-time computable, exponential-time computable, or even non-recursive. If we want to use such compilation schemes in practice, they should be reasonably efficient, of course. However, if we want to prove that one formalism is *not as least as expressive as* another one, we better prove that there is no compilation scheme regardless of how many computational resources it might use. For this reason, Bäckström's [1995] ESP-reductions could not be classified as compilation schemes because they are constrained to be computable in polynomial time.

So far, compilation schemes restrict only the size of domain structures. However, when measuring expressive power, the size of the generated plans should also play a role. In Bäckström's ESP-reductions [1995], the plan size must be identical. Similarly, the translation from $\text{STRIPS}_{\mathcal{C},\mathbf{L}}$ to STRIPS proposed by Gazen

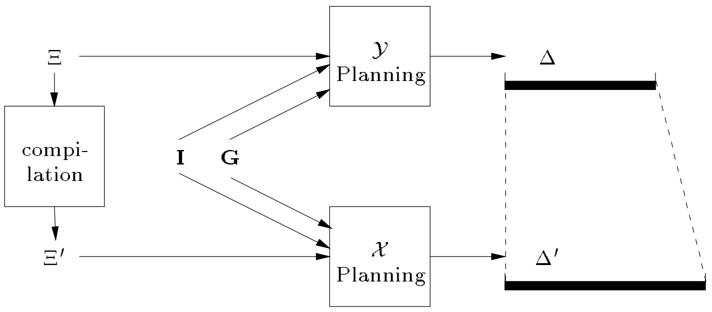

Figure 2: The compilation framework

and Knoblock [1997] guarantees that the plan length does not change. When comparing the expressiveness of different planning formalisms, we might, however, be prepared to accept some growth of the plans in the target formalism. For instance, we may accept an additional constant number of operators, or we may even be satisfied if the plan in the target formalism is linearly or polynomially larger. This leads to the schematic picture of compilation schemes as displayed in Figure 2.

Although Figure 2 gives a good picture of the *compilation framework*, it is not completely accurate. A compilation scheme may introduce some auxiliary propositional atoms that are used to control the execution of newly introduced operators. These atoms should most probably have an initial value and may appear in the goal specification of planning instances in the target formalism. We will assume that the compilation scheme takes care of this and adds some literals to the initial state and goal specifications.

Additionally, some translations of the initial state and goal specifications may be necessary. If we want to compile a formalism that permits for literals in preconditions and goals to one that requires atoms, some trivial translations are necessary. Similarly, if we want to compile a formalism that permits to use partial states to a formalism that requires complete state, a translation of the initial state specification is necessary. However, since we do not deal with incomplete state specifications or planning formalisms that allow only atoms in the precondition, we can ignore this issue here.[2]

A **compilation scheme from $\mathcal{X}$ to $\mathcal{Y}$** is a tuple of functions $\mathbf{f} = \langle f_\xi, f_i, f_g \rangle$ that induce a function F from $\mathcal{X}$-instances $\Pi = \langle \Xi, \mathbf{I}, \mathbf{G} \rangle$ to $\mathcal{Y}$-instances $F(\Pi)$ as follows:

$$F(\Pi) = \langle f_\xi(\Xi), \mathbf{I} \cup f_i(\Xi), \mathbf{G} \cup f_g(\Xi) \rangle,$$

and satisfies the following conditions:

[2] This means, we do not need the *state-translation functions* as introduced in the definition of a compilation scheme in an earlier paper [Nebel, 1998].

1. there exists a plan for Π iff there exists a plan for $F(\Pi)$,

2. and the size of the results of f_ξ, f_i, and f_g is polynomial in the size of the arguments.

Although there are no resource bounds on f_ξ, f_i, and f_g in the general case, we are also interested in *efficient compilation schemes*. We say that $\mathbf{f}$ is a **polynomial-time compilation scheme** if f_ξ, f_i, and f_g are polynomial-time computable functions.

In addition to that we measure the size of the corresponding plans in the target formalism. If a compilation scheme $\mathbf{f}$ has the property that for every plan Δ solving an instance Π there exists a plan Δ' solving $F(\Pi)$ such that $||\Delta'|| \leq ||\Delta|| + k$ for some positive integer constant k, $\mathbf{f}$ is a **compilation scheme preserving plan size exactly** (modulo an additive constant). If $||\Delta'|| \leq c \times ||\Delta|| + k$ for positive integer constants c and k, then $\mathbf{f}$ is a **compilation scheme preserving plan size linearly**, and if $||\Delta'|| \leq p(||\Delta||, ||\Pi||)$ for some polynomial p, then $\mathbf{f}$ is a **compilation scheme preserving plan size polynomially**. More generally, we say that a planning formalism $\mathcal{X}$ is **compilable** to formalism $\mathcal{Y}$ (in poly. time, preserving plan size exactly, linearly, or polynomially), if there exists a compilation scheme with the appropriate properties. We write $\mathcal{X} \preceq^x \mathcal{Y}$ in case $\mathcal{X}$ is compilable to $\mathcal{Y}$ or $\mathcal{X} \preceq^x_p \mathcal{Y}$ if the compilation can be done in polynomial time. The super-script x can be 1, c, or p depending on whether the scheme preserves plan size exactly, linearly, or polynomially, respectively.

From a practical point of view, one could regard compilability preserving plan size exactly or linearly as an indication that the planning formalism we use as the target formalism is *at least as expressive* as the source formalism. Conversely, if a super-linear (be it polynomial or super-polynomial) blowup of the plans in the target formalism is required by any compilation scheme, this is an indication that the source formalism *is more expressive* than the target formalism.

As is easy to see, all the notions of compilability introduced above are reflexive and transitive, i.e., compilability induces a pre-order on planning formalisms.

Proposition 1 [Nebel, 1998] *The relations $\preceq^x$ and $\preceq^x_p$ are transitive and reflexive.*

Furthermore, it is obvious that when moving upwards in the diagram displayed in Figure 1, there is always a polynomial-time compilation scheme preserving plan size exactly. If π_i denotes the projection to the i-th argument and $\emptyset$ the function that returns always the empty set, the generic compilation scheme for moving upwards in the partial order is $\mathbf{f} = \langle \pi_1, \emptyset, \emptyset \rangle$.

Proposition 2 [Nebel, 1998] *If $\mathcal{X} \sqsubseteq \mathcal{Y}$, then $\mathcal{X} \preceq^1_p \mathcal{Y}$.*

Using (a slight generalization of) the *compilation framework* described above, it has been shown that all formalisms that we consider in this paper can be compiled to each other in polynomial time preserving plan size polynomially [Nebel, 1998, Corollary 24]. However, if only linear growth of the plan in the target formalism is permitted, then for most of the formalisms there does not exist a compilation scheme. In particular, we have the following two results.

Theorem 3 [Nebel, 1998] $\text{STRIPS}_{\mathcal{C},\mathbf{L}} \not\preceq^c \text{STRIPS}_{\mathbf{B}}$.

Theorem 4 [Nebel, 1998] $\text{STRIPS}_{\mathbf{B}} \not\preceq^c \text{STRIPS}_{\mathcal{C},\mathbf{L}}$.

These results seem to invalidate the conjecture by Anderson *et al.* [1998] that the expressiveness of conditional effects and disjunctive preconditions in form of general boolean formulae are related. However, as we will see below, restricting the preconditions to be in CNF, a weak form of their conjecture is provable in the compilation framework.

4 Disjunctive Preconditions

Although the relative expressiveness of different planning formalisms has been extensively studied [Nebel, 1998], syntactically restricted formulae such as CNF and DNF formulae have not been considered yet.

It is a folklore in the planning community that DNF preconditions can be regarded as syntactic sugar. For each operator of the form

$$o = \langle (c_1 \vee c_2 \vee \ldots \vee c_k), L \rangle,$$

where $c_i \in \mathbf{L}_\Sigma$ and $L \subseteq \widehat{\Sigma}$, one simply generates k new operators o_i:

$$o_i = \langle c_i, L \rangle,$$

which are obviously $\text{STRIPS}_{\mathbf{L}}$ operators. Furthermore, such a translation requires obviously only polynomial time and it is solution preserving. Finally, it does not change the plan length. In other words, this translation is a *polynomial-time compilation scheme preserving plan size exactly.*

Proposition 5 $\text{STRIPS}_{\mathbf{D}} \preceq^1_p \text{STRIPS}_{\mathbf{L}}$.

For CNF preconditions the situation is much less clear. As mentioned above, Bäckström [1995] conjectured that CNF preconditions add to the expressiveness of $\text{STRIPS}_{\mathbf{L}}$, but he was not able to prove this conjecture using his framework of ESP-reductions. Using our compilability framework for measuring expressiveness, we can, however, prove his conjecture. In order to do so, we first introduce a variation of the planning problem.

The **fixed plan-size initial-state existence problem** (c-FISEX) is defined as follows. Given a domain structure $\Xi = \langle \Sigma, \mathbf{O} \rangle$, a goal $\mathbf{G} \subseteq \hat{\Sigma}$, a state $\mathbf{I} \subseteq \Sigma$, and a subset of the atoms called *choice set* $\mathbf{C} \subseteq \Sigma$, the question is whether there exists a set $C \subseteq \mathbf{C}$ such that $\langle \Xi, \mathbf{I} \cup C, \mathbf{G} \rangle$ can be solved by a plan with size c, where c is a positive constant.

Although this problem appears to be slightly harder than the ordinary plan existence problem, fixing the plan length to the positive constant c makes the problem easy, at least for the planning formalism $\text{STRIPS}_{\mathbf{L}}$.

Theorem 6 $\text{STRIPS}_{\mathbf{L}}$-$c$-FISEX *can be decided in polynomial time.*

Proof. Given an $\text{STRIPS}_{\mathbf{L}}$-$c$-FISEX instance $(\Xi, \mathbf{I}, \mathbf{G}, \mathbf{C})$, the number of possible operator sequences is $O(|\mathbf{O}|^c)$, which is polynomial in the instance size. For each such sequence, we can do a regression analysis starting with the goal $\mathbf{G}$ computing the weakest precondition, which is a set of literals. This can be done in polynomial time. Finally, one can easily check in polynomial time, whether there is a subset $C \subseteq \mathbf{C}$ that leads to an initial state $\mathbf{I} \cup C$ that entails the weakest precondition. This means, the problem can be solved in polynomial time for any fixed c. ■

If we allow for CNF preconditions, the problem becomes harder. Even if the plan length is restricted to one, 3SAT can be obviously reduced to the 1-FISEX problem in $\text{STRIPS}_{\mathbf{C}}$.

Proposition 7 $\text{STRIPS}_{\mathbf{C}}$-*1*-FISEX *is* NP-*complete.*

From that it follows immediately that there cannot exist any *polynomial-time* compilation scheme from $\text{STRIPS}_{\mathbf{C}}$ to $\text{STRIPS}_{\mathbf{L}}$ preserving plan size linearly—provided $\mathsf{P} \neq \mathsf{NP}$. We will show a stronger result, namely that there cannot exist *any* compilation scheme preserving plan size linearly by employing a proof technique first used by Kautz and Selman [Kautz and Selman, 1992].

In order to prove this result, we need the notion of *advice-taking Turing machines*. These are machines with an **advice oracle**, which is a (not necessarily recursive) function a from positive integers to bit strings. On input I, the machine loads the bit string $a(||I||)$ and then continues as usual. Note that the oracle derives its bit string only from the length of the input and not from the contents of the input. An advice is said to be a **polynomial advice** if the oracle string is polynomially bounded by the instance size. Further, if X is a complexity class defined in terms of resource-bounded machines, e.g., P or NP, then X/poly (also called **non-uniform** X) is the class of problems that can be decided on machines with the same resource bounds and polynomial advice.

Because of the advice oracle, the class P/poly appears to be much more powerful than P. However, it seems unlikely that P/poly contains all of NP. In fact, one can prove that $\mathsf{NP} \subseteq \mathsf{P/poly}$ implies certain relationships between uniform complexity classes that are believed to be very unlikely. In particular, Karp and Lipton [1980] have shown that $\mathsf{NP} \subseteq \mathsf{P/poly}$ implies that the polynomial hierarchy collapses on the second level—which is considered to be very unlikely.

Theorem 8 $\textsc{strips}_{\mathbf{C}} \not\preceq^c \textsc{strips}_{\mathbf{L}}$, *unless the polynomial hierarchy collapses.*

Proof. As a first step, we construct for each n a STRIPS domain structure Ξ_n and goal specification $\mathbf{G}_n$ with size polynomial in n and the following properties. Satisfiability of an arbitrary 3CNF formula φ_n of size n is equivalent to 1-step plan existence for the $\textsc{strips}_{\mathbf{C}}$-1-FISEX instance $(\Xi_n, \mathbf{G}_n, \mathbf{I}_{\varphi_n}, \mathbf{C}_n)$, where $\mathbf{I}_{\varphi_n}$ can be computed in polynomial time from φ_n.

Given a set of n atoms, denoted by $\mathbf{P}_n$, we define the set of clauses $\mathbf{A}_n$ to be the set containing all clauses with three literals that can be built using these atoms. The size of $\mathbf{A}_n$ is $O(n^3)$, i.e., polynomial in n. Let $\mathbf{D}_n$ be new atoms p_γ corresponding one-to-one to the clauses γ in $\mathbf{A}_n$. Finally, let s be a new atom which is not in $\mathbf{P}_n \cup \mathbf{D}_n$.

Now we construct a $\textsc{strips}_{\mathbf{C}}$ domain structure $\Xi_n = \langle \Sigma_n, \mathbf{O}_n \rangle$ goal $\mathbf{G}_n$, and the choice set $\mathbf{C}_n$ as follows:

$$\begin{aligned}
\Sigma_n &= \mathbf{P}_n \cup \mathbf{D}_n \cup \{\mathsf{s}\}, \\
\mathbf{O}_n &= \{o_n\}, \\
o_n &= \langle (\bigwedge_{\gamma \in \mathbf{A}_n} (p_\gamma \vee \gamma)), \{\mathsf{s}\} \rangle, \\
\mathbf{G}_n &= \{\mathsf{s}\}, \\
\mathbf{C}_n &= \mathbf{P}_n.
\end{aligned}$$

Let $cl(\varphi)$ be the set of clauses appearing in φ. Based on that we define $\mathbf{I}_{\varphi_n}$ as follows:

$$\mathbf{I}_{\varphi_n} = \{p_\gamma \in \mathbf{D}_n \mid \gamma \notin cl(\varphi_n)\}.$$

Now it is easy to see that φ_n is satisfiable iff for the $\textsc{strips}_{\mathbf{C}}$-1-FISEX instance $(\Xi_n, \mathbf{G}_n, \mathbf{I}_{\varphi_n}, \mathbf{C}_n)$ there exists a set of choices $C \subseteq \mathbf{C}_n$ such that the resulting planning instance $\langle \Xi_n, \mathbf{I}_{\varphi_n} \cup C, \mathbf{G}_n \rangle$ is solved by a one-step plan.

Let us now assume that there exists a compilation scheme from $\textsc{strips}_{\mathbf{C}}$ to $\textsc{strips}_{\mathbf{L}}$ preserving plan size linearly. Using this compilation scheme, we compile the $\textsc{strips}_{\mathbf{C}}$ domain structure Ξ_n into the $\textsc{strips}_{\mathbf{L}}$ domain structure $\Xi'_n = \langle \Sigma'_n, \mathbf{O}'_n \rangle$. Further, $\mathbf{G}'_n$, $\mathbf{I}'_{\varphi_n}$, and $\mathbf{C}'_n$ are defined as follows:

$$\begin{aligned}
\mathbf{G}'_n &= \{\mathsf{s}\} \cup f_g(\Sigma_n, \mathbf{O}_n), \\
\mathbf{I}'_{\varphi_n} &= \mathbf{I}_{\varphi_n} \cup f_i(\Sigma_n, \mathbf{O}_n), \\
\mathbf{C}'_n &= \mathbf{C}_n.
\end{aligned}$$

Note that all these sets can be computed in time polynomial in n, once we know the values of $f_g(\Sigma_n, \mathbf{O}_n)$ and $f_i(\Sigma_n, \mathbf{O}_n)$.

From the construction, it follows that the following statements are equivalent:

1. φ_n is satisfiable,

2. for the $\text{STRIPS}_{\mathbf{L}}$-1-FISEX instance $(\Xi_n, \mathbf{G}_n, \mathbf{I}_{\varphi_n}, \mathbf{C}_n)$, there exists a set of choices $C \subseteq \mathbf{C}_n$ such that $\Pi = \langle \Sigma_n, \mathbf{O}_n, \mathbf{I}_{\varphi_n} \cup C, \mathbf{G}_n \rangle$ has a one-step plan,

3. for the $\text{STRIPS}_{\mathbf{L}}$-$c$-FISEX instance $(\Xi'_n, \mathbf{G}'_n, \mathbf{I}'_{\varphi_n}, \mathbf{C}'_n)$, there exists a set of choices $C' \subseteq \mathbf{C}'_n$ such that $\Pi' = \langle \Sigma'_n, \mathbf{O}'_n, \mathbf{I}'_{\varphi_n} \cup C', \mathbf{G}'_n \rangle$ has a c-step plan,

One can now construct an advice-taking Turing machine that on input of a formula φ_n of size n loads the polynomial advice $\langle \Xi'_n, f_g(\Sigma_n, \mathbf{O}_n), f_i(\Sigma_n, \mathbf{O}_n) \rangle$ and then decides $\text{STRIPS}_{\mathbf{L}}$-$c$-FISEX for the instance $(\Xi'_n, \mathbf{G}'_n, \mathbf{I}'_{\varphi_n}, \mathbf{C}'_n)$, which by Theorem 6 can be done in polynomial time. Since the problem 3SAT, which is solved by this deterministic, advice-taking machine in polynomial time is NP-complete, we conclude that NP $\subseteq$ P/poly. This implies by Karp and Lipton's [1980] result that the polynomial hierarchy collapses on the second level, which proves the claim. ∎

The result above implies that adding CNF preconditions to $\text{STRIPS}_{\mathbf{L}}$ adds to the expressiveness. However, it is not immediately obvious whether a further generalization from CNF formulae to arbitrary boolean formulae would add another level of expressiveness. We will defer this question to the next section.

5 Compiling Disjunctive Preconditions into Conditional Effects

As mentioned in the Introduction, sometimes the expressive power of conditional effects and of disjunctive preconditions are claimed to be related. In this section, we will analyze this claim using the compilability framework.

As in the case of unconditional actions, it is commonly agreed that DNF formulae can be regarded as "syntactic sugar." Any operator containing a DNF precondition with k disjuncts can be split into k new operators containing only conjunctions of literals in the precondition. Similarly, any conditional effect with a DNF effect condition

$$c_1 \vee \ldots \vee c_n \Rightarrow L$$

can be equivalently expressed by a set of n conditional effects

$$c_i \Rightarrow L.$$

Obviously, this transformation can be viewed as a polynomial-time compilation scheme preserving plan size exactly.

Proposition 9 $\text{STRIPS}_{\mathcal{C},\mathbf{D}} \preceq^1_p \text{STRIPS}_{\mathcal{C},\mathbf{L}}$.

Interestingly, CNF preconditions and effect conditions do not appear to add to the expressive power once we have conditional effects—provided we accept that two formalisms have the same expressive power, if they are compilable to each other preserving plan size *linearly*. The main idea behind proving this is that operators with conditional effects can be used to evaluate the truth of clauses.

Theorem 10 $\text{STRIPS}_{\mathcal{C},\mathbf{C}} \preceq_p^c \text{STRIPS}_{\mathcal{C},\mathbf{L}}$.

Proof. Assume that the operators of the $\text{STRIPS}_{\mathcal{C},\mathbf{C}}$ domain structure $\Xi = \langle \Sigma, \mathbf{O} \rangle$ contain n clauses $\gamma_1, \gamma_2, \ldots, \gamma_n$ with $\gamma_i = l_{i1} \vee \ldots \vee l_{ik_i}$ in preconditions and effect conditions. For each clause γ_i, a new atom p_{γ_i} is introduced, and the set of these new atoms is denoted by Γ. Now, the operator EVAL, which will evaluate the truth values of all the clauses in a given state, can be defined as follows:

$$\text{EVAL} = \langle \top, \{l_{ij} \Rightarrow p_{\gamma_i}\} \rangle.$$

If all clauses γ_i in $\mathbf{O}$ are replaced by the new atoms p_{γ_i}—leading to the new set $\widehat{\mathbf{O}}$—the only remaining changes that are necessary are that we enforce that the EVAL operator is always executed before an operator from $\widehat{\mathbf{O}}$ is executed and that all operators from $\widehat{\mathbf{O}}$ set all the atoms from Γ to false.

In order to enforce sequences of operators alternating between operators from $\widehat{\mathbf{O}}$ and the EVAL-operator, one can introduce a new atom e that is added to the initial state. In addition, we modify the EVAL operator and all operators $\hat{o} \in \widehat{\mathbf{O}}$ as follows:

$$\begin{array}{rcl} \text{EVAL}' & = & \langle \mathsf{e},\ \mathit{post}(\text{EVAL}) \cup \{\top \Rightarrow \{\neg \mathsf{e}\}\} \rangle \\ o' & = & \langle \neg \mathsf{e} \wedge \mathit{pre}(\hat{o}),\ \mathit{post}(\hat{o}) \cup \{\top \Rightarrow \{\mathsf{e}\}\} \cup \{\top \Rightarrow \neg \Gamma\} \rangle. \end{array}$$

We can now specify a compilation scheme from $\text{STRIPS}_{\mathcal{C},\mathbf{C}}$ to $\text{STRIPS}_{\mathcal{C},\mathbf{L}}$ as follows:

$$\begin{array}{rcl} f_\xi\colon\ \langle \Sigma, \mathbf{O} \rangle & \mapsto & \langle \Sigma \cup \Gamma \cup \{\mathsf{e}\}, \{o' | \hat{o} \in \widehat{\mathbf{O}}\} \cup \{\text{EVAL}'\} \rangle, \\ f_i\colon\ \langle \Sigma, \mathbf{O} \rangle & \mapsto & \{\mathsf{e}\}, \\ f_g\colon\ \langle \Sigma, \mathbf{O} \rangle & \mapsto & \emptyset. \end{array}$$

This is obviously a polynomial-time compilation scheme that leads to $\text{STRIPS}_{\mathcal{C},\mathbf{L}}$ plans that are twice as long as the original $\text{STRIPS}_{\mathcal{C},\mathbf{C}}$ plans. ∎

This result appears to be relevant for practical planning algorithms because it suggests how to extend planning algorithms for conditional operators to algorithms for dealing with CNF preconditions and effect conditions. However, one may wonder whether we can improve on this result, coming up with a compilation scheme preserving plan size exactly. Interestingly, there does not appear to be an obvious way to do that. Further, it is completely unclear how to prove that such a compilation scheme is impossible.

Using the above result and Theorem 4, we can now easily give an answer to the question posed in the end of the previous section, namely, whether general boolean preconditions are more expressive than CNF preconditions.

Theorem 11 $\text{STRIPS}_{\mathbf{B}} \not\preceq^c \text{STRIPS}_{\mathbf{C}}$.

Proof. Assume for contradiction that there is a compilation scheme from $\text{STRIPS}_{\mathbf{B}}$ to $\text{STRIPS}_{\mathbf{C}}$ preserving plan size linearly. Since by Proposition 2 we have $\text{STRIPS}_{\mathbf{C}}$ $\preceq^c$ $\text{STRIPS}_{\mathcal{C},\mathbf{C}}$ and by Theorem 10 we have $\text{STRIPS}_{\mathcal{C},\mathbf{C}} \preceq^c \text{STRIPS}_{\mathcal{C},\mathbf{L}}$, we can conclude $\text{STRIPS}_{\mathbf{B}} \preceq^c \text{STRIPS}_{\mathcal{C},\mathbf{L}}$ using Proposition 1 twice. This, however, contradicts Theorem 4. ■

This leaves us with the question whether general boolean preconditions and effect conditions are more expressive than CNF preconditions and effect conditions. However, assuming that $\text{STRIPS}_{\mathcal{C},\mathbf{B}} \preceq^c \text{STRIPS}_{\mathcal{C},\mathbf{C}}$ leads immediately to the conclusion that $\text{STRIPS}_{\mathcal{C},\mathbf{B}} \preceq^c \text{STRIPS}_{\mathcal{C},\mathbf{L}}$ (using Theorem 10 and Proposition 1), which is impossible because of Theorem 4.

Proposition 12 $\text{STRIPS}_{\mathcal{C},\mathbf{B}} \not\preceq^c \text{STRIPS}_{\mathcal{C},\mathbf{C}}$.

6 Summary and Discussion

Using the *compilability framework* [Nebel, 1998], we analyzed the expressive power of disjunctive preconditions and conditional effects. In general, our results provide a complete classification of the relative expressiveness of STRIPS-like languages with restricted formulae and conditional effects – provided that literals are always allowed and states are always complete. Table 1 gives an overview of the results. The "$\sqsubseteq$" entries mark syntactic specialization relationships (see Figure 1). For

$\preceq^x$	$\text{STRIPS}_{\mathbf{L}}$	$\text{STRIPS}_{\mathbf{D}}$	$\text{STRIPS}_{\mathbf{C}}$	$\text{STRIPS}_{\mathbf{B}}$	$\text{STRIPS}_{\mathcal{C},\mathbf{L}}$	$\text{STRIPS}_{\mathcal{C},\mathbf{D}}$	$\text{STRIPS}_{\mathcal{C},\mathbf{C}}$	$\text{STRIPS}_{\mathcal{C},\mathbf{B}}$
$\text{STRIPS}_{\mathbf{L}}$	$=$	$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$
$\text{STRIPS}_{\mathbf{D}}$	$\preceq_p^1$ **(5)**	$=$	$\preceq_p^1$ (5)	$\sqsubseteq$	$\preceq_p^1$ (5)	$\sqsubseteq$	$\preceq_p^1$ (5)	$\sqsubseteq$
$\text{STRIPS}_{\mathbf{C}}$	$\not\preceq^c$ **(8)**	$\not\preceq^c$ (8,5)	$=$	$\sqsubseteq$	$\preceq_p^c$ (10)	$\preceq_p^c$ (10)	$\sqsubseteq$	$\sqsubseteq$
$\text{STRIPS}_{\mathbf{B}}$	$\not\preceq^c$ (11)	$\not\preceq^c$ (11,5)	$\not\preceq^c$ **(11)**	$=$	$\not\preceq^c$ **(4)**	$\not\preceq^c$ (4,9)	$\not\preceq^c$ (4,10)	$\sqsubseteq$
$\text{STRIPS}_{\mathcal{C},\mathbf{L}}$	$\not\preceq^c$ (3)	$\not\preceq^c$ (3)	$\not\preceq^c$ (3)	$\not\preceq^c$ **(3)**	$=$	$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$
$\text{STRIPS}_{\mathcal{C},\mathbf{D}}$	$\not\preceq^c$ (3)	$\not\preceq^c$ (3)	$\not\preceq^c$ (3)	$\not\preceq^c$ (3)	$\preceq_p^1$ **(9)**	$=$	$\preceq_p^1$ (9)	$\sqsubseteq$
$\text{STRIPS}_{\mathcal{C},\mathbf{C}}$	$\not\preceq^c$ (3)	$\not\preceq^c$ (3)	$\not\preceq^c$ (3)	$\not\preceq^c$ (3)	$\preceq_p^c$ **(10)**	$\preceq_p^c$ (10)	$=$	$\sqsubseteq$
$\text{STRIPS}_{\mathcal{C},\mathbf{B}}$	$\not\preceq^c$ (3)	$\not\preceq^c$ (3)	$\not\preceq^c$ (3)	$\not\preceq^c$ (3)	$\not\preceq^c$ (4)	$\not\preceq^c$ (4,9)	$\not\preceq^c$ **(12)**	$=$

Table 1: Results

all other entries we give the strongest compilation result or impossibility result. The number indicates the theorem from which the result has been derived. If the number is in bold face, it is just the statement of the theorem. Otherwise, the result can be derived from the theorem and the application of Propositions 1 and 2.

Two particular interesting results are that

1. CNF preconditions add to the power of basic STRIPS, confirming an earlier conjecture by Bäckström [1995], and that

2. CNF preconditions and effect conditions do not add anything to the expressive power if we already have conditional effects, confirming a weak version of a conjecture by Anderson *et al.* [1998].

In particular the latter result may have practical value for the design of planning algorithms. It suggests that when normalizing preconditions and effect conditions it is not necessary to convert them to disjunctive normal form, but conjunctive normal form is another option that can be easily dealt with. This option may sometimes help to avoid excessive space consumption, provided the formulae are already almost CNF.

Acknowledgments

The research reported in this paper was started and partly carried out while the author enjoyed being a visitor at the AI department of the University of New South Wales. Many thanks go to Norman Foo, Maurice Pagnucco, and Abhaya Nayak and the rest of the AI department for the discussions and cappuccinos.

References

[Anderson *et al.*, 1998] Corin R. Anderson, David E. Smith, and Daniel S. Weld. Conditional effects in graphplan. In *Proceedings of the 4th International Conference on Artificial Intelligence Planning Systems (AIPS-98)*. AAAI Press, Menlo Park, 1998.

[Bäckström, 1995] Christer Bäckström. Expressive equivalence of planning formalisms. *Artificial Intelligence*, 76(1–2):17–34, 1995.

[Cadoli and Donini, 1999] Marco Cadoli and Francesco M. Donini. A survey on knowledge compilation. *AI Communications*, 1999. To appear.

[Gazen and Knoblock, 1997] B. Cenk Gazen and Craig Knoblock. Combining the expressiveness of UCPOP with the efficiency of Graphplan. In Steel and Alami [1997], pages 221–233.

[Kambhampati *et al.*, 1997] Subbarao Kambhampati, Eric Parker, and Eric Lambrecht. Understanding and extending Graphplan. In Steel and Alami [1997], pages 260–272.

[Karp and Lipton, 1980] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Conference Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing (STOC-80)*, pages 302–309, Los Angeles, California, 28–30 April 1980.

[Kautz and Selman, 1992] Henry A. Kautz and Bart Selman. Forming concepts for fast inference. In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence (AAAI-92)*, pages 786–793, San Jose, CA, July 1992. MIT Press.

[Koehler *et al.*, 1997] Jana Koehler, Bernhard Nebel, Jörg Hoffmann, and Yannis Dimopoulos. Extending planning graphs to an ADL subset. In Steel and Alami [1997], pages 273–285.

[Nebel, 1998] Bernhard Nebel. On the compilability and expressive power of propositional planning formalisms. Technical Report 101, Albert-Ludwigs-Universität, Institut für Informatik, Freiburg, Germany, June 1998.

[Steel and Alami, 1997] S. Steel and R. Alami, editors. *Recent Advances in AI Planning. 4th European Conference on Planning (ECP'97)*, volume 1348 of *Lecture Notes in Artificial Intelligence*, Toulouse, France, September 1997. Springer-Verlag.