Expressive Equivalence of Planning Formalisms

Christer Bäckström[1]

Department of Computer and Information Science, Linköping University, S-581 83 Linköping, Sweden, email: cba@ida.liu.se

Abstract

A concept of expressive equivalence for planning formalisms based on polynomial transformations is defined. It is argued that this definition is reasonable and useful both from a theoretical and from a practical perspective; if two languages are equivalent, then theoretical results carry over and, more practically, we can model an application problem in one language and then easily use a planner for the other language. In order to cope with the problem of exponentially sized solutions for planning problems an even stronger concept of expressive equivalence is introduced, using the novel *ESP-reduction*. Four different formalisms for propositional planning are then analyzed, namely two variants of STRIPS, ground TWEAK and the SAS$^+$ formalism. Although these may seem to exhibit different degrees of expressive power, it is proven that they are, in fact, expressively equivalent under ESP reduction. This means that neither negative goals, partial initial states nor multi-valued state variables increase the expressiveness of 'standard' propositional STRIPS.

This article is published in *Artificial Intelligence*, vol. 76(1–2):17–34, 1995, ARTINT 1234.

1 Introduction

This article analyzes and compares four formalisms for propositional planning with respect to expressive power. The reason for this analysis is twofold. Firstly, if two formalisms can be proven equally expressive, under some reasonable notion of expressive equivalence, then various theoretical results will carry over between these formalisms. We may, for instance, desire that complexity results carry over. Furthermore, there will also be the more practical consequence that a planning algorithm designed for one of the formalisms can

[1] Supported by the Swedish Research Council for Engineering Sciences (TFR) under grant Dnr. 92-143.

be easily used also for planning in the other formalism. Secondly, by formally analysing the expressive power of the formalisms, we are able to prove or disprove some 'folklore' assumptions about the relative expressiveness of the formalisms.

There is hardly any consensus about what expressive equivalence between formalisms means. We have chosen in this article to say that two planning formalisms are expressively equivalent if the planning problem expressed in one of the formalisms can be polynomially transformed into the planning problem expressed in the other formalism. (Actually, we will use an even stronger form of polynomial reduction for the equivalence proofs in this article). That is, an instance of the planning problem in one formalism can be converted to an equivalent instance of the planning problem in the other formalism in polynomial time. Although other definitions may also be reasonable from some perspective, our definition is very appealing. It gives us the properties we wished above, that is, complexity results carry over immediately and a planning algorithm designed for one formalism can be easily and reasonably efficiently used also for another formalism.

The four formalisms we have chosen to analyze are all propositional variants of the STRIPS formalism [14]. The first two formalisms are both plain propositional variants of STRIPS, differing only in whether negative preconditions and goals are allowed or not. We refer to these formalisms as *Common Propositional STRIPS (CPS)* and *Propositional STRIPS with Negative goals (PSN)* respectively. The third formalism is the ground (*ie.* propositional) variant of TWEAK [11] and it is, thus, closely related to the first two formalisms. We refer to this formalism as *Ground TWEAK (GT)*. The fourth, and final, formalism in our analysis is the *extended simplified action structures (SAS$^+$)* formalism [2,3,6,17], which derives from the SAS formalism [4,5] and the original *action structures* formalism [23].

These four formalism seem to form a sequence of successively more and more expressive power, in the order presented above. The PSN formalism adds to the CPS formalism the capability of expressing negative goals and subgoals (preconditions), *ie.* we may not only state what must be true in the goal, but also what must *not* be true. The GT formalism further adds incomplete initial states, *ie.*, the truth value of a proposition may be undefined not only in the goal state but also in the initial state. The SAS$^+$ formalism, finally, generalizes the propositions to multi-valued state variables. All these features are summarized in Table 1.

Although it looks as if the four formalisms actually are of different and increasing expressive power, this turns out not to be the case. All four formalisms are, in fact, equally expressive for planning, as we will prove in Section 4 of this article. That is, neither of the features listed in Table 1 (negative goals

Table 1
A comparison of the CPS, PSN, GT and SAS$^+$ formalisms.

	CPS	PSN	GT	SAS$^+$
Partial goals	●	●	●	●
Negative preconditions		●	●	●
Negative goals		●	●	●
Partial initial states			●	●
Multi-valued state variables				●

and subgoals, incomplete initial states and multi-valued state variables) adds to the expressiveness of propositional planning.[2] This does not necessarily imply that we should always restrict ourselves to using one of these formalism only, perhaps elevating one of them to the status of being a 'standard'. There may be good reasons for using either formalism, depending on the purpose. One formalism may, for conceptual reasons, be better suited than another for modelling a certain application—perhaps making it easier and more natural to model this application. How the equivalence results *should* be interpreted is rather as follows. There is no *a priori* reason for choosing one formalism over the other; rather, knowing that they are all equally expressive, we know that it does not matter which one we use to model an application, we can always transform and compare our modelling to others done in the other formalisms. For instance, we may find it conceptually appealing and natural to use state variables, that is, the SAS$^+$ formalism, to model a certain application. At the same time, we may have a very good planning system for, say, the CPS formalism, perhaps providing us with a good user interface and other facilities. We then know that this is no conflict; we can go ahead and model our application in the SAS$^+$ formalism and it is then trivial to convert this modelling automatically to an equivalent modelling in the CPS formalism. That is, we can use both our favourite formalism for modelling and our favourite formalism for planning even when these do not coincide.

Since all four formalisms analyzed in this article are very basic and simple it may perhaps deserve some motivation why it is interesting to restrict an analysis to these four only. One reason is that much of the theoretical work in planning is still based on these formalisms, for good reasons; they are simple and clean enough for analyzing various theoretical issues, most of which could not be analyzed for more complex formalisms until sorted out for these simpler ones. Furthermore, even for these restricted formalisms, the relative expressive power is not obvious to most people. Hence, this analysis provides

[2] Actually, it is not important for the expressiveness whether the goals are allowed to be partial or negative, it only matters whether subgoals (*ie.* operator preconditions) are.

a good starting point for carrying on with analysing other features of planning formalisms.

The rest of this article is laid out as follows. Section 2 formally defines the four planning formalisms and Section 3 formally defines the concepts of planning problem and equal expressiveness. In Section 4 the four formalisms are proven expressively equivalent and Section 5 ends the article with a brief discussion and conclusions.

2 Four Planning Formalisms

This section introduces formally the four formalisms mentioned in the introduction. In order to simplify the analysis, we will only be concerned with total-order plans (linear plans) in this article. This does not restrict the analysis, however, since the set of partial-order plans solving an instance of a planning problem can be viewed as a compact representation of the set of total-order plans solving the same instance. Before introducing the four formalisms, we define some formalism-independent concepts.

Definition 1 *Given a set of operators $\mathcal{O}$, we define the set of all operator sequences over $\mathcal{O}$ as*

$$\mathit{Seqs}(\mathcal{O}) = \{\langle\rangle\} \cup \{\langle o\rangle;\omega \mid o \in \mathcal{O} \,\text{and}\, \omega \in \mathit{Seqs}(\mathcal{O})\},$$

where ; is the sequence concatenation operator.

Definition 2 *Given a set $\mathcal{P} = \{p_1, \ldots, p_n\}$ of propositional atoms, $\mathcal{L}_\mathcal{P}$ denotes the corresponding set of literals,* ie. *$\mathcal{L}_\mathcal{P} = \{p, \neg p \mid p \in \mathcal{P}\}$. A set $S \subseteq \mathcal{L}_\mathcal{P}$ of literals is* consistent *iff there is no atom p such that $\{p, \neg p\} \subseteq S$. The function $\mathit{Neg} : \mathbf{2}^{\mathcal{L}_\mathcal{P}} \to \mathbf{2}^{\mathcal{L}_\mathcal{P}}$ is defined for consistent $S \subseteq \mathcal{L}_\mathcal{P}$ as*

$$\mathit{Neg}(S) = \{p \mid \neg p \in S\} \cup \{\neg p \mid p \in S\}.$$

The function $\mathit{Compl}_\mathcal{P} : \mathbf{2}^\mathcal{P} \to \mathbf{2}^{\mathcal{L}_\mathcal{P}}$ is defined for all $S \subseteq \mathcal{P}$ as

$$\mathit{Compl}_\mathcal{P}(S) = S \cup \mathit{Neg}(\mathcal{P} - S).$$

In other words, $\mathit{Neg}(S)$ denotes the *negation* of S as this is defined in, for instance, Kleene's three-valued logics [18]. Note that Neg is well-defined also for $S \subseteq \mathcal{P}$ since $\mathcal{P} \subseteq \mathcal{L}_\mathcal{P}$. Further, $\mathit{Compl}_\mathcal{P}(S)$ denotes the *completion* of S, *ie.*, it converts a total state of atoms into a total state of literals.

2.1 Common Propositional STRIPS

In the Common Propositional STRIPS (CPS) formalism[3] a planning problem is modelled by a set of *propositional atoms*, a set of *operators*, an *initial state* and a *goal state*. Operators are modelled by a *precondition*, a *positive postcondition (the add list)* and a *negative postcondition (the delete list)*. The initial state and any state resulting from executing an operator is represented by a set of atoms—interpreted such that an atom is true in this state iff it is present there. The goal state is interpreted as a positive partial state, *ie.*, those atoms mentioned in the goal must be true. The atoms not mentioned are interpreted as 'don't care', *ie.*, we do not commit ourselves to any particular truth value for these; they may be either true or false after executing the plan. The goal is, thus, not the exact state we want to hold after executing the plan, but a minimum requirement for what that state must look like. The preconditions act as subgoals and are, thus, interpreted in the same way as the goal state. An operator can be executed in a state if its precondition is satisfied there—*ie.*, the precondition is a minimum requirement for what must hold in the state where the operator is executed. Finally, if an operator is executed in a state where its precondition is satisfied, then the resulting state is calculated from the current state by adding those atoms mentioned in the add list and removing those mentioned in the delete list. Hence, all atoms that are not explicitly deleted remain true after executing an operator—the so called *STRIPS assumption.*

The Propositional STRIPS with Negative Goals (PSN) formalism[4] is a generalization of the CPS formalism, allowing negative goals and negative preconditions. Hence, we cannot only specify what atoms must be true in the final state, but also which atoms must be false there. This is done by dividing the goal into one set of atoms that must be present in the final state and one set of atoms that must not be present in the final state. Preconditions are divided in the same way. Note that both the goal and the preconditions are still partial states since an atom need not be mentioned at all; such an atom may be either true or false.

Since the the CPS formalism can be viewed as a restriction of the PSN formalism, we define the PSN formalism first.

Definition 3 *An instance of the* PSN planning problem *is a quadruple* $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \langle \mathcal{G}^+, \mathcal{G}^- \rangle \rangle$ *where*

– $\mathcal{P}$ *is a finite set of* atoms*;*

[3] So called because it is probably the most frequent propositional variant of STRIPS, used by, for instance, Minton *et al.* [20] and McAllester and Rosenblitt [19].
[4] Used by, for instance, Bylander [9] and Nebel and Koehler [21].

– $\mathcal{O}$ *is a finite set of* operators *of the form* $\langle \varphi, \eta, \alpha, \delta \rangle$, *where* $\varphi, \eta \subseteq \mathcal{P}$ *denote the* positive *and* negative precondition *respectively, satisfying* $\varphi \cap \eta = \emptyset$, *and* $\alpha, \delta \subseteq \mathcal{P}$ *denote the* positive *and* negative postcondition *(add and delete list) respectively, satisfying* $\alpha \cap \delta = \emptyset$;
– $\mathcal{I} \subseteq \mathcal{P}$ *denotes the* initial state *and* $\mathcal{G}^+, \mathcal{G}^- \subseteq \mathcal{P}$ *denote the* positive *and* negative goal *respectively, satisfying* $\mathcal{G}^+ \cap \mathcal{G}^- = \emptyset$.

For $o = \langle \varphi, \eta, \alpha, \delta \rangle \subseteq \mathcal{O}$, we write $\varphi(o)$, $\eta(o)$, $\alpha(o)$ and $\delta(o)$ to denote φ, η, α and δ respectively. A sequence $\langle o_1, \ldots, o_n \rangle \in Seqs(\mathcal{O})$ of operators is called a *PSN plan* (or simply plan) over Π.

Definition 4 *The ternary relation* $Valid_{PSN} \subseteq Seqs(\mathcal{O}) \times 2^{\mathcal{P}} \times (2^{\mathcal{P}} \times 2^{\mathcal{P}})$ *is defined s.t. for arbitrary* $\langle o_1, \ldots, o_n \rangle \in Seqs(\mathcal{O})$ *and* $S, T^+, T^- \subseteq \mathcal{P}$, $Valid_{PSN}(\langle o_1, \ldots, o_n \rangle, S, \langle T^+, T^- \rangle)$ *iff either*

(i) $n = 0$, $T^+ \subseteq S$ *and* $T^- \cap S = \emptyset$ *or*
(ii) $n > 0$, $\varphi(o_1) \subseteq S$, $\eta(o_1) \cap S = \emptyset$ *and*
$Valid_{PSN}(\langle o_2, \ldots, o_n \rangle, (S - \delta(o_1)) \cup \alpha(o_1), \langle T^+, T^- \rangle)$.

A plan $\langle o_1, \ldots, o_n \rangle \in Seqs(\mathcal{O})$ *is a* solution *to* Π *iff* $Valid_{PSN}(\langle o_1, \ldots, o_n \rangle, \mathcal{I}, \mathcal{G})$.

The CPS planning problem can now be defined as the restriction of the PSN planning problem to instances having no negative goals and operator sets consisting only of operators with no negative preconditions.

Definition 5 *An instance of the* CPS planning problem *is a tuple* $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ *s.t.* $\langle \mathcal{P}, \mathcal{O}', \mathcal{I}, \langle \mathcal{G}, \emptyset \rangle \rangle$, *where* $\mathcal{O}' = \{\langle \varphi, \emptyset, \alpha, \delta \rangle \mid \langle \varphi, \alpha, \delta \rangle \in \mathcal{O}\}$, *is an instance of the PSN planning problem.*

2.2 Ground TWEAK

The Ground TWEAK (GT) formalism is the ground (or propositional) version of the TWEAK formalism [11], that is, the TWEAK formalism restricted to only ground literals. [5] In this formalism the initial state and the intermediate states resulting from executing operators are also partial states. [6] Hence, we can distinguish between atoms that are true, false and unknown respectively

[5] Chapman uses the word proposition instead of the more standard word literal.
[6] It seems not quite clear whether Chapman intended to allow incomplete initial states and whether the TWEAK *planner* makes use of this—his definitions are somewhat unclear at this point. However, the TWEAK formalism *per se* incorporates this feature, and this is also our interpretation of Chapman's definitions. Hence, we use the TWEAK formalism under this interpretation rather than introducing incomplete initial states into the PSN formalism.

also in these states. Since TWEAK uses literals, an operator need only be modelled by a precondition and a postcondition, both being sets of literals, that is, partial states. The initial state and the goal state are also partial states. We require states to be consistent, *ie.*, we do not allow both an atom and its negation to be present in a state.

Definition 6 *An instance of the* GT planning problem *is a quadruple* $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ *where*

- $\mathcal{P}$ *is a finite set of* atoms*;*
- $\mathcal{O}$ *is a finite set of* operators *of the form* $\langle pre, post \rangle$ *where* $pre, post \subseteq \mathcal{L}_{\mathcal{P}}$ *are consistent and denote the* pre *and* post condition *respectively;*
- $\mathcal{I}, \mathcal{G} \subseteq \mathcal{L}_{\mathcal{P}}$ *are consistent and denote the* initial *and* goal state *respectively.*

For $o = \langle pre, post \rangle \subseteq \mathcal{O}$, we write $pre(o)$ and $post(o)$ to denote pre and $post$ respectively. A sequence $\langle o_1, \ldots, o_n \rangle \in Seqs(\mathcal{O})$ of operators is called a *GT plan* (or simply a plan) over Π.

Definition 7 *The ternary relation* $Valid_{GT} \subseteq Seqs(\mathcal{O}) \times 2^{\mathcal{L}_{\mathcal{P}}} \times 2^{\mathcal{L}_{\mathcal{P}}}$ *is defined s.t. for arbitrary* $\langle o_1, \ldots, o_n \rangle \in Seqs(\mathcal{O})$ *and* $S, T \subseteq \mathcal{L}_{\mathcal{P}}$, $Valid_{GT}(\langle o_1, \ldots, o_n \rangle, S, T)$ *iff either*

(i) $n = 0$ *and* $T \subseteq S$ *or*
(ii) $n > 0$, $pre(o_1) \subseteq S$ *and*
$Valid_{GT}(\langle o_2, \ldots, o_n \rangle, (S - Neg(post(o_1)) \cup post(o_1), T)$.

A plan $\langle o_1, \ldots, o_n \rangle \in Seqs(\mathcal{O})$ *is a* solution *to* Π *iff* $Valid_{GT}(\langle o_1, \ldots, o_n \rangle, \mathcal{I}, \mathcal{G})$.

2.3 The SAS$^+$ Formalism

There are two main differences between the SAS$^+$ formalism and the other three formalisms. The first one is that the SAS$^+$ formalism uses partial, multi-valued state variables instead of propositional atoms or literals. In the GT formalism, an atom can, in principle, have three possible values in a state, namely true, false and unknown, depending on whether p, $\neg p$ or neither respectively is present in the state. In the SAS$^+$ formalism, this is generalized so that a state variable can have an arbitrary number of defined values in addition to the undefined value u. The second difference is that the operators have a *prevail-condition* in addition to the usual pre- and postconditions. This makes it possible to distinguish easily between those variables in the (STRIPS) precondition that are changed by the operator and those that remain unchanged.[7] That is, the (SAS$^+$) precondition of an operator specifies those state variables which must have a certain defined value in order to execute the operator and that will also be *changed* to some other value by the operator. The prevail-condition, on the other hand, specifies those state variables that must have a certain value but will remain *unchanged* after executing the operator. An operator can be executed in a state if both the pre- and prevail-condition are satisfied there, that is, all variables having some defined value in either of these conditions (a variable cannot be defined in both these conditions) has the same value in the state. When the operator is executed, then every variable that is defined in the postcondition of the operator will have that particular value in the resulting state; all other variables will have the same value as they had in the state the operator was executed in, which essentially is the STRIPS assumption.

In this article we will present a somewhat simplified and modified account of the SAS$^+$ formalism, ignoring all technical details not absolutely necessary for the proofs in Section 4.

Definition 8 *An instance of the SAS$^+$ planning problem is given by a quadruple $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_* \rangle$ with components defined as follows:*

- $\mathcal{V} = \{v_1, \ldots, v_m\}$ *is a set of* state variables. *Each variable $v \in \mathcal{V}$ has an associated* domain *of values $\mathcal{D}_v$, which implicitly defines an* extended domain $\mathcal{D}_v^+ = \mathcal{D}_v \cup \{\mathsf{u}\}$*, where* u *denotes the* undefined value. *Further, the* total state space $\mathcal{S} = \mathcal{D}_{v_1} \times \ldots \times \mathcal{D}_{v_m}$ *and the* partial state space $\mathcal{S}^+ = \mathcal{D}_{v_1}^+ \times \ldots \times \mathcal{D}_{v_m}^+$ *are implicitly defined. We write $s[v]$ to denote the value of the variable v in a state s.*
- $\mathcal{O}$ *is a set of* operators *of the form $\langle \mathsf{b}, \mathsf{e}, \mathsf{f} \rangle$, where $\mathsf{b}, \mathsf{e}, \mathsf{f} \in \mathcal{S}^+$ denote the* pre-, post- *and* prevail-condition *respectively. $\mathcal{O}$ is subject to the following two restrictions*
  (R1) *for all $\langle \mathsf{b}, \mathsf{e}, \mathsf{f} \rangle \in \mathcal{O}$ and $v \in \mathcal{V}$ if $\mathsf{b}[v] \neq \mathsf{u}$, then $\mathsf{b}[v] \neq \mathsf{e}[v] \neq \mathsf{u}$,*
  (R2) *for all $\langle \mathsf{b}, \mathsf{e}, \mathsf{f} \rangle \in \mathcal{O}$ and $v \in \mathcal{V}$, $\mathsf{e}[v] = \mathsf{u}$ or $\mathsf{f}[v] = \mathsf{u}$.*
- $s_0 \in \mathcal{S}^+$ *and $s_* \in \mathcal{S}^+$ denote the* initial *and* goal state *respectively.*

Restriction R1 essentially says that a state variable can never be made undefined, once made defined by some operator. Restriction R2 says that the pre- and prevail-conditions of an operator must never define the same variable. We further write $s \sqsubseteq t$ if the state s is subsumed (or satisfied) by state t, *ie.* if

[7] Although not technically necessary, this distinction between the precondition and the prevail-condition has shown to have conceptual advantages in some cases. For instance, it has been possible to identify certain restrictions that result in computationally tractable subcases of the SAS$^+$ planning problem [2–6,17]. Distinguishing the changed and unchanged parts of the preconditions has also made it easier to define criteria for possible parallel execution of operators [2,4,5].

$s[v] = \mathsf{u}$ or $s[v] = t[v]$. We extend this notion to states, defining

$$s \sqsubseteq t \text{ iff forall } v \in \mathcal{V}, s[v] = \mathsf{u} \text{ or } s[v] = t[v].$$

For $o = \langle \mathsf{b}, \mathsf{e}, \mathsf{f} \rangle$ is a SAS$^+$ operator, we write $\mathsf{b}(o)$, $\mathsf{e}(o)$ and $\mathsf{f}(o)$ to denote b, e and f respectively. A sequence $\langle o_1, \ldots, o_n \rangle \in \mathit{Seqs}(\mathcal{O})$ of operators is called a *SAS$^+$ plan* (or simply a plan) over Π.

Definition 9 *Given two states $s, t \in \mathcal{S}^+$, we define for all $v \in \mathcal{V}$,*

$$(s \oplus t)[v] = \begin{cases} t[v], & \text{if } t[v] \neq \mathsf{u}, \\ s[v], & \text{otherwise.} \end{cases}$$

The ternary relation $\mathit{Valid}_{\mathrm{SAS}^+} \subseteq \mathit{Seqs}(\mathcal{O}) \times \mathcal{S}^+ \times \mathcal{S}^+$ is defined recursively s.t. for arbitrary operator sequence $\langle o_1, \ldots, o_n \rangle \in \mathit{Seqs}(\mathcal{O})$ and arbitrary states $s, t \in \mathcal{S}^+$, $\mathit{Valid}_{\mathrm{SAS}^+}(\langle o_1, \ldots, o_n \rangle, s, t)$ iff either

(i) $n = 0$ *and* $t \sqsubseteq s$ *or*
(ii) $n > 0$, $\mathsf{b}(o_1) \sqsubseteq s$, $\mathsf{f}(o_1) \sqsubseteq s$ *and*
$\mathit{Valid}_{\mathrm{SAS}^+}(\langle o_2, \ldots, o_n \rangle, (s \oplus \mathsf{e}(o_1)), t)$.

A plan $\langle o_1, \ldots, o_n \rangle \in \mathit{Seqs}(\mathcal{O})$ is a solution *to* Π *iff $\mathit{Valid}_{\mathrm{SAS}^+}(\langle o_1, \ldots, o_n \rangle, s_0, s_*)$.*

3 Planning Problems and Equal Expressiveness

For each of the formalisms presented above, we have defined the concept of a planning problem implicitly by defining what its instances look like. More precisely, we define a planning problem as follows.

Definition 10 *Given a planning formalism X, the* (general) planning problem *in X (X-**GPP**) consists of a set of instances, each instance Π having an associated set $\mathit{Sol}(\Pi)$ of solutions. Furthermore, given a solution set $\mathit{Sol}(\Pi)$, for each $k \geq 0$, $\mathit{Sol}_k(\Pi)$ denotes the subset of $\mathit{Sol}(\Pi)$ restricted to solutions of length k only (ie. valid plans with k operators).*

We specialize this problem into the corresponding bounded and unbounded decision (*ie.* existence) and search (*ie.* generation) problems. The bounded problems are the optimization versions, *ie.* finding or deciding the existence of a minimal-length plan.

Definition 11 *Given a planning formalism X, the planning problem in X can be specialized as follows. The* plan existence problem *in X (X-**PE**) is: given an instance Π,* decide whether *$Sol(\Pi) \neq \emptyset$ or not, ie. whether Π has a solution or not. The* bounded plan existence problem *in X (X-**BPE**) takes an integer $k \geq 0$ as an additional parameter and asks if $Sol_n(\Pi) \neq \emptyset$ for some $n \leq k$, ie. whether Π has a solution of length k or shorter. The* plan generation problem *in X (X-**PG**) is: given an instance Π,* find *a member of $Sol(\Pi)$ or answer that $Sol(\Pi)$ is empty. The* bounded plan generation problem *in X (X-**BPG**) takes an integer $k \geq 0$ as an additional parameter and returns a member of $Sol_n(\Pi)$ for some $n \leq k$ or answers that $Sol_n(\Pi)$ is empty for all $n \leq k$.*

As mentioned in the introduction, we have chosen to define the concept equal expressiveness using polynomial transformations. [8] As long as we consider only the (bounded) plan existence problems, it is straightforward to define equal expressiveness under polynomial transformation.

Definition 12 *Given two planning formalisms X and Y, we say that X is* at least as expressive as *Y wrt. plan existence if Y-**GPP** $\leq_p$ X-**GPP**, ie. Y-**GPP** polynomially transforms into X-**GPP**. Further, X and Y are* equally expressive *with respect to plan existence iff both X-**GPP** $\leq_p$ Y-**GPP** and Y-**GPP** $\leq_p$ X-**GPP**. The corresponding definitions for bounded plan existence are analogous.*

The motivation for this definition of equal expressiveness is as follows. If we know that formalisms X and Y are equally expressive with respect to plan existence, then a proof that X-**PE** belongs to a certain complexity class immediately allows us to conclude that also Y-**PE** belongs to that complexity class, and *vice versa*. Similarly, hardness and completeness results for complexity classes also carry over immediately.

We are mostly interested in finding a plan, however, not only finding out whether one exists. Hence, we must also consider expressive equivalence with respect to plan generation. Usually there is a strong relationship between a decision problem and its corresponding search problem. For instance, if a decision problem is NP-complete, then the search problem is usually NP-equivalent and if two search problems can be polynomially transformed into each other, then the corresponding search problems can usually be Turing reduced to each other. Furthermore, the bounded plan generation problem can be solved by employing an oracle (or an algorithm) for the bounded plan existence problem and using *prefix search* [8,15]. For most problems, such a prefix search

[8] The reader not being familiar with the concepts of problems and polynomial transformations may wish to consult the literature [8,15,16].

strategy is a Turing reduction. This is not the case for the planning, however, since instances of the planning problems may have even minimal solutions of exponential length in the size of the instance [2,6]. It has been argued that such instances should be regarded as unrealistic in most cases [2,15]. Yet, such instances are allowed by the formalisms considered in this article and it seems non-trivial (or even impossible) to restrict the planning problem to instances with polynomially sized minimal solutions only. Hence, we must take these exponential solutions into account, even if they are not of any practical interest. For the formalisms considered in this article we will prove expressive equivalence in a very strong sense, using the novel *exact structure-preserving reduction (ESP-reduction)*. This reduction is a modification of the structure-preserving reduction invented by Ausiello, D'Atri and Protasi [1] and is defined as follows.

Definition 13 *Given two planning formalisms X and Y, an* ESP-reduction *of X-**GPP** into Y-**GPP**, with instance sets $Inst_X$ and $Inst_Y$ respectively, is a polynomial-time computable function $f : Inst_X \to Inst_Y$ s.t. for all $\Pi \in Inst_X$, (1) $f(\Pi) \in Inst_Y$ and (2) for all $k \geq 0$, $|Sol_k(\Pi)| = |Sol_k(f(\Pi))|$. We denote ESP-reducibility by $\leq_{ESP}$.*

ESP-reductions enjoy the usual properties for reductions and imply polynomial transformability between existence problems.

Theorem 14 *(1) ESP-reductions are reflexive and transitive. (2) Given two planning formalisms X and Y, if X-**GPP** $\leq_{ESP}$ Y-**GPP**, then X-**PE** $\leq_p$ Y-**PE** and X-**BPE** $\leq_p$ Y-**BPE**.*

Using ESP-reductions, an even stronger concept of equal expressiveness of planning formalisms can be defined, preserving the size distribution of the solution set, thus also preserving exponentially sized solutions.

Definition 15 *Given two planning formalisms X and Y, we say that X is* at least as expressive as *Y under ESP-reduction if Y-**GPP** $\leq_{ESP}$ X-**GPP**. Further, X and Y are* equally expressive *under ESP reduction iff both X-**GPP** $\leq_{ESP}$ Y-**GPP** and Y-**GPP** $\leq_{ESP}$ X-**GPP**.*

We will use this concept of expressive equivalence for the formalism considered in this paper. However, in general this may be an overly strong condition for expressive equivalence, since it requires that the structure of the solution set is preserved, and a more general and tolerant concept which still preserves exponentially sized solutions can be found in Bäckström [2, Section 5]. However, from another perspective, it may, to the contrary, seem that basing the concept of equal expressiveness on polynomial reductions of various kinds is too weak, for the following reason. Suppose A is an algorithm solving size n instances of Y-**PG** in $O(f(n))$ time, for some function f, and ξ is an ESP reduction converting size n instances of X-**PG** to size $O(p(n))$ instances of

Y-**PG**, for some polynomial p. Then we can solve a size n instance Π of X-**PG** with at most a polynomial blow-up in instance size, *ie.* in time $O(f(p(n)))$, by solving the instance $\xi(\Pi)$ using algorithm A. If we could solve Π directly also in $O(f(n))$ time by some algorithm A' for X-**PG**, which need not be possible, and f is an exponential function, then we risk an exponential slow-down by applying A to $\xi(\Pi)$ instead of solving Π directly using A'. It thus seems desirable to base a concept of equal expressiveness on an even more fine grained criterion. However, it seems hard to define such a criterion and our definition at least preserves membership in the usual complexity classes.

Finally, the definition of equal expressiveness should ideally also be accompanied by some requirement for constructive evidence that a sufficiently simple and natural ESP-reduction exists. Otherwise, we would get very contrived cases like the language of quantified Boolean formulae, considered as a planning formalism, being expressively equivalent to the CPS language since the existence problems are PSPACE-complete in both cases. Such equivalences were not intended by our concept, but are non trivial to define away since we can hardly define what "simple and natural" means. However, all equivalence proofs in this article are constructive, providing simple and straightforward reductions.

4 Equivalence Proofs

In this section we will prove[9] that the four formalisms defined in Section 2 are all equally expressive under ESP-reduction, that is, we will prove that the planning problem expressed in any of the four formalisms ESP-reduces to the planning problem expressed in either of the other three formalisms. More specifically, we will explicitly construct the ESP-reductions shown in Figure 1 where each ξ^X_Y is an ESP-reduction from X-**GPP** to Y-**GPP**. The rest follows from transitivity.

Bylander [9,10] has shown that plan existence is PSPACE-complete for both CPS plan existence and PSN plan existence. Hence, we know that there *exist* polynomial transformations between these problems, but we have no idea what these may look like. That is, we do not know how to solve one of these problems given an algorithm for the other one, with only polynomial overhead; we only know that this is possible. In contrast, the proofs given below are constructive in the sense that they provide explicit transformations. That is, they tell us how to convert a problem instance in one formalism to an equivalent instance in another formalism. Furthermore, they also handle the

[9] By request from the editors, most proofs are omitted in this article. Full proofs are provided in Bäckström [2] (using a slightly different type of reduction, though).

$$\begin{tikzcd}
& \boxed{\text{PSN}} \arrow[dr, "\xi^{PSN}_{GT}"] & \\
\boxed{\text{CPS}} \arrow[ur, "\xi^{CPS}_{PSN}"] & & \boxed{\text{GT}} \arrow[dl, "\xi^{GT}_{SAS^+}"] \\
& \boxed{\text{SAS}^+} \arrow[ul, "\xi^{SAS^+}_{CPS} = \xi^{SAS^+}_{GT} \circ \xi^{GT}_{CPS}"'] &
\end{tikzcd}$$

Fig. 1. ESP-reductions to be proven between the planning problemexpressed in the four formalisms.

(bounded) plan generation problem implicitly by using ESP-reduction instead of polynomial transformations.

Reducing CPS instances into equivalent PSN instances and PSN instances into equivalent GT instances respectively is straightforward.

Definition 16 (CPS to PSN) *Given a CPS instance* $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$*, we define* $\xi^{CPS}_{PSN}(\Pi) = \langle \mathcal{P}, \mathcal{O}', \mathcal{I}, \langle \mathcal{G}, \emptyset \rangle \rangle$*, where* $\mathcal{O}' = \{ \langle \varphi, \emptyset, \alpha, \delta \rangle \mid \langle \varphi, \alpha, \delta \rangle \in \mathcal{O} \}$.

Theorem 17 ξ^{CPS}_{PSN} *is an ESP-reduction from CPS-***GPP** *to PSN-***GPP***.*

Definition 18 (PSN to GT) *Given a PSN instance* $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \langle \mathcal{G}^+, \mathcal{G}^- \rangle \rangle$*, we define*

$$\xi^{PSN}_{GT}(\Pi) = \langle \mathcal{P}, \mathcal{O}', Compl_{\mathcal{P}}(\mathcal{I}), \mathcal{G}^+ \cup Neg(\mathcal{G}^-) \rangle,$$

where

$$\mathcal{O}' = \{ \langle \varphi \cup Neg(\eta), \alpha \cup Neg(\delta) \rangle \mid \langle \varphi, \eta, \alpha, \delta \rangle \in \mathcal{O} \}.$$

Theorem 19 ξ^{PSN}_{GT} *is an ESP-reduction from PSN-***GPP** *to GT-***GPP***.*

GT planning can be straightforwardly ESP-reduced into SAS$^+$ planning by mapping each propositional atom onto a binary state variable. Also, the precondition of an operator has to be split into the pre- and prevail-conditions for the corresponding operator type depending on whether this atom appears in the postcondition or not, but this is straightforward.

Definition 20 (GT to SAS$^+$) *Let* $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ *be an arbitrary instance of the GT planning problem. Wlg. assume* $\mathcal{P} = \{p_1, \ldots, p_m\}$*. Further define* $L_i = \{p_i, \neg p_i\}$ *for* $1 \leq i \leq m$*. Now define the set of variables* $\mathcal{V} = \{v_1, \ldots, v_m\}$*, s.t. for* $1 \leq i \leq m$*,* $\mathcal{D}_{v_i} = \{x_i, x'_i\}$ *and define the partial function* $\lambda : \mathbf{2}^{\mathcal{L}_{\mathcal{P}}} \rightarrow \mathcal{S}^+$

for all consistent $S \subseteq \mathcal{L}_{\mathcal{P}}$ s.t. for $1 \leq i \leq m$,

$$\lambda(S)[v_i] = \begin{cases} x_i, & \text{if } S \cap L_i = \{p_i\}, \\ x'_i, & \text{if } S \cap L_i = \{\neg p_i\}, \\ \mathsf{u}, & \text{otherwise } (\textit{ie. } S \cap L_i = \emptyset). \end{cases}$$

We define $\xi^{GT}_{SAS^+}(\Pi) = \langle \mathcal{V}, \mathcal{O}', \lambda(\mathcal{I}), \lambda(\mathcal{G}) \rangle$, s.t. $\mathcal{O}' = \{o' | o \in \mathcal{O}\}$, where for each $o = \langle pre, post \rangle \in \mathcal{O}$, the corresponding operator $o' = \langle \mathsf{b}, \mathsf{e}, \mathsf{f} \rangle \in \mathcal{O}'$ is defined s.t. for $1 \leq i \leq m$,

$$\mathsf{b}(o)[v_i] = \begin{cases} \lambda(pre)[v_i], & \text{if } post \cap L_i \neq \emptyset, \\ \mathsf{u}, & \text{otherwise}, \end{cases}$$
$$\mathsf{e}(o)[v_i], = \lambda(post)[v_i],$$
$$\mathsf{f}(o)[v_i] = \begin{cases} \lambda(pre)[v_i], & \text{if } post \cap L_i = \emptyset, \\ \mathsf{u}, & \text{otherwise}. \end{cases}$$

Theorem 21 *$\xi^{GT}_{SAS^+}$ is an ESP-reduction from GT-***GPP** *to SAS$^+$-***GPP**.

The reduction $\xi^{SAS^+}_{CPS}$ will be constructed as the composition of two reductions $\xi^{SAS^+}_{GT}$ and ξ^{GT}_{CPS} as follows. Starting with $\xi^{SAS^+}_{GT}$, it may seem as if *SAS$^+$*-**GPP** could easily be reduced to *GT*-**GPP** by mapping every state variable onto n atoms, where n is the number of values in the domain of the state variable. However, this would not be guaranteed to yield an ESP-reduction in the general case, for the following reason. Suppose $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_* \rangle$ is an instance of the SAS$^+$ planning problem. We let $m = |\mathcal{V}|$, $n = \max_{v \in \mathcal{V}} |\mathcal{D}_v|$ and $l = |\mathcal{O}|$. We make the usual assumption of 'conciseness' [15] for the encoding of Π. For each $v \in \mathcal{V}$ we must represent $\mathcal{D}_v$, so the representation of $\mathcal{V}$ is of size $O(m \log n)$. Obviously, also each state takes $O(m \log n)$ space to represent, so $\mathcal{O}$ requires $O(l\, m \log n)$ space, thus dominating the size of Π. Now let $\Pi' = \langle \mathcal{P}, \mathcal{O}', \mathcal{I}, \mathcal{G} \rangle$ be a corresponding GT instance. Encoding each state variable domain as a set of mutually exclusive atoms requires $O(mn)$ atoms, so both $\mathcal{P}$ and each state requires $O(mn)$ space each. Furthermore, $\mathcal{O}'$ requires $O(l\, mn)$ space, thus dominating the size of Π'. We see that Π' may require an exponentially larger representation than Π if n is the dominating factor, so we have no guarantee that Π can be converted to Π' in polynomial time if we use the mapping above. Hence, we must map each state variable domain onto a logarithmic number of atoms, as follows.

Definition 22 *Let $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_* \rangle$ be an arbitrary instance of the SAS$^+$ planning problem. Wlg. assume $\mathcal{V} = \{v_1, \ldots, v_m\}$. For all $v \in \mathcal{V}$, define $k_v =$*

$\lceil \log_2 |\mathcal{D}_v| \rceil$ *and* $P_v = \{p_{v,1}, \ldots, p_{v,k_v}\}$. *Also define the set* $\mathcal{P} = P_{v_1} \cup \ldots \cup P_{v_m}$ *of all such atoms. Further define the partial function* $\mu_v : \mathcal{D}_v^+ \to \mathbf{2}^{\mathcal{L}_{P_v}}$ *as an arbitrary injection satisfying that* $\mu_v(\mathsf{u}) = \emptyset$ *and for all* $x \in \mathcal{D}_v$, $\mu_v(x)$ *is consistent and* $|\mu_v(x)| = k_v$.[10] *Then define the composite partial function* $\mu : \mathcal{S}^+ \to \mathbf{2}^{\mathcal{L}_\mathcal{P}}$ *s.t. for all consistent* $s \in \mathcal{S}^+$,

$$\mu(s) = \mu_{v_1}(s[v_1]) \cup \ldots \cup \mu_{v_m}(s[v_m]).$$

Finally, define

$$\xi_{GT}^{SAS^+}(\Pi) = \langle \mathcal{P}, \mathcal{O}', \mu(s_0), \mu(s_*) \rangle,$$

where

$$\mathcal{O}' = \{ \langle \mu(\mathsf{b}(o)) \cup \mu(\mathsf{f}(o)), \mu(\mathsf{e}(o)) \rangle \mid o \in \mathcal{O} \}.$$

To prove that $\xi_{GT}^{SAS^+}$ is an ESP-reduction, we first need to prove some properties of the function μ.

Lemma 23 *The following properties hold for the function* μ*:*

(i) The inverse μ^{-1} *of* μ *is defined for all consistent states* $S \subseteq \mathcal{L}_\mathcal{P}$, ie. μ *is a bijection from* $\mathcal{S}^+$ *to the set of consistent states in* $\mathcal{L}_\mathcal{P}$.
(ii) For all $s, t \in \mathcal{S}^+$, $s \sqsubseteq t$ *iff* $\mu(s) \subseteq \mu(t)$.
(iii) For arbitrary $s, t \in \mathcal{S}^+$, $\mu(s \oplus t) = \mu(s) - Neg(\mu(t)) \cup \mu(t)$.

Proof. 1 and 2 are trivial. To prove 3 it suffices to prove that for all $v \in \mathcal{V}$,

$$\begin{aligned} & \mu(s \oplus t) \cap \mathcal{L}_{P_v} \\ & = \mu_v((s \oplus t)[v]) \\ & = \begin{cases} \mu_v(t[v]), & \text{if} \, t[v] \neq \mathsf{u}, \\ \mu_v(s[v]), & \text{otherwise}, \end{cases} \\ & = \begin{cases} \mu_v(t[v]), & \text{if} \, \mu_v(t[v]) \neq \emptyset, \\ \mu_v(s[v]), & \text{otherwise}, \end{cases} \end{aligned}$$

[10] For instance, if assuming $\mathcal{D}_v^+ = \{0, \ldots, |\mathcal{D}_v^+| - 1\}$, then for all defined $x \in \mathcal{D}_v^+$ we can let $\mu_v(x)$ be the binary encoding of x s.t. for $1 \leq i \leq k_v$, the literals $p_{v,i}$ and $\neg p_{v,i}$ encode the values 1 and 0 respectively for bit $i - 1$ in this encoding.

$$= \begin{cases} \mu(t) \cap \mathcal{L}_{P_v}, & \text{if } \mu(t) \cap \mathcal{L}_{P_v} \neq \emptyset, \\ \mu(s) \cap \mathcal{L}_{P_v}, & \text{otherwise}, \end{cases}$$

$$= \begin{cases} (\mu(s) \cap \mathcal{L}_{P_v}) - (\mathit{Neg}(\mu(t) \cap \mathcal{L}_{P_v})) \cup (\mu(t) \cap \mathcal{L}_{P_v}), & \text{if } \mu(t) \cap \mathcal{L}_{P_v} \neq \emptyset, \\ \mu(s) \cap \mathcal{L}_{P_v}, & \text{otherwise}, \end{cases}$$

$$= \begin{cases} (\mu(s) - \mathit{Neg}(\mu(t)) \cup \mu(t)) \cap \mathcal{L}_{P_v}, & \text{if } \mu(t) \cap \mathcal{L}_{P_v} \neq \emptyset, \\ (\mu(s) - \mathit{Neg}(\mu(t)) \cup \mu(t)) \cap \mathcal{L}_{P_v}, & \text{otherwise}, \end{cases}$$

$$= (\mu(s) - \mathit{Neg}(\mu(t)) \cup \mu(t)) \cap \mathcal{L}_{P_v}$$

and, hence, $\mu(s \oplus t) = \mu(s) - \mathit{Neg}(\mu(t)) \cup \mu(t)$. □

Theorem 24 *$\xi_{GT}^{SAS^+}$ is an ESP-reduction from SAS^+-***GPP** *to GT-***GPP***.*

Proof. Let $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_* \rangle$ be an arbitrary SAS$^+$ instance and let $\Pi' = \xi_{GT}^{SAS^+}(\Pi) = \langle \mathcal{P}, \mathcal{O}', \mathcal{I}, \mathcal{G} \rangle$. Obviously, $\xi_{GT}^{SAS^+}$ can be computed in polynomial time, so it remains to prove that for each $k \geq 0$, $|\mathit{Sol}_k(\Pi)| = |\mathit{Sol}_k(\Pi')|$, *ie.* that there exists a bijection between $\mathit{Sol}(\Pi)$ and $\mathit{Sol}(\Pi')$. We prove this by showing that for all states $s, t \in \mathcal{S}^+$ and for every plan $\langle o_1, \ldots, o_n \rangle \in \mathit{Seqs}(\mathcal{O})$ and its corresponding plan $\langle o'_1, \ldots, o'_n \rangle \in \mathit{Seqs}(\mathcal{O}')$,

$$\mathit{Valid}_{\mathrm{SAS+}}(\langle o_1, \ldots, o_n \rangle, s, t) \text{ iff } \mathit{Valid}_{GT}(\langle o'_1, \ldots, o'_n \rangle, \mu(s), \mu(t)).$$

Proof by induction over n.

Basis: For the case where $n = 0$ it is sufficient to prove that $t \sqsubseteq s$ iff $\mu(t) \subseteq \mu(s)$, which is immediate from Lemma 23, parts 1 and 2.

Induction: Suppose the claim holds for all $n \leq k$ for some $k > 0$. We prove that the claim holds also for $n = k + 1$, tacitly using Lemma 23(1).

$\mathit{Valid}_{\mathrm{SAS+}}(\langle o_1, \ldots, o_n \rangle, s, t)$
iff $\mathsf{b}(o_1) \sqsubseteq s, \mathsf{f}(o_1) \sqsubseteq s$ and $\mathit{Valid}_{\mathrm{SAS+}}(\langle o_2, \ldots, o_n \rangle, (s \oplus \mathsf{e}(o_1)), t)$
iff (Using Lemma 23(2) and def. $\xi_{GT}^{SAS^+}$)
$\mu(\mathsf{b}(o_1)) \subseteq \mu(s), \mu(\mathsf{f}(o_1)) \subseteq \mu(s)$ and
$\mathit{Valid}_{GT}(\langle o'_2, \ldots, o'_n \rangle, \mu(s \oplus \mathsf{e}(o_1)), \mu(t))$
iff (Using Lemma 23(3))
$\mu(\mathsf{b}(o_1)) \cup \mu(\mathsf{f}(o_1)) \subseteq \mu(s)$ and
$\mathit{Valid}_{GT}(\langle o'_2, \ldots, o'_n \rangle, (\mu(s) - \mathit{Neg}(\mu(\mathsf{e}(o_1)))) \cup \mu(\mathsf{e}(o_1)), \mu(t))$
iff (Using def. $\xi_{GT}^{SAS^+}$)

$$\begin{aligned}&pre(o'_1) \subseteq \mu(s) \text{ and } Valid_{\mathrm{SAS^+}}(\langle o_2, \ldots, o_n\rangle, s \oplus \mathsf{e}(o_1), t)\\ \text{iff } & Valid_{GT}(\langle o'_1, \ldots, o'_n\rangle, \mu(s), \mu(t)).\end{aligned}$$

It is now immediate that $\xi_{GT}^{SAS^+}$ is an ESP-reduction. □

A perhaps more surprising result is that GT planning ESP-reduces to CPS planning. The trick used is to represent *each literal* in a GT problem instance with a *unique atom* in the corresponding CPS problem instance.

Definition 25 (GT to CPS) *Given an instance* $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G}\rangle$ *of the GT planning problem, we define* $\xi_{CPS}^{GT}(\Pi) = \langle \mathcal{L}_\mathcal{P}, \mathcal{O}', \mathcal{I}, \mathcal{G}\rangle$*, where* $\mathcal{O}' = \{\langle pre, post, Neg(post)\rangle \mid \langle pre, post\rangle \in \mathcal{O}\}$.

Note that all literals are treated as distinct, unrelated atoms in $\xi_{CPS}^{GT}(\Pi)$.[11]

Theorem 26 ξ_{CPS}^{GT} *is an ESP-reduction from GT-***GPP** *to CPS-***GPP**.

Proof. Let $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G}\rangle$ be an arbitrary GT instance and let $\Pi' = \xi_{CPS}^{GT}(\Pi) = \langle \mathcal{L}_\mathcal{P}, \mathcal{O}', \mathcal{I}, \mathcal{G}\rangle$. Obviously, ξ_{CPS}^{GT} can be computed in polynomial time, so it remains to prove that for each $k \geq 0$, $|Sol_k(\Pi)| = |Sol_k(\Pi')|$, *ie.* that there exists a bijection between $Sol(\Pi)$ and $Sol(\Pi')$. We prove this by showing that for all states $S, T \subseteq \mathcal{L}_\mathcal{P}$, and for every plan $\langle o_1, \ldots, o_n\rangle \in Seqs(\mathcal{O})$ and its corresponding plan $\langle o'_1, \ldots, o'_n\rangle \in Seqs(\mathcal{O}')$,

$$Valid_{CPS}(\langle o_1, \ldots, o_n\rangle, S, T) \text{ iff } Valid_{GT}(\langle o'_1, \ldots, o'_n\rangle, S, T).$$

Proof by induction over n.

Basis: The case where $n = 0$ is trivial.

Induction: Suppose the claim holds for all $n \leq k$ for some $k > 0$. We prove that the claim holds also for $n = k + 1$.

$$\begin{aligned}& Valid_{CPS}(\langle o_1, \ldots, o_n\rangle, S, T)\\ \text{iff } & \varphi(o_1) \subseteq S \text{ and } Valid_{CPS}(\langle o_2, \ldots, o_n\rangle, (S - \delta(o_1)) \cup \alpha(o_1), T)\\ \text{iff } & pre(o'_1) \subseteq S \text{ and } Valid_{GT}(\langle o'_2, \ldots, o'_n\rangle, (S - Neg(post(o'_1))) \cup post(o'_1), T)\\ \text{iff } & Valid_{GT}(\langle o'_1, \ldots, o'_n\rangle, S, T). \quad \square\end{aligned}$$

[11] To be precise, we should introduce a new atom in the CPS instance for each negative literal in $\mathcal{L}_\mathcal{P}$, but in order to keep the proof short and simple we make this implicitly by treating each negative literal as a unique atom in the CPS instance, trusting the benevolent reader to see how to make this distinction explicit.

Corollary 27 (SAS$^+$ to CPS) *The composite function* $\xi_{CPS}^{SAS^+} = \xi_{GT}^{SAS^+} \circ \xi_{CPS}^{GT}$ *is an ESP-reduction from* SAS^+-**GPP** *to* CPS-**GPP**.

Interreducibility between all pairs of formalisms then follows from transitivity of ESP-reductions.

Corollary 28 *The formalisms CPS, PSN, GT and SAS$^+$ are equally expressive under ESP-reduction.*

The equalities are not invariant with respect to further restrictions, though. For example, negative preconditions do add to the expressiveness if the delete lists are required to be empty. [12]

5 Discussion

Although the analysis in this article is restricted to four very basic formalisms, the results are nevertheless interesting. Many planning researchers seem to have assumed that at least some of these formalism exhibit different expressive power. The discovery that this is, in fact, not the case lead us to pose the question of which features actually do add to the expressiveness of a planning formalism. Complexity analyses have been presented [6,9,12] for restricted versions of the formalisms in this article, thus telling us something about the relative expressive power of restricted cases within some of the formalisms. However, even the unrestricted cases considered in this article are often considered too limited for most practical applications. One may, hence, ask which of the features that have been added to these basic formalisms in the literature actually do increase the expressive power.

It is beyond the scope of this article to make a thorough investigation of all such additions, but some observations are fairly easy to make. For instance, one simple extension to the GT formalism would be to allow sets of disjunctions of literals in the precondition, *ie.* allowing preconditions in conjunctive normal form. The only obvious way to encode such an extended GT operator in the 'standard' GT formalism seems to be to split it into several operators for each disjunction, resulting in an exponential blow-up of the problem instance. There seems to be no way of avoiding this exponentiality and it is thus highly unlikely that there should exist a polynomial transformation from the GT plan existence problem to this extended GT plan existence problem, al-

[12] This follows from the fact that PSN plan existence with empty delete lists is NP-complete [9, Theorem 4], but becomes polynomial if negative preconditions are not allowed [9, Footnote 2]. Hence, there can exist no polynomial transformation of the first problem to the second, unless P=NP.

lowing disjunctive preconditions. That is, disjunctive preconditions most likely increase the expressive power.

Another common and simple extension is to add variables to the GT formalism, resulting in the standard TWEAK formalism [11]. Allowing infinite variable domains makes plan existence undecidable [11,13] and, thus, trivially adds to the expressive power. For most real-world applications it will likely suffice to use finite variable domains, though, which, in principle, would correspond to the propositional case. Things are not quite that simple, however, since it seems that a TWEAK operator with the precondition $P(x)$, say, must be split into a number of operators, one for each object in the domain. That is, the operator with $P(x)$ in its preconditions must be replaced with several operators having preconditions $P(c_1), \ldots, P(c_n)$ respectively, where $c_1, \ldots, c_n$ are constants denoting the objects in the domain. If an operator has several variables in its precondition, we get an exponential blow-up also in this case. Hence, it seems that variables also add to the expressive power of the planning formalisms, even if restricted to finite domains.

To conclude, we have argued that it is appealing and useful to base the concept of expressive equivalence of formalisms on the concept of polynomial transformations. Using this definition, we have, further, proven that four common propositional planning formalisms that seem to exhibit various degrees of expressive power are in fact equally expressive. We believe that this may serve as a starting point for asking and analysing which of all the features incorporated into planning formalisms in the literature actually do add to the expressive power. It may, of course, be motivated to add a certain feature for conceptual reasons, making it easier and more natural to model certain applications. However, if claiming that this addition is necessary, one should also prove that it indeed adds to the expressive power.

Acknowledgement

The author would like to thank Bernhard Nebel, Bart Selman, Andrzej Lingas, Christos Levcopoulos and the anonymous reviewers for helpful comments and discussions on this article and earlier versions of it.

References

[1] G. Ausiello, A. D'Atri, and M. Protasi. Structure preserving reductions among convex optimizations problems. *J. Comput. Syst. Sci.*, 21:136–153, 1980.

[2] C. Bäckström. *Computational Complexity of Reasoning about Plans.* Doctoral dissertation, Linköping University, Linköping, Sweden, June 1992.

[3] C. Bäckström. Equivalence and tractability results for SAS$^+$ planning. In B. Swartout and B. Nebel, editors, *Proceedings of the 3rd International Conference on Principles on Knowledge Representation and Reasoning (KR-92)*, pages 126–137, Cambridge, MA, USA, Oct. 1992. Morgan Kaufmann.

[4] C. Bäckström and I. Klein. Parallel non-binary planning in polynomial time. In Reiter and Mylopoulos [22], pages 268–273.

[5] C. Bäckström and I. Klein. Planning in polynomial time: The SAS-PUBS class. *Computational Intelligence*, 7(3):181–197, Aug. 1991.

[6] C. Bäckström and B. Nebel. Complexity results for SAS$^+$ planning. In Bajcsy [7].

[7] R. Bajcsy, editor. *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, Chambéry, France, Aug.–Sept. 1993. Morgan Kaufmann.

[8] J. L. Balcázar, J. Diaz, and J. Gabarró. *Structural Complexity I.* Springer, 1988.

[9] T. Bylander. Complexity results for planning. In Reiter and Mylopoulos [22], pages 274–279.

[10] T. Bylander. The computational complexity of propositional STRIPS planning. Research technical report, Laboratory for Artificial Intelligence Research, The Ohio State University, Columbus, OH, USA, May 1992. To appear in *Artificial Intelligence.*

[11] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[12] K. Erol, D. S. Nau, and V. S. Subrahmanian. On the complexity of domain-independent planning. In *Proceedings of the 10th (US) National Conference on Artificial Intelligence (AAAI-92)*, pages 381–386, San José, CA, USA, July 1992. American Association for Artificial Intelligence.

[13] K. Erol, D. S. Nau, and V. S. Subrahmanian. When is planning decidable? In J. Hendler, editor, *Artificial Intelligence Planning Systems: Proceedings of the 1st International Conference*, pages 222–227, College Park, MD, USA, June 1992. Morgan Kaufmann.

[14] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[15] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York, 1979.

[16] D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Algorithms and Complexity*, volume A, chapter 2, pages 67–161. Elsevier, Amsterdam, 1990.

[17] P. Jonsson and C. Bäckström. Tractable planning with state variables by exploiting structural restrictions. In *Proceedings of the 12th (US) National Conference on Artificial Intelligence (AAAI-94)*, Seattle, WA, USA, July–Aug. 1994. American Association for Artificial Intelligence.

[18] S. C. Kleene. *Introduction to Metamathematics.* van Nostrand, 1952.

[19] D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proceedings of the 9th (US) National Conference on Artificial Intelligence (AAAI-91)*, pages 634–639, Anaheim, CA, USA, July 1991. American Association for Artificial Intelligence, AAAI Press/MIT Press.

[20] S. Minton, J. Bresina, and M. Drummond. Commitment strategies in planning: A comparative analysis. In Reiter and Mylopoulos [22], pages 259–265.

[21] B. Nebel and J. Koehler. Plan modification versus plan generation: A complexity-theoretic perspective. In Bajcsy [7], pages 1436–1441.

[22] R. Reiter and J. Mylopoulos, editors. *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, Sydney, Australia, Aug. 1991. Morgan Kaufmann.

[23] E. Sandewall and R. Rönnquist. A representation of action structures. In *Proceedings of the 5th (US) National Conference on Artificial Intelligence (AAAI-86)*, pages 89–97, Philadelphia, PA, USA, Aug. 1986. American Association for Artificial Intelligence, Morgan Kaufmann.