On the Complexity of Planning Operator Subsumption

Patrick Eyerich and **Michael Brenner** and **Bernhard Nebel**
Department of Computer Science
Albert-Ludwigs-Universität Freiburg
D-79110 Freiburg, Germany *

Abstract

Formal action models play a central role in several subfields of AI because they are used to model application domains, e.g., in automated planning. However, there are hitherto no automated methods for relating such domain models to each other, in particular for checking whether one is a specialization or generalization of the other. In this paper, we introduce two kinds of subsumption relations between operators, both of which are suitable for modeling and verifying hierarchies between actions and operators: *applicability* subsumption considers an action to be more general than another if the latter can be replaced by the first at each point in each sound sequence of actions; *abstraction* subsumption exploits relations between actions from an ontological point of view. For both kinds of subsumption, we prove complexity results for verifying operator subsumption in three important subclasses: The problems are $\mathcal{NP}$-complete when the expressiveness of the operators is restricted to the well-known basic STRIPS formalism, Σ_2^p-complete when we admit boolean logical operators and undecidable when the full power of the planning language ADL is permitted.

Introduction

Formal action models play a central role in several subfields of AI, e.g., in automated planning. Generally speaking, an action model describes how and under which circumstances an action transforms a state of the world into another one. An *operator* is a schematic action description using *parameters* as placeholders for objects. The instantiation of these placeholders with domain objects generates actions.

In this paper, we propose two notions of operator subsumption. The first is based on *action abstraction*, which supports reuse and inheritance of operators. The basic intention is to treat operators similar to concepts in ontology reasoning. In that sense, precondition and effects are properties of actions and an action can be specialized by refining such a property. For example, a *move* action would be considered as more general than a *move-by-truck* action. This notion of an *action abstraction* subsumption is quite different from the other notion, the *applicability* subsumption, which considers one action as more general as another one if the former can be used in more situations than the latter. In this case the more general action must have the same or a weaker precondition and the same postcondition. Interestingly, it turns out that both kinds of subsumption are very similar in design when their abstract ideas are mapped to formal definitions.

Since its introduction in 1998 by McDermott and colleagues, the *Planning Domain Definition Language PDDL* (Fox and Long 2003) has become the de facto standard for representing planning domains. Numerous PDDL domains have been specified which often model the same or similar application areas. It would certainly be useful to automatically detect coherences and similarities between those domains. Beside the comparison of types and their hierarchical structure, the possibility to relate operators in a suitable way is necessary to compare two domains. Such a comparison, however, has hitherto been limited to testing the *verbatim equality* of action models due to the lack of a formal concept of *subsumption* between action models. At this point, both the applicability and the abstraction subsumption could be utilized.

Knowledge about subsumption hierarchies can be useful in various ways: When designing new planning domains, abstraction subsumption hierarchies help finding related domains and enable the reuse of existing components, e.g., by means of *inheritance* from more abstract operator models. Also, applicability subsumption hierarchies can be used to find unnecessary actions. In large domains, a hierarchical ordering between operators similar to a type hierarchy would be very helpful during the design process as well as for maintainers of the domain.

If a planning model already supports hierarchically structured operator descriptions, as is the case in HTN planning (Erol, Hendler, and Nau 1994; Nau et al. 2001), subsumption can be used to automatically *verify* a planning domain: An HTN method decomposition can be regarded as a macro operator which must be a specialization of the original method. Even if the planning process is not hierarchical, algorithms can exploit the subsumption hierarchy to generate heuristics. (Helmert 2006; Knoblock 1994).

For both kinds of subsumption, we investigate the computational complexity of the subsumption problem for three

*This work has been supported by the German Federal Ministry of Education and Research (BMBF) under grant no. 01IME01-ALU (DESIRE) and by the EU under grant no. FP6-004250 (CoSy).

important special cases in which the expressiveness of the involved operators is restricted: Basic STRIPS (Fikes and Nilsson 1971), basic STRIPS permitting boolean logical operators in the precondition and general ADL (Pednault 1989). While our approach is based on a formalism which encodes precondition and effects of an action explicitly, like it is the case in AI Planning, it should be noted that it is not limited to this field, but can be transferred to other formalisms which use a similar solution to the frame problem, e.g., the situation calculus (Reiter 2001) using a suitable mapping (Eyerich et al. 2006).

The remainder of the paper is structured as follows. After discussing related work, we introduce two kinds of subsumption relation between operators that are suitable for modeling and verifying abstraction hierarchies between actions and operators. Next, we state syntactic criteria which can be used to decide the subsumption problems for two operators. Afterwards, we use these criteria to investigate the computational complexity of the subsumption problem for the three special cases STRIPS, boolean logical operators and ADL. We conclude with a summary and an outlook on future work.

Related Work

Reasoning about hierarchies in planning domains is by no means new. As described in the survey paper by Gil (2005), most of the work in this area is connected with description logics (Baader et al. 2003).

For example, RAT (Heinsohn et al. 1992), CLASP (Devanbu and Litman 1991) and T-REX (Weida and Litman 1992) are systems for representing actions using description logics. In all these systems, one can describe actions and reason about plans using a subsumption relation over actions borrowed from description logics. Similarly, Liebig and Rösner (Liebig and Rösner 1997) have designed a framework for actions, in which it is possible to reason about subsumption relations between operators. However, the expressivity within this framework is restricted to disjunctions and conjunctions of feature chain agreements (equality of role fillers of functional roles). Also Kemke (2003) has formulated a formal theory for describing STRIPS-like actions using description logics and has addressed the issue of subsumption between actions. Together with Walker she has developed a planning algorithm integrating action abstractions and plan decomposition hierarchies (Kemke and Walker 2006).

In contrast to these approaches, we do not employ description logics to define the subsumption relation between operators. In fact, we abstract from parameter names in operators and base the subsumption relation between operators on the subsumption relation between the corresponding actions, which does not seem to be possible when mapping operators into a description logic framework. Furthermore, we specify a sound and complete operator subsumption algorithm and analyze the computational complexity of the operator subsumption problem.

Independently from description logic approaches, in planning mainly two kinds of abstraction techniques are considered. On the one hand, actions are combined in order to generate new actions, i.e. macros (Botea et al. 2005) or HTN tasks (Erol, Hendler, and Nau 1994). On the other hand, abstractions can be formed by omitting details from the representation of the world state. This can lead to *reduced models* (Knoblock 1994) or *relaxed models* (Sacerdoti 1974). Both variants are limited to STRIPS-like formalisms and focus on the *generation* of abstractions. Our approach, however, can *detect* subsumption relations between operators, which are in addition more expressive than STRIPS.

Prerequisites

Before we can propose subsumption relations between operators, we have to introduce some basic notions.

A *type* T is a set variable. An *object-domain* $\Delta = \langle \Lambda, \theta \rangle$ consists of a finite set of objects (individuals, constants) Λ and mapping θ, assigning a subset of Λ to some types (to all types not in the domain of θ the empty set is assigned). There is a special type T in each object-domain containing all objects of Λ (so it always holds that $\theta(\mathrm{T}) = \Lambda$). An object-domain together with a set of typed relation schemes ρ forms a domain. A typed relation scheme consists of a relation symbol and a fixed number of typed variables, which means that these variables can only be instantiated by objects of the appropriate types.

Definition 1. An *atom* of a domain $\langle \Delta, \rho \rangle$ is formed by instantiating the variables of a predicate scheme from ρ with objects of Δ of the appropriate types. A *state* is a complete truth assignment to the set of atoms.

Alternatively, we can identify a state with the set of atoms that are true in this state (assuming the closed world assumption and presuming that the domain closure assumption as well as the unique name assumption are satisfied).

Definition 2. An *action* a is a tuple $a = \langle pre(a), \mathit{eff}(a) \rangle$. $pre(a)$ (the *precondition* of a) is a FOL formula without free variables and without functional terms[1]. $\mathit{eff}(a)$ (the *effect* of a) is a conjunction of the form $\bigwedge_{1 \leq i \leq n} (c_i \rhd e_i)$ with finite n, whereby c_i is a FOL formula without free variables and functional terms and e_i is a ground literal.

For an action a, we refer to the positive e_i by $pos(a)$ and to the negative e_i by $neg(a)$.

An action a is applicable in a given state s iff $s \models pre(a)$. The result from applying a in a state s, denoted by $R(a, s)$, is $R(a, s) = s \setminus \{e_i | s \models c_i \wedge e_i \in \text{neg(a)}\} \cup \{e_i | s \models c_i \wedge e_i \in \text{pos(a)}\}$ if a is applicable in s, undefined otherwise.

Definition 3. A *hierarchy of types* $\boldsymbol{T}$ is a finite set of rules of the form $T \Rightarrow T'$, whereby T and T' are types. An object-domain $\Delta = \langle \Lambda, \theta \rangle$ *respects* a hierarchy of types $\mathbf{T}$ iff $\theta(T_1) \subseteq \theta(T_2)$ holds for each rule $T_1 \Rightarrow T_2$ in $\mathbf{T}$.

Definition 4. An *operator* A is a triple $A = \langle pre(A), \mathit{eff}(A), types_A \rangle$. $pre(A)$ (the *precondition* of A) is a FOL formula without functional terms (variables are allowed now). $\mathit{eff}(A)$ (the *effect* of A) is a conjunction of the form $\bigwedge_i (V_i.(C_i \rhd E_i))$, whereby C_i is a FOL formula without

[1] With *functional terms* we mean only those terms with arity ≥ 1, the use of constants is allowed, of course.

functional terms and E_i is a literal. V_i is a set of typed variables (written as $v : T$ where T is the type of variable v). $types_A$ is a mapping from all free variables of $pre(A)$ and *eff*(A), which are not occurring in a V_j, to types. We refer to the type of a variable in a set V_i as $T_i(v_i)$.

The variables in the domain of $types_A$ are called the *schema variables* or *parameters* $\mathcal{S}(A)$ of A.

An operator A represents a set of actions for each object-domain Δ.

Definition 5. By first choosing an admissible variable assignment α for $\mathcal{S}(A)$ (at this point admissible means that $\forall x \in \mathcal{S}(A) : \alpha(x) \in \theta(types_A(x))$) and afterwards introducing new effects (c_{ij}, e_{ij}) for each admissible variable assignment β_j for V_i having each $y \in V_i$ in $(C_i \rhd E_i)$ replaced by $\beta_j(y)$ (this time admissible means $\forall y \in V_i : \beta_j(y) \in \theta(T_i(y))$), we obtain an action $a = \langle pre(a), \textit{eff}(a)\rangle$ represented by A in Δ. We call α the generating assignment of a and write $\alpha_a^{A,\Delta}$ for it.

The set of actions which is represented by A in Δ is called $Actions_\Delta(A)$.
As an example, we assume to have the following move operator M:

$$\langle At(x,y) \wedge Connected(y,z),$$
$$(\top \rhd \neg At(x,y)) \wedge (\top \rhd At(x,z)) \wedge$$
$$(\{b : Ball\}.Carry(x,b) \rhd \neg At(b,y)) \wedge$$
$$(\{b : Ball\}.Carry(x,b) \rhd At(b,z)),$$
$$\{x \mapsto Agent, y \mapsto Location, z \mapsto Location\}\rangle$$

and an object-domain

$$\Delta = \langle \{a_1, a_2, l_1, l_2, l_3, b_1, b_2\}, \{Agent \mapsto \{a_1, a_2\},$$
$$Location \mapsto \{l_1, l_2, l_3\}, Ball \mapsto \{b_1, b_2\}\}\rangle$$

By choosing the admissible variable assignment $\alpha_{m_1}^{M,\Delta} = \{x \mapsto a_1, y \mapsto l_1, z \mapsto l_2\}$ and replacing the universally quantified effects as described in definition 5, we obtain the action m_1:

$$\langle At(a_1,l_1) \wedge Connected(l_1,l_2),$$
$$(\top \rhd \neg At(a_1,l_1)) \wedge (\top \rhd At(a_1,l_2)) \wedge$$
$$(Carry(a_1,b_1) \rhd \neg At(b_1,l_1)) \wedge$$
$$(Carry(a_1,b_2) \rhd \neg At(b_2,l_1)) \wedge$$
$$(Carry(a_1,b_1) \rhd At(b_1,l_2) \wedge$$
$$(Carry(a_1,b_2) \rhd At(b_2,l_2)\rangle$$

Analogous all other possible actions of $Actions_\Delta(M) = \{m_1, \ldots, m_{18}\}$ are generated.

We call an operator A *defined* for an object-domain $\Delta = \langle \Lambda, \theta\rangle$, if $|\theta(types_A(x))| \geq 1$ for all $x \in \mathcal{S}(A)$ (an operator which is defined in an object-domain represents at least one action therein).

To facilitate the investigation of complexity issues, operators are separated into different classes in accordance with the expressiveness of their precondition and effects:

Definition 6. If we allow only for the use of conjunctions of atoms in $pre(A)$ and conjunctions of literals in *eff*(A) and additionally enforce any c_i to be equivalent to $\top$ and any V_i to be the empty set, we obtain the class $\mathcal{A}_{STRIPS}$. The class $\mathcal{A}_{BOOLE}$ is defined by allowing conjunctions also in the c_i and negations and disjunctions in $pre(A)$ and the c_i. If we relax these restrictions further by allowing existential and universal quantification in $pre(A)$ and the c_i as well as conditional effects (which means that the V_i can contain any finite number of typed variables), we obtain the class $\mathcal{A}_{ADL}$.

As the names suggest, these definitions are chosen in a way that $\mathcal{A}_{STRIPS}$ captures the expressiveness of the planning formalism STRIPS (Fikes and Nilsson 1971) in its basic variant (only operators and background theories) and $\mathcal{A}_{ADL}$ the one of the planning language ADL (Pednault 1989).

Subsumption of actions and operators

Next, we define what exactly it means for an action to be subsumed by another. We propose two different notions of subsumption, namely *abstraction* subsumption and *applicability* subsumption. In spite of the fact that the basic motivation for them seems to be quite orthogonal, it turns out that the only difference between the two lies in how effects are dealt with.

Abstraction subsumption

The basic intention is to treat operators similar as concepts in ontology reasoning. In that sense, precondition and effects are properties of actions. Intuitively, a specialization a of an action b should inherit all properties of b. More precisely, if $s \models pre(a)$ holds for a state s, then $s \models pre(b)$ should hold, too. Furthermore, all changes from s to $R(b,s)$ after the application of b should arise after the application of a as well. To specialize an action, one can add a new parameter, strengthen the precondition, add a new effect (or relaxing the effect condition of a given effect), and restrict a parameter to a subtype.

If we put aside conditional effects for a moment, these ideas can be mapped straight-forwardly to the following definition.

Definition 7. An action a is *abstractly subsumed* by an action b ($a \subseteq_{as} b$) iff

1. $pre(a) \Rightarrow pre(b)$
2. $\forall i(e_i \in eff(b) \Rightarrow e_i \in eff(a))$

If we now take conditional effects into account again, we have to extend the second condition in definition 7. It no longer suffices that each effect e_i of b has to occur in $eff(b)$ as well, rather we additionally have to make sure that the effect condition of e_i in $eff(b)$ is at least as strong as in $eff(a)$. This extension is captured in definition 8.

To keep the definition as simple as possible, we assume the effects $eff(a)$ of an action a to be some kind of "normalized" in the sense that each e_i occurs at most once in $eff(a)$.

Definition 8. An action a is *abstractly subsumed* by an action b ($a \subseteq_{as} b$) iff

1. $pre(a) \Rightarrow pre(b)$
2. $\forall i(e_i \in eff(b) \Rightarrow \exists j(e_j \in eff(a) \wedge (e_j = e_i) \wedge (c_i \Rightarrow c_j)))$

Frequently, there are situations in which predicates in two domains are named differently but have the same intended meaning (e.g. $Connected(x, y)$ and $Conn(x, y)$). It would be very useful when one would be allowed to state the equivalence of these predicates. A similar situation occurs when one wants a predicate to follow logically from another one. For instance, take our last example: If one wants to distinguish between different forms of connectivity, he might use the predicate $Connected\_by\_Rail(r, s)$ in place of $Connected(x, y)$. For this reason, we extend our definition incorporating axioms of the following form:

$$\forall \vec{x}(P(\vec{x}) \Rightarrow Q(\vec{y}))$$

whereby the variables in $\vec{y}$ are a subset of the variables in $\vec{x}$.

We call such an axiom a predicate inclusion axiom (PIA) and a conjunction of such axioms a predicate inclusion hierarchy **P**. To integrate predicate inclusion axioms into our definition we have to extend the notion of "normalization": An action a' is a *normalization* for an action a and a predicate inclusion hierarchy **P**, iff

- for each two effects $(c_i \rhd e_i)$ and $(c_j \rhd e_j)$, $c_i \wedge \mathbf{P} \Rightarrow c_j$ holds whenever $e_i \wedge \mathbf{P} \Rightarrow e_j$ holds and
- $R(a, s) = R(a's)$ for each state s

Hence follows the extended definition:

Definition 9. An action a is *abstractly subsumed* by an action b regarding a predicate inclusion hierarchy **P** ($a \subseteq^{\mathbf{P}}_{as} b$) iff

1. $pre(a) \wedge \mathbf{P} \Rightarrow pre(b)$
2. $\forall i(e_i \in eff(b) \Rightarrow \exists j(e_j \in eff(a) \wedge (e_j \wedge \mathbf{P} \Rightarrow e_i)) \wedge (c_i \wedge \mathbf{P} \Rightarrow c_j))$

For an example take the two actions

$$g = \langle P(c), (C_1(c) \vee C_2(c)) \rhd E(c) \rangle$$

and

$$s = \langle P(c) \wedge Q(c), \big(C_1(c) \rhd E(c)\big) \wedge \big(C_3(c) \rhd F(c)\big) \rangle.$$

$s \subseteq_{as} g$ does not hold. But if we consider the predicate inclusion hierarchy

$$\mathbf{P} = (\forall x(C_2(x) \Rightarrow C_3(x)) \wedge \forall x(F(x) \Rightarrow E(x))).$$

and the normalization s' of s

$$s' = \langle P(c) \wedge Q(c), \big((C_1(c) \vee C_3(c)) \rhd E(c)\big) \wedge \big(C_3(c) \rhd F(c)\big) \rangle,$$

$s' \subseteq^{\mathbf{P}}_{as} g$ holds.

Since operators represent sets of actions, it seems to be quite natural to define the subsumption of operators on top of the subsumption of their represented actions. However, operator subsumption should not only depend on a single object-domain but on any in which the involved operators are defined.

When one starts with the represented actions of an operator A and wants to specialize them, there should be a specialized action in the set of represented actions of the specialization of A for each single action in A. Analogous, if one wants to generalize the operator, there should be a generalized action in the set of represented actions of the generalization of A for each single action in A. Therefore, the following definition seems to be natural and intuition capturing:

Definition 10. An operator A is *abstractly subsumed* by an operator B ($A \subseteq_{as} B$) iff there are mappings

$$f_1 : Actions_\Delta(A) \mapsto Actions_\Delta(B)$$

and

$$f_2 : Actions_\Delta(B) \mapsto Actions_\Delta(A)$$

such that

1. $\forall a \in Actions_\Delta(A) : a \subseteq^{\mathbf{P}}_{as} f_1(a)$
2. $\forall b \in Actions_\Delta(B) : f_2(b) \subseteq^{\mathbf{P}}_{as} b.$

for each object-domain Δ in which A and B are defined.

For an example assume we have additionally to M an operator for moving with a truck MWT:

$$\begin{array}{c}\langle At(q, r) \wedge Connected(r, s) \wedge At(t, r),\\ (\top \rhd \neg At(q, r)) \wedge (\top \rhd At(q, s)) \wedge\\ (\top \rhd \neg At(t, r)) \wedge (\top \rhd At(t, s)) \wedge\\ (\{b : Ball\}.Carry(q, b) \rhd \neg At(b, r)) \wedge\\ (\{b : Ball\}.Carry(q, b) \rhd At(b, s)),\\ \{q \mapsto Agent, r \mapsto Location,\\ s \mapsto Location, t \mapsto Truck\}\rangle\end{array}$$

Due to the similar structure of M and MWT it is easy to verify that $M \subseteq_{as} MWT$ holds: If a is an action in $Actions_\Delta(M)$ for an arbitrary object-domain Δ and $\alpha_a^{M,\Delta}$ its generating assignment, any assignment $\alpha_b^{MWT,\Delta}$ with $\alpha_b^{MWT,\Delta}(q) = \alpha_a^{M,\Delta}(x)$, $\alpha_b^{MWT,\Delta}(r) = \alpha_a^{M,\Delta}(y)$ and $\alpha_b^{MWT,\Delta}(s) = \alpha_a^{M,\Delta}(z)$ generates an action b such that $a \subseteq_{as} b$ holds. Analogous, one finds an appropriate action a for each action b from $Actions_\Delta(M)$.

Our definition so far requires all shared parameters of two operators A and B to be of the same type in order to make $A \subseteq_{as} B$ true. However, according to our intuition, it should be possible to restrict the type of a parameter in an operator to obtain a specialization. An example is the subsumption relation between $A = \langle P(x), (), \{x \mapsto T_1\}\rangle$ and $B = \langle P(x), (), \{x \mapsto T_2\}\rangle$. $A \subseteq_{as} B$ does not hold due to the different types of the two x. To see this, consider for example the object-domain $\Delta = \langle \{c_1, c_2\}, \{T_1 \mapsto \{c_1\}, T_2 \mapsto \{c_1, c_2\}\}\rangle$. There is no action a in $Actions_\Delta(A)$ such that $a \subseteq_{as} \langle P(c_2), ()\rangle$ holds and therefore $A \subseteq_{as} B$ does not hold.

However, if we declare T_1 to be a subtype of T_2, $A \subseteq_{as} B$ should hold according to our intuition. This can be formalized by restricting definition 10 to object-domains respecting the given type hierarchy and weaken the second condition in definition 10 by only considering actions with objects of the appropriate subtypes. These ideas are formalized in definition 11.

Definition 11. An operator A is *abstractly subsumed* by an operator B regarding a predicate inclusion hierarchy **P** and a type hierarchy **T** ($A \subseteq^{\mathbf{P},\mathbf{T}}_{as} B$) iff there are mappings

$$f_1 : Actions_\Delta(A) \mapsto Actions_\Delta(B)$$

and

$$f_2 : Actions_{\Delta_\mathbf{T}}(B) \mapsto Actions_\Delta(A)$$

such that

1. $\forall a \in Actions_\Delta(A) : a \subseteq^\mathbf{P}_{as} f_1(a)$
2. $\forall b \in Actions_\Delta(B) : f_2(b) \subseteq^\mathbf{P}_{as} b$.

for each object-domain Δ respecting $\mathbf{T}$ and in which A and B are defined.

Thereby $\Delta_\mathbf{T}$ is constructed by reducing Δ according to the type hierarchy $\mathbf{T}$: We start with Δ and reduce it as follows: For each $c \in \Delta$, if there is a rule $T_1 \Rightarrow T_2$ in $\mathbf{T}$, $c \in \theta(T_2)$ and $c \notin \theta(T_1)$, remove c from $\theta(T_2)$.

Applicability subsumption

Sometimes another form of action subsumption is used known as *plug-in* subsumption or *applicability* subsumption. Here, an action a is defined to be subsumed by an action b if a can be replaced by b in each situation. This means, that the first condition of definition 8 stays the same, but the second condition is that the effects in both actions have to be exactly the same.

Definition 12. An action a is *applicably subsumed* by an action b regarding a predicate inclusion hierarchy $\mathbf{P}$ ($a \subseteq^\mathbf{P}_{ps} b$) iff

1. $pre(a) \wedge \mathbf{P} \Rightarrow pre(b)$
2. $\forall i(e_i \in eff(b) \Rightarrow \exists j(e_j \in eff(a) \wedge (e_j \wedge \mathbf{P} \Leftrightarrow e_i \wedge \mathbf{P})) \wedge (c_i \wedge \mathbf{P} \Leftrightarrow c_j \wedge \mathbf{P}))$

As for abstract subsumption, we define the subsumption of operators on top of the subsumption of actions:

Definition 13. An operator A is *applicably subsumed* by an operator B regarding a predicate inclusion hierarchy $\mathbf{P}$ and a type hierarchy $\mathbf{T}$ ($A \subseteq^{\mathbf{P},\mathbf{T}}_{ps} B$) iff there are mappings

$$f_1 : Actions_\Delta(A) \mapsto Actions_\Delta(B)$$

and

$$f_2 : Actions_{\Delta_\mathbf{T}}(B) \mapsto Actions_\Delta(A)$$

such that

1. $\forall a \in Actions_\Delta(A) : a \subseteq^\mathbf{P}_{ps} f_1(a)$
2. $\forall b \in Actions_\Delta(B) : f_2(b) \subseteq^\mathbf{P}_{ps} b$.

for each object-domain Δ respecting $\mathbf{T}$ and in which A and B are defined.

Syntactic criteria for operator subsumption

After stating these definitions concerning the subsumption of operators following our intuition, the question arises how to decide in general whether an operator A is subsumed by another one B. According to definition 11, *each* object-domain has to be considered to answer this question. Naturally, these are infinitely many. Hence, we are interested in syntactic criteria as the basis on which to decide this problem.

The operator resulting from the replacement of one or more pairwise different variable names $y_1, y_2, y_3, \ldots \in \mathcal{S}(A)$ by pairwise different variable names $x_1, x_2, x_3, \ldots \notin \mathcal{S}(A)$ is called the substitution A^s concerning the parameter replacement $s = [\frac{x_1}{y_1}\frac{x_2}{y_2}\frac{x_3}{y_3}\ldots]$.

Theorem 1. *$A \subseteq^{\mathbf{P},\mathbf{T}}_{as} B$ holds for two operators A and B without universally quantified effects regarding a predicate inclusion hierarchy $\mathbf{P}$ and a type hierarchy $\mathbf{T}$ iff there is a substitution A^s of A, such that*

(S1) $pre(A^s) \wedge \mathbf{P} \Rightarrow pre(B)$

(S2) $\forall i(E_i \in eff(B) \Rightarrow \exists j(E_j \in eff(A^s) \wedge (E_j \wedge \mathbf{P} \Rightarrow E_i)) \wedge (C_i \wedge \mathbf{P} \Rightarrow C_j))$

(S3) $types_{A^s}(x)$ *is a subtype of* $types_B(x)$ *according to* $\mathbf{T}$ *for all* $x \in \big(\mathcal{S}(A^s) \cap \mathcal{S}(B)\big)$

Proof. "⇒"
Without loss of generality we assume that there is no shared variable name in A and B and that actions are noted in the syntactical form of the operators (e.g. the action a resulting from the operator $A = \langle P(x) \wedge P(y), (), \{x \mapsto \mathrm{T}, y \mapsto \mathrm{T}\}\rangle$ and the assignment $\alpha^{A,\Delta}_a = \{x \mapsto c_1, y \mapsto c_1\}$ is noted as $\langle P(c_1) \wedge P(c_1), ()\rangle$ and not in its simplified version $\langle P(c_1), ()\rangle$).

We consider an arbitrary object-domain $\Delta = \langle \Lambda, \theta \rangle$ which satisfies the following conditions:

(T1) $|\theta(T)| \geq n$ for all types T in the range of $types_A$ and $types_B$, whereby n=max(# parameter of A, # parameter of B).

(T2) Δ respects $\mathbf{T}$.

(T3) If $T_1 \Rightarrow T_2$ does not hold in $\mathbf{T}$ for any two types T_1 and T_2, then there are no common objects in T_1 and T_2.

Due to T1 there is at least one action $b \in Actions_\Delta(B)$ such that $\alpha^{B,\Delta}_b$ assigns pairwise different objects to the parameters of B. We choose one of these actions and refer to $f_2(b)$ as a.

In the following, we construct a substitution A^s of A from $\alpha^{A,\Delta}_a$, $\alpha^{B,\Delta}_b$, A and B such that S1-S3 holds for A^s and B:

We compare the parameters $\vec{x} \in \mathcal{S}(A)$ and $\vec{y} \in \mathcal{S}(B)$ pairwise until we find a pair x, y such that $\alpha^{A,\Delta}_a(x) = \alpha^{B,\Delta}_b(y) = c$. If there is no other parameter $x' \in \mathcal{S}(A)$ such that $\alpha^{A,\Delta}_a(x) = c$, we replace c in a and b by y resulting in two "actions" (they are not really actions since they now contain variables, but for this purpose we can treat them as such) a' and b' such that $a' \subseteq_{as} b'$ still holds.

If there are other parameters $x', x'', \ldots \in \mathcal{S}(A)$ such that $\alpha^{A,\Delta}_a(x) = \alpha^{A,\Delta}_a(x') = \alpha^{A,\Delta}_a(x'') = \ldots$, then there must be at least one x, such that we can replace all occurrences of c in b and the occurrences of c in a at the places were x stands in A resulting in two "actions" a' and b' and $a' \subseteq_{as} b'$ holds.

Then we replace a with a' and b with b' and start again searching for two parameters which were replaced by the same object. This procedure is pursued until no such parameters can be found. If a parameter $x \in \mathcal{S}(A)$ is assigned c' by $\alpha^{A,\Delta}_a$ and there is no variable $y \in \mathcal{S}(B)$, which is assigned c' by $\alpha^{B,\Delta}_b$, we replace c' in a by x.

If a variable $y \in \mathcal{S}(B)$ is assigned c'' by $\alpha^{B,\Delta}_b$ and there is no variable $x \in \mathcal{S}(A)$, which is assigned c'' by $\alpha^{A,\Delta}_a$, we replace c'' in b by y.

In this way we receive operators A^s and B' satisfying:

1. $B' = B$.
2. A^s is a substitution of A.
3. S1 and S2 follow from $a \subseteq_{as} b$ and the kind of our replacements.
4. When x is a shared parameter of A^s and B', the same object must have stood in A and B at this place, so S3 holds due to T2 and T3.

"$\Leftarrow$"

First notice that if $\Phi \Rightarrow \Psi$ holds for two FOL formulas Φ and Ψ and we substitute a free variable in Φ and Ψ by a constant resulting in Φ' and Ψ', then $\Phi' \Rightarrow \Psi'$ holds as well.(*)

Let $\Delta = \langle \Lambda, \theta \rangle$ be an arbitrary object-domain and a an arbitrary action in $Actions_\Delta(A^s)$ such that $\alpha_a^{A,\Delta}$ assigns different objects to each parameter $x \in \mathcal{S}(A)$. We choose the generating assignment $\alpha_b^{B,\Delta}$ depending on $\alpha_a^{A,\Delta}$: $\alpha_b^{B,\Delta}(x) := \alpha_a^{A,\Delta}(x)$, if $x \in \mathcal{S}(A^s) \cap \mathcal{S}(B)$ and $\alpha_b^{B,\Delta}(x)$ arbitrary (but different to the other objects chosen so far), if $x \notin \mathcal{S}(A^s) \cap \mathcal{S}(B)$. Due to (*), $a \subseteq_{as} f_1(a)$ holds. Similarly, we can construct an action b such that $f_2(b) \subseteq_{as} b$ holds. Since Δ, a and b were chosen randomly, $A^s \subseteq_{as} B$ follows and therefore $A \subseteq_{as} B$ holds. □

Theorem 2. *$A \subseteq_{ps}^{\mathbf{P},\mathbf{T}} B$ holds for two operators A and B without universally quantified effects regarding a predicate inclusion hierarchy **P** and a type hierarchy **T** iff there is a substitution A^s of A, such that*

(S1) $pre(A^s) \wedge \boldsymbol{P} \Rightarrow pre(B)$

(S2) $\forall i(E_i \in eff(B) \Rightarrow \exists j(E_j \in eff(A^s) \wedge (E_j \wedge \boldsymbol{P} \Leftrightarrow E_i \wedge \boldsymbol{P})) \wedge (C_i \wedge \boldsymbol{P} \Leftrightarrow C_j \wedge \boldsymbol{P}))$

(S3) $types_{A^s}(x)$ *is a subtype of* $types_B(x)$ *according to* **T** *for all* $x \in \big(\mathcal{S}(A^s) \cap \mathcal{S}(B)\big)$

The proof of theorem 2 is analogous to the one of theorem 1.

The problem to decide whether there is an abstraction subsumption relation between two given operators regarding a given predicate inclusion hierarchy **P** and a given type hierarchy **T** is named AS. The three special cases in which A and B are of class $\mathcal{A}_{STRIPS}$, $\mathcal{A}_{BOOLE}$ and $\mathcal{A}_{ADL}$ are called AS_{STRIPS}, AS_{BOOLE} and AS_{ADL}, respectively. A similar naming convention is used for applicability subsumption: In that case, the base problem is named PS and the same indices are used to refer to the respective subproblems.

Complexity of AS_{STRIPS} and PS_{STRIPS}

Theorem 3. *AS_{STRIPS} is $\mathcal{NP}$-complete.*

Proof. $\mathsf{AS}_{STRIPS} \in \mathcal{NP}$:
We guess an admissible substitution A^s of A and verify that $A^s \subseteq_{as} B$ holds. Due to theorem 1 it suffices to show S1-S3 for A and B. Since A and B (and thus A^s) are in class $\mathcal{A}_{STRIPS}$, for showing S1 it suffices to show that for each predicate $P(\vec{x})$ in $pre(B)$ there is a predicate $P'(\vec{x})$ in $pre(A^s)$, whereby $P = P'$ or $P' \Rightarrow P$ holds in **P** and $types_B(x_i) = types_A(x_i)$ or $types_B(x_i) \Rightarrow types_A(x_i)$ holds in **T**.

Let k be the number of predicate inclusion axioms in **P**, l the number of rules in **T**, m the number of predicates in $pre(A^s)$, n the number of predicates in $pre(B)$ and o the number of arguments of the predicate with the most arguments. We compare the name of a predicate $P(\vec{x}) \in pre(B)$ with the names of all predicates $P'(\vec{x}) \in pre(A^s)$ and those predicate names following from P' in **P** (these are $O(k^3 * m)$ comparisons). If $P = P'$ or P follows from P' in **P**, we proceed with testing whether $types_B(x_i) = types_A(x_i)$ or $types_B(x_i) \Rightarrow types_A(x_i)$ holds in **T** for all i (these are $O(o * l^3)$ comparisons). If this is satisfied for all i, we mark $P(\vec{x})$.

This is done for each predicate $P(\vec{x})$ in $pre(B)$ (and thus n times). $pre(A^s) \Rightarrow pre(B)$ holds iff each predicate in $pre(B)$ is marked after this procedure. Thus, there are $O(n * (k^3 * m + o * l^3))$ many comparisons all in all which is polynomial in the size of A, B, **T** and **P**.

As no conditional effects are allowed in $\mathcal{A}_{STRIPS}$, S2 is simplified to $\forall i(E_i \in eff(B) \Rightarrow \exists j(E_j \in eff(A^s) \wedge (E_j \wedge \mathbf{P} \Rightarrow E_i)$. Thus, the procedure used for the verification of S1 also works for the verification of S2 (and is therefore doable in polynomial time as well).

Obviously, S3 is verifiable for each parameter polynomial in the size of the type hierarchy.

Hence, $\mathsf{AS}_{STRIPS} \in \mathcal{NP}$ holds.

Hardness is proved by reducing the subgraph isomorphism decision problem SI to AS_{STRIPS}.

SI is the problem to decide for two given undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ whether the former can be embedded isomorphic into the latter. At this point isomorphic embedding means that there exists a graph $G_2' = (V_2', E_2')$ whereby $V_2' \subseteq V_2$ and $E_2' = E_2 \cap \{(u, v) | u, v \in V_2\}$ and a bijective mapping $f : V_1 \mapsto V_2'$, such that $(u, v) \in E_1 \Leftrightarrow (f(u), f(v)) \in E_2'$.

Without loss of generality, we assume the nodes of G_1 to be named as x_i^1 for $1 \leq i \leq |V_1|$ and they of G_2 as x_j^2 for $1 \leq j \leq |V_2|$.

G_k is mapped to an operator $A_k \in \mathcal{A}_{STRIPS}$ as follows:

1. We introduce a parameter of type T for each node $x_i^k \in V_k$.
2. We add the predicates $E^+(x_i^k, x_j^k)$ and $E^+(x_j^k, x_i^k)$ for each edge $(x_i^k, x_j^k) \in E_k$ to $pre(A_k)$.
3. We add the predicates $E^-(x_i^k, x_j^k)$ and $E^-(x_j^k, x_i^k)$ for each pair of nodes x_i^k, x_j^k, for which $(x_i^k, x_j^k) \notin E_k$ to $pre(A_k)$.
4. There are no effects in A_k.

Next, we show that G_1 can be embedded isomorphic into G_2 iff $A_2 \subseteq_{as} A_1$ holds.

Let us first consider the case that G_1 can be indeed embedded isomorphic into G_2: Then there exists a graph $G_2' = (V_2', E_2')$ such that $V_2' \subseteq V_2$ and $E_2' = E_2 \cap \{(u, v) | u, v \in V_2\}$ and there exists a bijective mapping $f : V_1 \mapsto V_2'$, such that: $(u, v) \in E_1 \Leftrightarrow (f(u), f(v)) \in E_2'$. Furthermore, due to our mapping, there are exactly two predicates for each edge $(x_i^1, x_j^1) \in E_1$ in $pre(A_1)$: $E^+(x_i^1, x_j^1)$ and $E^+(x_j^1, x_i^1)$. For each pair of nodes x_i^1, x_j^1 for which there is no edge in E_1, exactly the two predicates $E^-(x_i^1, x_j^1)$ and

$E^-(x_j^1, x_i^1)$ exist in $pre(A_1)$. For E_2 and $pre(A_2)$ things are analogous.

Thus, when we substitute $f(x_i^1)$ by x_i^1 for all i, each $E^+(x_i^1, x_j^1) \in pre(A_1)$ and each $E^-(x_i^1, x_j^1) \in pre(A_1)$ also occurs in $pre(A_2)$. Since $A_1, A_2 \in \mathcal{A}_{STRIPS}$ and there are neither effects nor special types in A_1 and A_2 (and therefore S2 and S3 are satisfied trivially), $A_2 \subseteq_{as} A_1$ follows directly.

Let us now turn to the case in which G_1 cannot be embedded isomorphic into G_2. In this case, there cannot be a graph $G_2' = (V_2', E_2')$ such that $V_2' \subseteq V_2$ and $E_2' = E_2 \cap \{(u, v) | u, v \in V_2\}$ such that there is a bijective mapping $f : V_1 \mapsto V_2'$ such that $(u, v) \in E_1 \Leftrightarrow (f(u), f(v)) \in E_2'$. Hence, either there is an edge $(x_i^1, x_j^1) \in E_1$ for each subgraph G_2' from G_2 and each mapping $f : V_1 \mapsto V_2'$ such that $(f(x_i^1), f(x_j^1)) \notin E_2$ or an edge $(x_i^1, x_j^1) \in E_1$ is lacking, such that $(f(x_i^1), f(x_j^1)) \in E_2$.

Thus, the mapping generates either a predicate $E^+(x_i^1, x_j^1) \in pre(A_1)$, such that $E^+(f(x_i^1), f(x_j^1)) \notin pre(A_2)$ or it generates a predicate $E^-(x_i^1, x_j^1) \in pre(A_1)$, such that $E^-(f(x_i^1), f(x_j^1)) \notin pre(A_2)$. For this reason, $f(x_i^1)$ cannot be substituted by x_i^1 for at least one i (otherwise there would be a predicate in $pre(A_1)$ which does not occur in $pre(A_2)$). Therefore, there is no substitution such that each predicate of $pre(A_1)$ also occurs in $pre(A_2)$. Since $A_1, A_2 \in \mathcal{A}_{STRIPS}$ and there are neither effects nor special types in A_1 and A_2 (and therefore S2 and S3 are satisfied trivially), $A_2 \not\subseteq_{as} A_1$ follows directly.

Obviously, the mapping is polynomial in the size of the encoding of the graphs and is therefore indeed a polynomial many-one reduction.

Example 1. For an example of the mapping used in the proof of theorem 3 consider G_1 and G_2 from figure 1. Obviously, G_1 can be embedded isomorphic into G_2 (too see this, take a look at figure 2 and confirm that $f = \{x_1 \mapsto y_5, x_2 \mapsto y_4, x_3 \mapsto y_1, x_4 \mapsto y_2\}$ is a bijective mapping between $V_2' = \{y_1, y_2, y_4, y_5\}$ and V_1 satisfying the conditions stated above.

The mapping maps G_1 and G_2 to two operators A_1 and A_2, respectively:

$$\begin{aligned} A_1 = \langle & E^+(x_1, x_2) \wedge E^+(x_2, x_1) \wedge E^+(x_2, x_3) \wedge E^+(x_3, x_2) \wedge \\ & E^+(x_2, x_4) \wedge E^+(x_4, x_2) \wedge E^+(x_3, x_4) \wedge E^+(x_4, x_3) \wedge \\ & E^-(x_1, x_3) \wedge E^-(x_3, x_1) \wedge E^-(x_1, x_4) \wedge E^-(x_4, x_1), \\ & (), \\ & \{x_1 \mapsto \mathrm{T}, x_2 \mapsto \mathrm{T}, x_3 \mapsto \mathrm{T}, x_4 \mapsto \mathrm{T}\} \rangle \end{aligned}$$

$$\begin{aligned} A_2 = \langle & E^+(y_1, y_2) \wedge E^+(y_2, y_1) \wedge E^+(y_2, y_3) \wedge E^+(y_3, y_2) \wedge \\ & E^+(y_2, y_4) \wedge E^+(y_4, y_2) \wedge E^+(y_4, y_5) \wedge E^+(y_5, y_4) \wedge \\ & E^+(y_1, y_4) \wedge E^+(y_4, y_1) \wedge E^+(y_3, y_5) \wedge E^+(y_5, y_3) \wedge \\ & E^-(y_1, y_3) \wedge E^-(y_3, y_1) \wedge E^-(y_1, y_5) \wedge E^-(y_5, y_1) \wedge \\ & E^-(y_2, y_5) \wedge E^-(y_5, y_2) \wedge E^-(y_3, y_4) \wedge E^-(y_4, y_3), \\ & (), \\ & \{y_1 \mapsto \mathrm{T}, y_2 \mapsto \mathrm{T}, y_3 \mapsto \mathrm{T}, y_4 \mapsto \mathrm{T}, y_5 \mapsto \mathrm{T}\} \rangle \end{aligned}$$

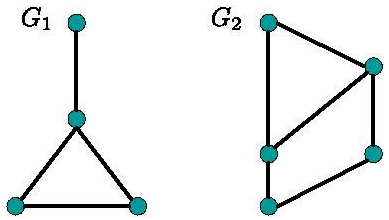

Figure 1: G_1, G_2

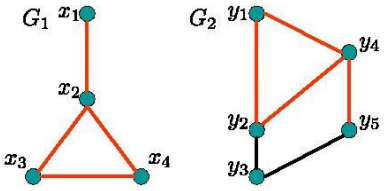

Figure 2: G_1 can be embedded isomorphic into G_2

It is easy to see that $A_2[\frac{x_3}{y_1}\frac{x_4}{y_2}\frac{x_2}{y_4}\frac{x_1}{y_5}] \subseteq_{as} A_1$ holds and therefore $A_2 \subseteq_{as} A_1$ holds as well due to theorem 1.

Theorem 4. PS_{STRIPS} *is* $\mathcal{NP}$*-complete.*

The proof is analogous to the one of theorem 3.

□

Complexity of AS_{BOOLE} and PS_{BOOLE}

Theorem 5. AS_{BOOLE} *is* Σ_2^p*-complete.*

Proof. Let t be the number of rules in **T**, p the number of axioms in **P**, a the highest number of parameters and u the number of different predicates in A and B (whereby a predicate $P(\vec{x})$ is different from another one $P'(\vec{x'})$, if $P \neq P'$ or $x_i \neq x_i'$ for any i).

Membership follows from the following algorithm:

Guess a substitution A^s of A and examine $A^s \subseteq_{as} B$:

S1: Transform $pre(A)$ and $pre(B)$ into two propositional formulas ϕ_A and ϕ_B by introducing a new proposition for each combination of predicate name and parameters (e.g. replace $P(x_1)$ with P_x_1). Next, instantiate **P** for each different predicate (e.g. for an formula $\forall\vec{x}(P(\vec{x}) \Rightarrow Q(\vec{x}))$ and the two predicates $P(x_1, c_3)$ and $P(c_2, x_4)$ introduce the two propositional formulas $(\neg P\_x_1\_c_3 \vee Q\_x_1\_c_3)$ and $(\neg P\_c_2\_x_4 \vee Q\_c_2\_x_4)$). This produces $O(u * p)$ axioms which is polynomial in the size of **P** and the number of different predicates. Next, form a conjunction $\phi_{\mathbf{P}}$ of all these axioms.
Then, transform ϕ_A, ϕ_B and $\phi_{\mathbf{P}}$ into three propositional formulas ϕ_A', ϕ_B' and $\phi_{\mathbf{P}}'$ in conjunctive normal form (which is known to be possible in polynomial time) and let a SAT oracle verify that $\phi_A' \wedge \neg\phi_B \wedge \phi_{\mathbf{P}}'$ is unsatisfiable.

S2: Normalize $eff(A)$ and $eff(B)$ (consuming polynomial time): Check for each $(c_i, e_i) \in eff(A)$ and $(c_j, e_j) \in$

$eff(A)$ whether $e_j \Rightarrow e_i$ holds in **P**. If this is the case, set $c_i := c_i \vee c_j$ (analogously for $eff(B)$).
Then proceed similar to S1.

S3: Obviously possible in polynomial time.

Hence, $\mathsf{AS}_{BOOLE} \in \Sigma_2^p$ holds.
Hardness is proved by reducing 2-QBF to AS_{BOOLE}. For that purpose we map a 2-QBF-formula

$$\Psi = \exists x_1, \ldots, x_n \forall y_1, \ldots, y_m \Phi(\vec{x}, \vec{y})$$

to two operators $A, B \in \mathcal{AS}_{BOOLE}$, such that $A \subseteq_{as} B$ holds iff Ψ is valid:

1. Create a parameter x_i in B and two parameters v_i^+ and v_i^- in A with $types_B(x_i) = types_A(v_i^+) = types_A(v_i^-) = T_i$ for each existentially quantified variable.
2. Create a parameter y_i in B with $types_B(y_i) = T_{-1}$ for each universally quantified variable.
3. Generate a formula $(P(v_i^+) \Leftrightarrow \top) \wedge (P(v_i^-) \Leftrightarrow \bot)$ for each existentially quantified variable x_i and add it as a disjunct to $pre(A)$.
4. Each existentially quantified variable x_i (y_i) and each constant c_i in $\Phi(\vec{x}, \vec{y})$ is replaced by a predicate $P(x_i)$ ($P(y_i)$) and $P(c_i)$, respectively. When all possible replacements are done, add $\Phi(\vec{x}, \vec{y})$ as a disjunction to $pre(B)$.
5. There are no effects in A and B.

To simplify matters, we will not draw any distinction between a variable x in Ψ and a predicate $P(x)$ in the generated operators in the following.

The mapping generates two operators A and B of the following forms:

$$\begin{gathered} A = \langle (P(v_1^+) \Leftrightarrow \top) \wedge (P(v_1^-) \Leftrightarrow \bot) \\ \wedge \ldots \wedge \\ (P(v_n^+) \Leftrightarrow \top) \wedge (P(v_n^-) \Leftrightarrow \bot), \\ (), \\ \{v_1^+ \mapsto T_1, v_1^- \mapsto T_1, \ldots, v_n^+ \mapsto T_n, v_n^- \mapsto T_n\} \rangle \end{gathered}$$

and

$$\begin{gathered} B = \langle \Phi(\vec{x}, \vec{y}), \\ (), \\ \{x_1 \mapsto T_1, \ldots, x_n \mapsto T_n, y_1 \mapsto T_{-1}, \ldots, y_m \mapsto T_{-1}\} \rangle \end{gathered}$$

Next, we show that Ψ is valid iff $A \subseteq_{as} B$ holds:
"$\Rightarrow$"
Let Ψ be valid. We show, that $A \subseteq_{as} B$ holds for the generated operators A and B.

Since there is no type hierarchy and due to the choice of the types in our mapping, the only way to create a substitution is to replace either v_i^+ or v_i^- by x_i.

Because $\Psi = \exists x_1, \ldots, x_n \forall y_1, \ldots, y_m \Phi(\vec{x}, \vec{y})$ is valid, there is at least one assignment α for $\{x_1 \ldots x_n\}$, such that $\forall y_1, \ldots, y_m \Phi(\alpha(\vec{x}), \vec{y})$ is valid. We consider the substitution A^s of A, in which v_i^+ was replaced by x_i if $\alpha(x_i) = 1$, and v_i^- by x_i if $\alpha(x_i) = 0$.(*)

Without loss of generality let $\alpha(x_i) = 1$ for all $1 \leq i \leq n$ in the following.

According to theorem 1 we have to show S1-S3 in order to show that $A \subseteq_{as} B$ holds. Due to the absence of effects in A and B S2 is satisfied trivially. As a result of the choice of the considered substitution S3 is satisfied as well. Thus, in order to show that $A \subseteq_{as} B$ holds, we just have to show that S1 holds. This is done by showing that the following formula is valid:

$$\begin{gathered} (P(x_1) \Leftrightarrow \top) \wedge (P(v_1^-) \Leftrightarrow \bot) \\ \wedge \ldots \wedge \\ (P(x_n) \Leftrightarrow \top) \wedge (P(v_n^-) \Leftrightarrow \bot) \\ \Rightarrow \\ \Phi(\vec{x}, \vec{y}) \end{gathered}$$

If the left side of this implication is not satisfied, the whole implication is satisfied. So we assume the left side to be satisfied in the following. Then, $\Phi(\vec{x}, \vec{y})$ has to be true as well, since all interpretations making the left side true have to set the existentially quantified variables $\{x_1, \ldots, x_n\}$ to the values $\alpha(x_i)$ for which $\forall y_1, \ldots, y_m \Phi(\alpha(\vec{x}), \vec{y})$ is true (see (*)). Thus, $A \subseteq_{as} B$ holds.
"$\Leftarrow$"

Let $A \subseteq_{as} B$ hold for the generated operators A and B. Then, according to theorem 1, $pre(A^s) \Rightarrow pre(B)$ is valid for a substitution A^s of A. There is no possible substitution for which the left side of the implication cannot be satisfied (since we can only substitute either v_i^+ or v_i^+ by x_i but not both at the same time). Therefore, Ψ has to be valid (if there are existentially quantified variables, we can restrict the interpretations satisfying the left side to those with the appropriate logical value).

Obviously, the mapping is polynomial in the size of Ψ and is therefore indeed a polynomial many-one reduction.

Example 2. For an example of the mapping used in the proof of theorem 5 consider the 2-QBF-formula

$$\Psi = \exists x_1 \forall y_1 (\neg y_1 \vee x_1).$$

The translation to propositional logic yields $\Psi' = ((\neg 1 \vee 1) \vee (\neg 1 \vee 0)) \wedge ((\neg 0 \vee 1) \vee (\neg 0 \vee 0)) \equiv (1 \vee 0) \wedge (1 \vee 1) \equiv 1 \wedge 1 \equiv 1$, which means that Ψ is valid.

The mapping generates two operators out of Ψ:

$$\begin{gathered} A = \langle (P(v_1^+) \Leftrightarrow \top) \wedge \\ (P(v_1^-) \Leftrightarrow \bot), \\ (), \\ \{v_1^+ \mapsto T_1, v_1^- \mapsto T_1\} \rangle \end{gathered}$$

and

$$\begin{gathered} B = \langle \neg P(y_1) \vee P(x_1), \\ (), \\ \{x_1 \mapsto T_1, y_1 \mapsto T_{-1}\} \rangle \end{gathered}$$

To decide whether $A \subseteq_{as} B$ holds, we have to test whether there is a substitution A^s of A such that $pre(A^s) \Rightarrow pre(B)$ holds - which means that $pre(A^s) \wedge \neg pre(B)$ has to be unsatisfiable.

There are two possible substitutions of A: $A[\frac{x_1}{v_1^-}]$ and $A[\frac{x_1}{v_1^+}]$.

Let us consider $A[\frac{x_1}{v_1^-}]$ first:

The test for unsatisfiability of $(P(v_1^+) \Leftrightarrow \top) \wedge (P(x_1) \Leftrightarrow$

$\bot) \wedge \neg(\neg P(y_1) \vee P(x_1))$ fails since we find a assignment: $\{P(v_1^+) \mapsto 1, P(x_1) \mapsto 0, P(y_1) \mapsto 1\}$.

So we consider $A[\frac{x_1}{v_1^+}]$ now:

For each assignment for $(P(x_1) \Leftrightarrow \top) \wedge (P(v_1^-) \Leftrightarrow \bot) \wedge \neg(\neg P(y_1) \vee P(x_1))$, $P(x_1) \mapsto 1$ and $P(v_1^-) \mapsto 0$ must hold. Therefore, $\neg(\neg P(y_1) \vee P(x_1))$ is always 0 and $(P(x_1) \Leftrightarrow \top) \wedge (P(v_1^-) \Leftrightarrow \bot) \wedge \neg(\neg P(y_1) \vee P(x_1))$ is indeed unsatisfiable.

Hence, $pre(A[\frac{x_1}{v_1^+}]) \Rightarrow pre(B)$ holds. Obviously, also S2 and S3 hold, and so $A[\frac{v_1^+}{x_1}] \subseteq_{as} A$ and due to theorem 1 $A \subseteq_{as} B$ hold as well.

Theorem 6. *PS_{BOOLE} is Σ_2^p-complete.*

The proof is analogous to the one of theorem 5. □

Complexity of AS_{ADL} and PS_{ADL}

In $\mathcal{A}_{ADL}$ we allow existentially and universally quantification in the precondition of the operator as well as in the precondition of the effects (see Definition 6) in addition to $\mathcal{A}_{BOOLE}$. Hence, in $\mathcal{A}_{ADL}$ we are endued with the full power of arbitrary FOL formulas. Although the number of objects in an object domain is always finite, AS_{ADL} and PS_{ADL} are undecidable.

Theorem 7. *AS_{ADL} and PS_{ADL} are undecidable.*

Proof. This follows directly from the theorem of Trakhtenbrot (1950), which states that the satisfiability problem of arbitrary FOL formulas is undecidability even if all domains are finite (therefore, the validity problem of FOL formulas and in particular the validity of $pre(A) \Rightarrow pre(B)$ and thus AS_{ADL} and PS_{ADL} are undecidable). □

Implementation

We implemented a prototype subsumption checker module for AS_{STRIPS} and AS_{BOOLE}. Since AS_{STRIPS} is $\mathcal{NP}$- and AS_{BOOLE} even Σ_2^p-complete, one would expect it to be very difficult to come up with reasoning procedures deciding these problems in reasonable time. Fortunately, many interesting problems are good-natured in several aspects.

A closer look at the proof of theorem 3 shows that the $\mathcal{NP}$-hardness of AS_{STRIPS} is due to the subproblem of finding an admissible substitution. In general, the number of admissible substitutions is exponential: If n is the number of parameters in A and m the number of parameters in B, there are $\frac{m!}{(m-n)!}$ many admissible substitutions. However, in many problems this number is more restricted due to other properties of the involved operators: At first, we can find an admissible substitution only if A has at least as many parameters as B (since each atom occurring in $pre(A^s)$ has to occur in $pre(B)$ as well). Another restriction is given by typed parameters: A parameter $x \in \mathcal{S}(A)$ can only be substituted by a parameter $y \in \mathcal{S}(B)$, if the type of x is a subtype of the type of y according to $\mathbf{T}$. Furthermore, a parameter y occurring in a predicate P in $pre(B)$ at position i has to occur at the same position of a predicate P' in $pre(A)$ such that $P \wedge \mathbf{P} \Rightarrow P'$ holds. Similar restrictions can be found for AS_{BOOLE}.

In future work, we intend to further investigate these restrictions in order to identify possibly tractable subsets of the problems.

Conclusion

We proposed two subsumption relations between operators, both of which are suitable for modeling and verifying abstraction hierarchies between actions and operators. Furthermore, we investigated the complexity of the operator subsumption problems for three special cases and proved that the problem is $\mathcal{NP}$-complete when the expressiveness of the operators is restricted to the basic STRIPS formalism, Σ_2^p-complete when we admit boolean logical operators and undecidable when the full power of the planning language ADL is permitted. For operators without quantifiers, subsumption relations were checked in reasonable time, however, using a prototype subsumption module, which we implemented.

Our results can be extended in several directions. In future work we will study the influence of universally quantified effects on the computational complexity of ADL formulae and investigate empirically whether there are efficient approximation algorithms for the subsumption problem of two ADL operators. We will also examine whether the concept of predicate inclusion axioms can be extended without making the decision problems undecidable. Another aim is to detect tractable subclasses for AS_{BOOLE} or even AS_{ADL}.

References

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Botea, A.; Enzenberger, M.; Müller, M.; and Schaeffer, J. 2005. Macro-FF: Improving AI planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research* 24:581–621.

Devanbu, P. T., and Litman, D. J. 1991. Plan-based terminological reasoning. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, 128–138.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1123–1128.

Eyerich, P.; Nebel, B.; Lakemeyer, G.; and Claßen, J. 2006. Golog and PDDL: What is the relative expressiveness? In *Proceedings of the International Symposium on Practical Cognitive Agents and Robots*, 73–80.

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

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Gil, Y. 2005. Description logics and planning. *AI Magazine* 26(2):73–84.

Heinsohn, J.; Kudenko, D.; Nebel, B.; and Profitlich, H.-J. 1992. RAT — representation of actions using terminological logics. In *DFKI Workshop on Taxonomic Reasoning — Proceedings*, number D-92-08.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Kemke, C., and Walker, E. 2006. Planning with action abstraction and plan decomposition hierarchies. In *Proceedings of the 2006 International Conference on Intelligent Agent Technology*, 447–451.

Kemke, C. 2003. A formal theory for describing action concepts in terminological knowledge bases. In *Proceedings of the Sixteenth Canadian Conference on Artificial Intelligence*, volume 2671, 458–465.

Knoblock, C. A. 1994. Automatically generating abstractions for planning. *Artificial Intelligence* 68(2):243–302.

Liebig, T., and Rösner, D. 1997. Action hierarchies in description logics. In *Proceedings of the 1997 International Workshop on Description Logics*, volume 410 of *URA-CNRS*.

Nau, D. S.; Muoz-Avila, H.; Cao, Y.; Lotem, A.; and Mitchell, S. 2001. Total-order planning with partially ordered subtasks. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 425–430.

Pednault, E. 1989. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 324–332.

Reiter, R. 2001. *Knowledge in Action*. MIT Press.

Sacerdoti, E. D. 1974. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence* 5(2):115–135.

Trakhtenbrot, B. A. 1950. Impossibility of an algorithm for the decision problem in finite classes. *Doklady Akademii Nauk SSSR* 70:569–572.

Weida, R. A., and Litman, D. J. 1992. Terminological reasoning with constraint networks and an application to plan recognition. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, 282–293.