Progression of Decomposed Situation Calculus Theories*

Denis Ponomaryov
Institute of Artificial Intelligence
University of Ulm, Germany;
A.P. Ershov Institute of Informatics Systems,
Novosibirsk, Russia
http://persons.iis.nsk.su/en/person/ponom

Mikhail Soutchanski
Department of Computer Science,
Ryerson University, Toronto, Canada
http://www.scs.ryerson.ca/mes

Abstract

In many tasks related to reasoning about consequences of a logical theory, it is desirable to decompose the theory into a number of components with weakly-related or independent signatures. This facilitates reasoning when the signature of a query formula belongs to only one of the components. However, an initial theory may be subject to change due to execution of actions affecting features mentioned in the theory. Having once computed a decomposition of a theory, one would like to know whether a decomposition has to be computed again for the theory obtained from taking into account the changes resulting from execution of an action. In the paper, we address this problem in the scope of the situation calculus, where change of an initial theory is related to the well-studied notion of progression. Progression provides a form of forward reasoning; it relies on forgetting values of those features which are subject to change and computing new values for them. We prove new results about properties of decomposition components under forgetting and show when a decomposition can be preserved in progression of an initial theory.

1 Introduction and Motivation

Modularity of theories has been established as an important research topic in knowledge representation. It includes both theoretical and practical aspects of modularity of theories formulated in different logical languages $\mathcal{L}$ ranging from weak (but practical) description logic (DL) $\mathcal{EL}$ to more expressive logics (Konev et al. 2009; 2010; Ponomaryov 2008; Grüninger et al. 2012), to cite a few. Surprisingly, this research topic is little explored in the context of reasoning about actions with a few exceptions, e.g., (Gu and Soutchanski 2008; Inclezan and Gelfond 2011; Kakas, Michael, and Miller 2011). More specifically, it is natural to decompose a large heterogeneous knowledge base (KB) covering several loosely coupled application domains into components that have little or no intersection in terms of signatures. Potentially, such decomposition can facilitate solving the projection problem that requires answering whether a given logical formula is true after executing a sequence of actions (events). In cases, when a query is a logical formula composed from symbols occuring only in one of the components, the query can be answered more easily than in the case when the whole KB is required. In turn, this can help in solving other reasoning problems such as planning or high-level program execution that require solution to the projection problem as a prerequisite. To the best of our knowledge, the only previous work that explored decomposition of logical theories for the purposes of solving the projection problem are the papers (Amir 2000; 2002). These papers investigate decomposition in the situation calculus (McCarthy and Hayes 1969; Reiter 2001), a well-known logical formalism for representation of actions and their effects. The author proposed reasoning procedures for a situation calculus theory by dividing syntactically the whole theory into weakly related partitions. Specifically, he developed algorithms that use local computation inside syntactically identified partitions and message passing between partitions. We take a different approach in our paper. Instead of decomposing the whole action theory into subsets, as in (Amir 2000; 2002), we consider signature decompositions of an initial KB only. Our components are not necessarily syntactic subsets of the KB. We concentrate on foundations, and explore properties of components produced by our decomposition. Whenever possible, we try to formulate these properties in a general logical language $\mathcal{L}$ that is a fragment of second order logic, but when necessary, we talk about a specific logic. As a reasoning technique for solving the projection problem, we concentrate on *progression* (Lin and Reiter 1997; Liu and Lakemeyer 2009), and on a related operation of *forgetting* (Lin and Reiter 1994). When an executed action has effects on some of the features, their old values should be forgotten, and then the new values of the affected features must be computed to obtain progression of the KB to the next stage. Not surprisingly, both forgetting and progression have intricate interactions with properties of decomposed components. We investigate whether *decomposability* and *inseparability*, important meta-theoretic properties studied previously, are preserved under forgetting and progression. The main contributions of our paper are sufficient conditions when components in a decomposed initial KB preserve their properties after progression. This invariance is important since progression may continue indefinitely as long as new actions are being executed. Also, it is important that progression can be sometimes computed within one of the components. This can be achieved under reasonable assumptions of coherence between signatures of components of a decomposed initial KB and signatures of axioms characterizing effects of actions, when the latter axioms can be grouped into weakly related sets according to their syntax.

2 Background

2.1 Model-Theoretic Definitions

Let $\mathcal{L}$ be a logic (possibly many-sorted) which is a fragment of second-order logic (either by syntax or by translation of formulas) and has the standard model-theoretic Tarskian semantics. We call *signature* a subset of non-logical symbols of $\mathcal{L}$. For a set of formulas $\mathcal{T}$ in $\mathcal{L}$, we denote by $\mathtt{sig}(\mathcal{T})$ the signature of $\mathcal{T}$, i.e. the set of all non-logical symbols

*An extended version with proofs can be downloaded from http://persons.iis.nsk.su/en/person/ponom/papers

which occur in $\mathcal{T}$. We will use the same notation $\mathtt{sig}(\varphi)$ for the signature of a formula φ in $\mathcal{L}$. If t is a term in the language of second–order logic then the same notation $\mathtt{sig}(t)$ will be used for the set of all non-logical symbols occurring in t. Throughout this paper, we use the notion of *theory* as a synonym for a set of formulas in $\mathcal{L}$ which are sentences when translated into second-order logic. Whenever we mention a set of formulas, it is assumed that this set is in $\mathcal{L}$, if the context is not specified. For two theories $\mathcal{T}_1$ and $\mathcal{T}_2$, the notation $\mathcal{T}_1 \equiv \mathcal{T}_2$ will be the abbreviation for the semantic equivalence. If $\mathcal{T}$ is a set of formulas in $\mathcal{L}$ and Δ is a signature then $\mathtt{Cons}(\mathcal{T}, \Delta)$ will denote the set of semantic consequences of $\mathcal{T}$ in signature Δ (in $\mathcal{L}$), i.e. $\mathtt{Cons}(\mathcal{T}, \Delta) = \{\varphi \in \mathcal{L} \mid \mathcal{T} \models \varphi \text{ and } \mathtt{sig}(\varphi) \subseteq \Delta\}$. We emphasize that this is a notation for a set of formulas in $\mathcal{L}$, because $\mathcal{T}$ may semantically entail formulas which are in second-order logic, but are outside of $\mathcal{L}$.

Next, we formulate two component properties of theories considered in this paper. The notion of inseparability has been previously introduced in the context of DLs, e.g., see (Konev et al. 2009; Lutz and Wolter 2010).

Definition 1 (Δ–inseparability) *Theories $\mathcal{T}_1$ and $\mathcal{T}_2$ in $\mathcal{L}$ are called Δ–inseparable for a signature Δ, if $\mathtt{Cons}(\mathcal{T}_1, \Delta) = \mathtt{Cons}(\mathcal{T}_2, \Delta)$. That is, no formula in signature Δ "witnesses" any distinction between $\mathcal{T}_1$ and $\mathcal{T}_2$.*

In other words, $\mathcal{T}_1$ and $\mathcal{T}_2$ are Δ–inseparable, if for any formula ψ in signature Δ, $\mathcal{T}_1$ entails ψ iff $\mathcal{T}_2$ does. The following notion is introduced in (Ponomaryov 2008), and is applied to the study of modularity in (Konev et al. 2010).

Definition 2 (Δ–decomposability property) *Let $\mathcal{T}$ be a theory in $\mathcal{L}$ and $\Delta \subseteq \mathtt{sig}(\mathcal{T})$ be a subsignature. We call $\mathcal{T}$ Δ–decomposable, if there are theories $\mathcal{T}_1$ and $\mathcal{T}_2$ in $\mathcal{L}$ such that*

- $\mathtt{sig}(\mathcal{T}_1) \cap \mathtt{sig}(\mathcal{T}_2) = \Delta$ *and* $\mathtt{sig}(\mathcal{T}_1) \neq \Delta \neq \mathtt{sig}(\mathcal{T}_2)$*;*
- $\mathtt{sig}(\mathcal{T}_1) \cup \mathtt{sig}(\mathcal{T}_2) = \mathtt{sig}(\mathcal{T})$ *and* $\mathcal{T} \equiv \mathcal{T}_1 \cup \mathcal{T}_2$.

The pair $\langle \mathcal{T}_1, \mathcal{T}_2 \rangle$ is called Δ–decomposition of $\mathcal{T}$ and the theories $\mathcal{T}_1$ and $\mathcal{T}_2$ are called Δ–decomposition components of $\mathcal{T}$. We will sometimes omit the word "decomposition" and call the sets $\mathcal{T}_1$ and $\mathcal{T}_2$ simply components of $\mathcal{T}$, when the signature Δ is clear from the context.

The notion of signature Δ–decomposition is defined using a pair of theories, but is easily extended to the case of a family of theories. It is important to realize that $\mathcal{T}_1$ and $\mathcal{T}_2$ need not be syntactic subsets of $\mathcal{T}$ in the above definition. Clearly, if $\mathcal{L}$ satisfies compactness and $\mathcal{T}$ is a finite Δ–decomposable theory in $\mathcal{L}$ for a signature Δ, then there is a Δ–decomposition $\langle \mathcal{T}_1, \mathcal{T}_2 \rangle$ of $\mathcal{T}$, where $\mathcal{T}_1$ and $\mathcal{T}_2$ are finite. Although, the union $\mathcal{T}_1 \cup \mathcal{T}_2$ must entail all consequences of $\mathcal{T}$ in signature Δ, the components $\mathcal{T}_1$ and $\mathcal{T}_2$ may not be Δ–inseparable, if we demand them to be finite. For example, the set of Δ–consequences of $\mathcal{T}_2$ may not be finitely axiomatizable in $\mathcal{L}$ by axioms of $\mathcal{T}_1$.

Both Δ–decomposition and Δ–inseparability are required to achieve modularity. Without Δ–inseparability components are not self-sufficient, since a component may not entail some of the consequences in the shared vocabulary Δ. The ideal case is when a theory $\mathcal{T}$ has Δ-decomposition into finite Δ–inseparable components, as noted in the following.

The well-known property of logics related to signature decompositions of theories is the Parallel Interpolation Property (PIP) first considered in a special form in (Kourousias and Makinson 2007) and studied later in a more general form in (Konev et al. 2010). Note that PIP is closely related to Craig's interpolation (Craig 1957; 2008).

Definition 3 (Parallel Interpolation Property) *The logic $\mathcal{L}$ is said to have the parallel interpolation property (PIP) if for any theories $\mathcal{T}_1, \mathcal{T}_2$ in $\mathcal{L}$ with $\mathtt{sig}(\mathcal{T}_1) \cap \mathtt{sig}(\mathcal{T}_2) = \Delta$ and any formula φ in $\mathcal{L}$, the condition $\mathcal{T}_1 \cup \mathcal{T}_2 \models \varphi$ yields the existence of sets of formulas $\mathcal{T}_1'$ and $\mathcal{T}_2'$ in $\mathcal{L}$ such that:*

- $\mathcal{T}_i \models \mathcal{T}_i'$*, for* $i = 1, 2$*, and* $\mathcal{T}_1' \cup \mathcal{T}_2' \models \varphi$*;*
- $\mathtt{sig}(\mathcal{T}_i') \setminus \Delta \subseteq (\mathtt{sig}(\mathcal{T}_i) \cap \mathtt{sig}(\varphi)) \setminus \Delta$.

In fact, PIP can be understood as an iterated version of Craig's interpolation in the logics that have compactness and deduction theorem (see Lemma 1 in (Ponomaryov 2008)). Many logics known to have Craig's interpolation, e.g. second and first-order logics, numerous modal logics and some description logics also have PIP. Also, it is known that there are logics which do not have Craig's interpolation, and therefore do not have PIP. It is easy to show that in presence of PIP, decomposing a set $\mathcal{T}$ of formulas into inseparable components wrt a signature Δ gives a family of theories that imply all the consequences of $\mathcal{T}$ in their own subsignatures:

Fact *Let $\mathcal{L}$ have PIP, $\mathcal{T}$ be a theory in $\mathcal{L}$, and Δ be a signature. Let $\langle \mathcal{T}_1, \mathcal{T}_2 \rangle$ be a Δ–decomposition of $\mathcal{T}$ with $\mathcal{T}_1$ and $\mathcal{T}_2$ being Δ–inseparable. Then for any formula φ with $\mathtt{sig}(\varphi) \subseteq \mathtt{sig}(\mathcal{T}_i)$, $i \in \{1,2\}$, we have $\mathcal{T} \models \varphi$ iff $\mathcal{T}_i \models \varphi$.*

In other words, in presence of PIP, inseparable decomposition components can be used instead of the original theory for checking entailment of formulas in the corresponding subsignatures. This is the reason for our interest in the inseparability property in connection with decompositions. As shown in (Amir 2000; 2002), having a decomposed theory can be beneficial even without inseparability by applying the known methods of distributed reasoning via message passing between components. However, having inseparability of components allows the reasoner to avoid message passing completely.

2.2 Basics of the Situation Calculus

The situation calculus (SC) is a many-sorted predicate logic language for axiomatizing dynamic worlds. Its basic ingredients consist of *objects*, *actions* a, *situations* s and *fluents* (Reiter 2001). *Objects* are constants naming some entities, while *actions*, $A(\bar{x})$, are functions with possibly empty tuples $\bar{x}$ of object arguments. A *situation* is a first-order term denoting an unique sequence of actions. Such sequences are represented using a binary function symbol do: $do(a, s)$ denotes the sequence resulting from adding an action term a to the sequence s. The special constant S_0 denotes the *initial situation*, namely the empty action sequence. The relational *fluents* are predicates that have a single situation term as the last argument. Those predicate and function symbols which do not have situation arguments are called *static*. SC includes the distinguished predicate $Poss(a, s)$ to characterize actions a that are possible to execute in s, and the distinguished predicate $s_1 \preceq s_2$ to specify precedence between situations s_1 and s_2. As usual, we say that a SC formula $\psi(s)$

is *uniform* in s, if s is the only situation term mentioned in $\psi(s)$, the formula ψ has no occurrences of the predicates $Poss, \preceq$, and mentions neither quantifiers over variables of sort situation, nor equalities between situation terms.

The language of SC is used to formulate *basic action theories* ($\mathcal{BAT}$s) (Reiter 2001). Every $\mathcal{BAT}$ consists of a set of foundational axioms Σ axiomatizing properties of the relation $s_1 \preceq s_2$ w.r.t. $do(a, s)$, e.g., $\forall a, s\ (s \preceq do(a, s))$, a theory D_{una} stating the unique name assumption (UNA) for action functions and objects, an initial theory D_{S_0} describing knowledge in the initial situation S_0 as a set of sentences uniform in S_0, a theory D_{ap} specifying preconditions of action execution, and a theory D_{ss} (the set of successor-state axioms, SSAs for short). In axioms of a $\mathcal{BAT}$, all free variables (starting with lower case letters) are implicitly universally quantified at front. In theory D_{ss}, there is a single SSA for each relational fluent $F(\bar{x}, s)$, with a syntactic form $F(\bar{x}, do(a, s)) \leftrightarrow \Phi_F(\bar{x}, a, s)$, where $\Phi_F(\bar{x}, a, s)$ is a formula uniform in s with free variables among $\bar{x}, a, s$. We consider SSAs of the following form:

$F(\bar{x}, do(a, s)) \leftrightarrow \bigvee_i [\exists \bar{y}] a = \mathit{PosAction}_i(\bar{x}, \bar{y}) \wedge \gamma_i^+(\bar{x}, \bar{y}, s) \vee$
$F(\bar{x}, s) \wedge \neg\big(\bigvee_j [\exists \bar{z}] a = \mathit{NegAction}_j(\bar{x}, \bar{z}) \wedge \gamma_j^-(\bar{x}, \bar{z}, s)\big),$

where $\mathit{PosAction}_i$ is an action that makes the fluent F true and $\gamma_i^+(\bar{x}, \bar{y}, s)$ is the formula expressing a context in which this positive effect can occur; similarly, $\mathit{NegAction}_j$ is an action that can make the fluent F false if the formula $\gamma_j^-(\bar{x}, \bar{z}, s)$ holds in s. The optional $[\exists]$ means quantifiers over variables $\bar{y}, \bar{z}$, if any, other than those in $\bar{x}$ which may occur in action functions or in context formulas. If the executed action a is none of these, then the truth value of F remains unchanged (a has no effect). Thus, SSAs represent non-effects of actions compactly and characterize the truth values of the fluent F in the next situation $do(a, s)$ in terms of values of fluents in the situation s and static predicates and functions. Action function $A(\bar{x})$ is said to be in *active position* of some SSA $\varphi \in \mathcal{D}_{ss}$ for the fluent F, if $A(\bar{x})$ occurs either as an action having a positive effect on F, or as an action that has a negative effect on F. Rephrasing (Liu and Levesque 2005), where an action with local effects was originally defined, an SSA $\varphi \in \mathcal{D}_{ss}$ for the fluent F is called *local-effect* if the set of arguments of every action function in active position of φ contains all object variables from F. Notice that a generic SSA above is a local-effect SSA; in this paper, we do not consider actions with non local effects. A $\mathcal{BAT}$ is said to be local-effect if every axiom of $\mathcal{D}_{ss}$ is a local-effect SSA; subsequently, we consider only local-effect basic action theories.

Throughout the paper, we assume that $\mathcal{D}_{S_0}$ is a theory in language $\mathcal{L}$ which can be translated into a set of sentences of first-order logic uniform in S_0. In particular, $\mathcal{D}_{S_0}$ can include both an ABox and a TBox in an appropriate DL, as argued in (Gu and Soutchanski 2010; Soutchanski and Yehia 2012).

Proposition 1 (Theorem 1 in (Pirri and Reiter 1999)) *A basic action theory $\Sigma \cup D_{una} \cup D_{S_0} \cup D_{ap} \cup D_{ss}$ is satisfiable iff $D_{una} \cup D_{S_0}$ is satisfiable.*

One of the most important reasoning tasks in SC is the *projection problem*, that is, to determine whether a certain property holds after performing a sequence of actions. Planning and high-level program execution are two important settings, where this problem arises naturally. In this paper, we solve the projection problem using progression (Section 4), which provides forward style reasoning. In local-effect $\mathcal{BAT}$s, computing progression of initial theory $\mathcal{D}_{S_0}$ relies on updating $\mathcal{D}_{S_0}$ with new facts inferred from SSAs and forgetting old facts in $\mathcal{D}_{S_0}$ which are no longer true. Local-effect $\mathcal{BAT}$s are a well-known class of theories for which the operation of progression can be computed effectively (Liu and Lakemeyer 2009). They are special in the sense that the truth value of each fluent in a local-effect $\mathcal{BAT}$ can change only for a finite set of objects occurring as arguments of an executed action. Therefore, in this case forgetting old values of fluents can be done efficiently, contributing to the overall effectiveness of computing progression in local-effect $\mathcal{BAT}$s.

Before we proceed to component properties of forgetting (Section 3) and to progression of initial theories (Section 4), we consider an example that helps to illustrate the advantages of decomposition. Our example combines the simplified Blocks World (BW) with a kind of Stacks World. A complete axiomatization of BW modelled as a finite collection of finite chains can be found in (Cook and Liu 2003).

Example 1 (of $\mathcal{BAT}$) The blocks-and-stacks-world consists of a finite set of blocks and a finite set of other entities. Blocks can be located on top of each other, while other entities can be either in a heap of unlimited capacity, or can be organized in stacks. There is an unnamed manipulator that can move a block from one block to another, provided that there is nothing on the top of the blocks. It can also put an entity from the heap upon a stack with a named top element, or move the top element of a stack into the heap. For stacking/unstacking operations we adopt the push/pop terminology and use the unary predicate $Block$ to distinguish between blocks and other entities. We use the following action functions and relational fluents to axiomatize the mentioned world as a local-effect $\mathcal{BAT}$ in SC.

Actions
- $move(x, y, z)$: Move block x from block y onto block z, provided both x and z are clear.
- $push(x,y)$: Stack entity x from the heap on top of entity y.
- $pop(x)$: Unstack entity x into the heap, provided x is the top element and is not in the heap.

Fluents
- $On(x, z, s)$: Block x is on block z, in situation s.
- $Clear(x, s)$: Block x has no other blocks on top of it in s.
- $Top(x, s)$: Entity x is the top element of a stack in s.
- $Inheap(x, s)$: Entity x is in the heap in situation s.
- $Under(x, y, s)$: Entity y is directly under x in situation s.

Successor state axioms (theory $\mathcal{D}_{ss}$)

$On(x, z, do(a, s)) \leftrightarrow\ \exists y(a = move(x, y, z)) \vee$
$\quad On(x, z, s) \wedge\ \neg\exists y(a = move(x, z, y))$

$Clear(x, do(a, s)) \leftrightarrow\ \exists y, z(a = move(y, x, z) \wedge$
$\quad On(y, x, s)) \vee\ Clear(x, s) \wedge \neg\exists y, z(a = move(y, z, x))$

$Inheap(x, do(a, s)) \leftrightarrow\ a = pop(x) \vee$
$\quad Inheap(x, s) \wedge \neg\exists y(a = push(x, y))$

$Top(x, do(a, s)) \leftrightarrow\ \exists y(\, a = push(x, y)\,) \vee$
$\quad \exists y(\, a = pop(y) \wedge Under(y, x, s)\,) \vee$
$\quad Top(x, s) \wedge a \neq pop(x) \wedge \neg\exists y(a = push(y, x))$

$Under(x, y, do(a, s)) \leftrightarrow\ a = push(x, y) \vee$
$\quad Under(x, y, s) \wedge a \neq pop(x)$

Action precondition axioms (theory $\mathcal{D}_{ap}$)

$$Poss(move(x,y,z),s) \leftrightarrow Block(x) \wedge Block(y) \wedge Block(z) \wedge Clear(x,s) \wedge Clear(z,s) \wedge x \neq z$$

$$Poss(push(x,y),s) \leftrightarrow \neg Block(x) \wedge \neg Block(y) \wedge Top(y,s) \wedge Inheap(x,s)$$

$$Poss(pop(x),s) \leftrightarrow \neg Block(x) \wedge Top(x,s)$$

Initial Theory ($\mathcal{D}_{S_0}$) is defined as the set of axioms:

$\neg\exists y On(y,x,S_0) \wedge \exists y On(x,y,S_0) \wedge \neg Inheap(x,S_0) \rightarrow Clear(x,S_0)$
$\exists y\, On(x,y,S_0) \rightarrow Block(x)$
$(Top(x,S_0) \vee Inheap(x,S_0)) \rightarrow \neg Block(x)$
$On(A,B,S_0) \wedge Block(B) \wedge Block(C) \wedge Clear(A,S_0) \wedge Clear(C,S_0)$

Notice that all fluents are syntactically related in $\mathcal{D}_{S_0}$, so purely syntactic techniques fail to decompose $\mathcal{D}_{S_0}$ into components sharing no fluents. Finally, the theory $\mathcal{D}_{una}$ is the set of unique-name axioms for all pairs of object constants and action functions used above. Then $\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{S_0}$ is the resulting local-effect $\mathcal{BAT}$. Note that $\mathcal{D}_{ss}$ is the union of two theories with the intersection of signatures equal to $\{do\}$. At the same time, the initial theory $\mathcal{D}_{S_0}$ is Δ–decomposable for $\Delta = \{Block, S_0\}$ into two distinct Δ–inseparable components:

$\neg\exists y\, On(y,x,S_0) \wedge \exists y\, On(x,y,S_0) \rightarrow Clear(x,S_0)$
$\exists y\, On(x,y,S_0) \rightarrow Block(x)$
$On(A,B,S_0) \wedge Block(B) \wedge Block(C) \wedge Clear(A,S_0) \wedge Clear(C,S_0)$

and

$(Top(x,S_0) \vee Inheap(x,S_0)) \rightarrow \neg Block(x)$
$\exists x\, Block(x)$

We will show in Theorem 1 that progression for $\mathcal{BAT}$s of this kind preserves both decomposability and inseparability of the decomposition components.

3 Properties of Forgetting

As progression is closely related to forgetting, we take a look at some properties of this operation first. Let us define a relation on structures as follows. Let σ be a signature or a ground atom and $\mathcal{M}$, $\mathcal{M}'$ be two many–sorted structures. Then we set $\mathcal{M} \sim_\sigma \mathcal{M}'$ if:

- $\mathcal{M}$ and $\mathcal{M}'$ have the same domain for each sort;
- $\mathcal{M}$ and $\mathcal{M}'$ interpret all symbols which are not in σ identically;
- if σ is a ground atom $P(\bar{t})$ then $\mathcal{M}$ and $\mathcal{M}'$ agree on interpretation $\bar{u}$ of $\bar{t}$ and for every vector of elements $\bar{v} \neq \bar{u}$, we have $\mathcal{M} \models P(\bar{v})$ iff $\mathcal{M}' \models P(\bar{v})$.

Obviously, $\sim_\sigma$ is an equivalence relation on structures.

The following notion summarizes the well-known Definitions 1 and 7 in (Lin and Reiter 1994).

Definition 4 (Forgetting an atom or a signature) *Let $\mathcal{T}$ be a theory in $\mathcal{L}$ and σ be either a signature, or some ground atom. A set $\mathcal{T}'$ of formulas in a fragment of second-order logic is called the result of forgetting σ in $\mathcal{T}$ (denoted by $\texttt{forget}(\mathcal{T},\sigma)$) if for any structure $\mathcal{M}'$, we have $\mathcal{M}' \models \mathcal{T}'$ iff there is a model $\mathcal{M} \models \mathcal{T}$ such that $\mathcal{M} \sim_\sigma \mathcal{M}'$.*

It is known that $\texttt{forget}(\mathcal{T},\sigma)$ always exists, i.e. is second–order definable, for a finite set of formulas $\mathcal{T}$ in $\mathcal{L}$ and a finite signature or a ground atom σ (see (Lin and Reiter 1994), or Section 2.1 in (Liu and Lakemeyer 2009)). The definition yields $\mathcal{T} \models \texttt{forget}(\mathcal{T},\sigma)$, thus $\texttt{forget}(\mathcal{T},\sigma)$ is a set of second-order consequences of $\mathcal{T}$ which suggests that it may not always be definable in the logic where $\mathcal{T}$ is formulated and it may not be finitely axiomatizable in this logic, even if so is $\mathcal{T}$. For the case when σ is a signature, $\texttt{forget}(\mathcal{T},\sigma)$ is known as $\texttt{sig}(\mathcal{T}) \setminus \sigma$–uniform interpolant of $\mathcal{T}$ wrt the language $\mathcal{L}$ and second-order queries, see Definition 13 in (Konev et al. 2009) and Lemma 39 in (Lutz and Wolter 2010) for a justification. In other words, $\mathcal{T}$ and $\texttt{forget}(\mathcal{T},\sigma)$ semantically entail the same second-order formulas in signature $\mathcal{T} \setminus \sigma$.

Let $\mathcal{L}$ be first-order logic. In contrast to forgetting a signature, for any recursively axiomatizable theory $\mathcal{T}$ in $\mathcal{L}$ and a ground atom σ, one can effectively construct the set of formulas $\texttt{forget}(\mathcal{T},\sigma)$ in $\mathcal{L}$ such that $\texttt{forget}(\mathcal{T},\sigma)$ is finitely axiomatizable iff $\mathcal{T}$ is. This follows from Theorem 4 in (Lin and Reiter 1994), where it is shown that forgetting a ground atom $P(\bar{t})$ in a theory $\mathcal{T}$ can be computed by simple syntactic manipulations. Below we formulate a number of new properties of forgetting that we will subsequently need when proving new results about progression.

Proposition 2 (Interplay of forgetting and entailment) *Let $\mathcal{T}$ and $\mathcal{T}_1$ be two sets of formulas in $\mathcal{L}$ with $\mathcal{T} \models \mathcal{T}_1$ and σ be a signature or a ground atom. Then the following holds:*

$$\begin{array}{ccc} \mathcal{T} & \models & \mathcal{T}_1 \\ \top & & \top \\ \texttt{forget}(\mathcal{T},\sigma) & \models & \texttt{forget}(\mathcal{T}_1,\sigma) \end{array}$$

Proposition 3 (Preservation of consequences) *Let $\mathcal{T}$ be a theory in $\mathcal{L}$ and σ be either a signature or a ground atom. Let φ be a formula such that either $\texttt{sig}(\varphi) \cap \sigma = \varnothing$ (in case σ is a signature), or which does not contain the predicate from σ (if σ is a ground atom). Then $\mathcal{T} \models \varphi$ iff $\texttt{forget}(\mathcal{T},\sigma) \models \varphi$.*

Now we formulate some necessary results on preservation of inseparability under forgetting. By Proposition 3, when studying preservation of Δ–inseparability of two sets of formulas for a signature Δ, it is sufficient to consider the case of forgetting a subset of Δ or a ground atom with predicate from Δ, respectively. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two sets of formulas in $\mathcal{L}$ with $\texttt{sig}(\mathcal{T}_1) \cap \texttt{sig}(\mathcal{T}_2) = \Delta$ for a signature Δ and let σ be either a subsignature of Δ or a ground atom with predicate from Δ. It is known that in general, forgetting σ may not be distributive over the union of sets of formulas. The entailment $\texttt{forget}(\mathcal{T}_1 \cup \mathcal{T}_2,\sigma) \models \texttt{forget}(\mathcal{T}_1,\sigma) \cup \texttt{forget}(\mathcal{T}_2,\sigma)$ holds by Proposition 2, but Example 2 easily shows that even strong semantic conditions related to modularity do not guarantee the reverse entailment. On the other hand, forgetting something outside of the common signature of $\mathcal{T}_1$ and $\mathcal{T}_2$ is distributive over union, as formulated in Corollary 1 which is a consequence of criterion in Prop. 4. This is important since a KB is a union of components.

Example 2 (Failure of componentwise forgetting in Δ) *Let $\mathcal{L}$ be first-order logic and $\Delta = \{P, c\}$ be a signature consisting of unary predicate P and constant c. Define theories $\mathcal{T}_1$ and $\mathcal{T}_2$ as: $\mathcal{T}_1 = \{A \rightarrow P(c)\}$, $\mathcal{T}_2 = \{P(c) \rightarrow B\}$, where A, B are nullary predicate symbols. We have $\texttt{sig}(\mathcal{T}_1) \cap \texttt{sig}(\mathcal{T}_2) = \Delta$ and for $i = 1, 2$, any model of $\mathcal{T}_i$ can be expanded to a model of $\mathcal{T}_1 \cup \mathcal{T}_2$. Clearly, $\mathcal{T}_1$ and $\mathcal{T}_2$ are Δ–inseparable and for $i = 1, 2$, $\texttt{Cons}(\mathcal{T}_i, \Delta)$ is the set of tautologies in Δ. By definition of forgetting, for $i = 1, 2$, $\texttt{forget}(\mathcal{T}_i, P(c))$ is a set of tautologies and thus, $\texttt{forget}(\mathcal{T}_1, P(c)) \cup \texttt{forget}(\mathcal{T}_2, P(c)) \not\models \texttt{forget}(\mathcal{T}_1 \cup$*

$\mathcal{T}_2, P(c))$, *because* $\texttt{forget}\,(\mathcal{T}_1 \cup \mathcal{T}_2, P(c)) \models A \to B$ *(by Proposition 3). For the case of forgetting a signature, say nullary predicate* P*, it suffices to consider* $\Delta = \{P\}$ *and theories* $\mathcal{T}_1 = \{A \to P\}$, $\mathcal{T}_2 = \{P \to B\}$, *where* A, B *are nullary predicates.*

Proposition 4 (A criterion for componentwise forgetting) *Let* $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *be two sets of formulas and* σ *be either a signature or a ground atom. Then the following are equivalent:*

- $\texttt{forget}\,(\mathcal{T}_1, \sigma) \cup \texttt{forget}\,(\mathcal{T}_2, \sigma) \models \texttt{forget}\,(\mathcal{T}_1 \cup \mathcal{T}_2, \sigma)$
- *for any two models* $\mathcal{M}_1 \models \mathcal{T}_1$ *and* $\mathcal{M}_2 \models \mathcal{T}_2$*, with* $\mathcal{M}_1 \sim_\sigma \mathcal{M}_2$*, there exists a model* $\mathcal{M} \models \mathcal{T}_1 \cup \mathcal{T}_2$ *such that* $\mathcal{M} \sim_\sigma \mathcal{M}_i$ *for some* $i \in \{1, 2\}$.

To compare this criterion with Example 2, observe that there exist models $\mathcal{M}_1 \models \mathcal{T}_1$ and $\mathcal{M}_2 \models \mathcal{T}_2$ with a common domain such that $\mathcal{M}_1 \models A \wedge P(c) \wedge \neg B$ and $\mathcal{M}_2 \models A \wedge \neg P(c) \wedge \neg B$. Thus, $\mathcal{M}_1 \sim_{P(c)} \mathcal{M}_2$; however, there does not exist a model $\mathcal{M}$ of $\mathcal{T}_1 \cup \mathcal{T}_2$ such that $\mathcal{M} \sim_{P(c)} \mathcal{M}_i$ for some $i \in \{1, 2\}$. Neither $\mathcal{M}_1$, nor $\mathcal{M}_2$ is a model for $\mathcal{T}_1 \cup \mathcal{T}_2$.

Corollary 1 (Forgetting in scope of one component) *Let* $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *be sets of formulas with* $\texttt{sig}\,(\mathcal{T}_1) \cap \texttt{sig}\,(\mathcal{T}_2) = \Delta$ *for a signature* Δ *and* σ *be either a subsignature of* $\texttt{sig}\,(\mathcal{T}_1) \setminus \Delta$ *or a ground atom with predicate from* $\texttt{sig}\,(\mathcal{T}_1) \setminus \Delta$*. Then* $\texttt{forget}\,(\mathcal{T}_1 \cup \mathcal{T}_2, \sigma)$ *is equivalent to* $\texttt{forget}\,(\mathcal{T}_1, \sigma) \cup \mathcal{T}_2$*. Also, if* $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *are* Δ*–inseparable, then so are* $\texttt{forget}\,(\mathcal{T}_1, \sigma)$ *and* $\mathcal{T}_2$.

4 Component Properties under Progression

The operation of progression is closely related to forgetting in initial theories (Lin and Reiter 1997). We use the following notations further in this section. For a ground action term α in the language of the situation calculus, we denote by S_α the situation term $do(\alpha, S_0)$. To define progression, let us introduce an equivalence relation on many-sorted structures in situation calculus signature. For two structures $\mathcal{M}$, $\mathcal{M}'$ and a ground action α, we set $\mathcal{M} \sim_{S_\alpha} \mathcal{M}'$ if:

- $\mathcal{M}$ and $\mathcal{M}'$ have the same sorts for action and object;
- $\mathcal{M}$ and $\mathcal{M}'$ interpret all situation-independent predicate and function symbols identically;
- $\mathcal{M}$ and $\mathcal{M}'$ agree on interpretation of all fluents at S_α, i.e. for every fluent F and every variable assignment θ, we have $\mathcal{M}, \theta \models F(\bar{x}, S_\alpha)$ iff $\mathcal{M}', \theta \models F(\bar{x}, S_\alpha)$.

The following is a reformulation of the definitions from Sect. 4 in (Lin and Reiter 1997) and Def.9.1.1 in (Reiter 2001).

Definition 5 (Progression) *Let* $\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ *be a basic action theory with the initial theory* $\mathcal{D}_{S_0}$ *and let* α *be a ground action term. A set* $\mathcal{D}_{S_\alpha}$ *of formulas in a fragment of second-order logic is called progression of* $\mathcal{D}_{S_0}$ *wrt* α *if it is uniform in the situation term* S_α *and for any structure* $\mathcal{M}$*,* $\mathcal{M}$ *is a model of* $\Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_\alpha}$ *iff there is a model* $\mathcal{M}'$ *of* $\mathcal{D}$ *such that* $\mathcal{M} \sim_{S_\alpha} \mathcal{M}'$.

Below, we use $\mathcal{D}_{S_\alpha}$ to denote progression of the initial theory wrt the action term α, if the context of $\mathcal{BAT}$ is clear. We sometimes abuse terminology and call progression not only the theory $\mathcal{D}_{S_\alpha}$ itself, but also the operation of computing this theory (when the existence of an effective operation is implicitly assumed). It can be seen (Theorem 2 in (Lin and Reiter 1997) and Theorem 2.10 in (Liu and Lakemeyer 2009)) that progression always exists, i.e. is second-order definable, if the signature of $\mathcal{BAT}$ is finite and the initial theory $\mathcal{D}_{S_0}$ is finitely axiomatizable. On the other hand, by the definition, for any $\mathcal{BAT}$ $\mathcal{D}$, we have $\mathcal{D} \models \mathcal{D}_{S_\alpha}$ and, similarly to the operation of forgetting, it is possible to provide an example (see Definition 2, Conjecture 1, and Theorem 2 in (Vassos and Levesque 2008)), when progression $\mathcal{D}_{S_\alpha}$ is not definable (even by an infinite set of formulas) in the logic in which $\mathcal{D}_{S_0}$ is formulated.

The progression $\mathcal{D}_{S_\alpha}$ is a set of consequences of $\mathcal{BAT}$ which are uniform in the situation term S_α: it can be viewed as the *strongest postcondition* of the precondition $\mathcal{D}_{S_0}$ wrt the action α. Thus, informally, $\mathcal{D}_{S_\alpha}$ is a knowledge about the situation S_α implied by $\mathcal{BAT}$. Moreover, it contains all knowledge from $\mathcal{BAT}$ about the situation S_α, as guaranteed by the model-theoretic property with the relation $\sim_{S_\alpha}$ in the definition. Recall that the initial theory of $\mathcal{BAT}$ describes knowledge about the initial situation S_0 and SSAs are essentially the rules for computing new values of fluents after performing actions. Thus, progression $\mathcal{D}_{S_\alpha}$ can be viewed as "modification" of the initial theory obtained after executing the action α. Subsequently, since it is convenient to reuse S_0, we consider substitution of S_α with S_0 in progression, denoted $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$, where S_0/S_α is the result of replacing every occurrence of S_α with S_0. Let $\varphi(s)$ be a formula uniform in a situation variable s. To solve the projection problem for $\varphi(S_\alpha)$, i.e., to find whether $\varphi(S_\alpha)$ holds in the situation S_α wrt $\mathcal{BAT}$ $\mathcal{D}$, one might wish to compute progression $\mathcal{D}_{S_\alpha}$ and then check whether $\mathcal{D}_{una} \cup \mathcal{D}_{S_\alpha} \models \varphi(S_\alpha)$ holds (or equivalently, whether $\mathcal{D}_{una} \cup \mathcal{D}_{S_\alpha}(S_0/S_\alpha) \models \varphi(S_0)$ holds). By Proposition 1, this is equivalent to $\mathcal{D} \models \varphi(S_\alpha)$, i.e, this approach solves the projection problem for $\varphi(S_\alpha)$.

Consequently, of interest are the cases when progression can be computed effectively as a theory in the same logic used to formulate underlying $\mathcal{D}_{S_0}$, independently of the fact whether satisfiability in this logic is decidable. The well-known approach is to consider the local-effect $\mathcal{BAT}$s. The essence of computing progression for the local-effect BATs is to identify effectively from SSAs the set of ground atoms that need to be forgotten. Subsequently, in $\mathcal{D}_{S_\alpha}$, they are replaced with the new values of fluents; these new values are also computed from SSAs. An interested reader may consult the whole paper (Liu and Lakemeyer 2009), while here we introduce only those necessary constructions from Def. 3.4 of (Liu and Lakemeyer 2009) which are useful in order to understand our results.

Let $\mathcal{D}$ be a local-effect $\mathcal{BAT}$ with a set $\mathcal{D}_{ss}$ of SSAs, an initial theory $\mathcal{D}_{S_0}$, and a unique name assumption theory $\mathcal{D}_{una}$, and let α be a ground action term. Denote

$\Delta_F = \{\bar{t} \mid \bar{x} = \bar{t}$ appears in a positive effect or a negative effect part of an SSA $\varphi \in \mathcal{D}_{ss}$ instantiated with α and equivalently rewritten wrt $\mathcal{D}_{una}\}$,

$\Omega(s) = \{F(\bar{t}, s) \mid \bar{t} \in \Delta_F\}$.

Notice $\Omega(S_0)$ is a finite set of ground atoms to be forgotten. Recall from (Lin and Reiter 1994) that forgetting several ground atoms can be done consecutively in any order.

An *instantiation* of $\mathcal{D}_{ss}$ wrt α and $\Omega(S_0)$, denoted by $\mathcal{D}_{ss}[\Omega(S_0)]$, is the set of formulas of the form:

$$F(\bar{t}, do(\alpha, S_0)) \leftrightarrow \gamma_F^+(\bar{t}, \alpha, S_0) \vee F(\bar{t}, S_0) \wedge \neg\gamma_F^-(\bar{t}, \alpha, S_0),$$

where $\gamma^+_F(\bar{t}, \alpha, S_0), \gamma^-_F(\bar{t}, \alpha, S_0)$ are formulas uniform in S_0 obtained from instantiation. Observe that $\mathcal{D}_{ss}[\Omega(S_0)]$ effectively defines new values for the fluents affected by the action α. However, these definitions use fluents wrt S_0, which may include the values to be forgotten. For this reason, forgetting should be performed both in $\mathcal{D}_{S_0}$ and in $\mathcal{D}_{ss}[\Omega(S_0)]$.

Proposition 5 (Th. 3.6 in (Liu and Lakemeyer 2009)) *In the notations above, the following is a progression of $\mathcal{D}_{S_0}$ wrt α in the sense of Definition 5:*

$$\mathcal{D}_{S_\alpha} = [\ forget(\mathcal{D}_{ss}[\Omega(S_0)] \cup \mathcal{D}_{S_0},\ \Omega(S_0))\ ](S_\alpha/S_0)$$

Thus, computing progression in a local-effect $\mathcal{BAT}$ is an effective syntactic transformation of the initial theory, which leads to the *unique* form of the updated theory $\mathcal{D}_{S_\alpha}$. This fact is used in the proof of Theorem 1.

Now we are ready to formulate the results about decomposability and inseparability properties under progression. We start with negative examples in which a $\mathcal{BAT}$ is local-effect and initial theories are formulated in first-order logic. As progression $\mathcal{D}_{S_\alpha}$ is a set of formulas uniform in S_α, and this situation term may occur in every formula of $\mathcal{D}_{S_\alpha}$ (hence, potentially spoiling decomposability), we consider the decomposability and inseparability properties wrt the theory $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$ instead of $\mathcal{D}_{S_\alpha}$. Otherwise, every time we would have to speak of $\Delta \cup \mathtt{sig}\,(S_\alpha)$–decomposability of progression instead of Δ–decomposability (notice that $\mathtt{sig}\,(S_\alpha)$ includes $\mathtt{sig}\,(\alpha)$, which includes constants – arguments of α). Informally, $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$ serves as a new initial theory uniform in S_0.

Consider a $\mathcal{BAT}$ $\mathcal{D}$ with a Δ–decomposable initial theory $\mathcal{D}_{S_0}$ for a signature Δ. The general syntactic form of SSAs provides enough flexibility to design examples showing loss or gain either of the decomposability property of $\mathcal{D}_{S_0}$ or inseparability of its components. As context conditions in an SSA may contain symbols that are even not present in $\mathtt{sig}\,(\mathcal{D}_{S_0})$, or symbols from both components of $\mathcal{D}_{S_0}$ (if decomposition exists), this should not be a surprise for the reader. Therefore, it makes sense to restrict our study to those $\mathcal{BAT}$s, where SSAs have one of the well-studied forms, for instance, to local-effect theories. It turns out that this form is still general enough to formulate negative results showing that the mentioned properties are not preserved without further stipulations. First, we provide a trivial Example 3 showing that the decomposability property of the initial theory can be easily lost under progression. This example is given rather as a simple illustration of progression for readers new to this notion. Next, we show that Δ–inseparability of components of initial theory $\mathcal{D}_{S_0}$ can be easily lost when fluents are present in Δ (Example 4). These observations hold already for local-effect $\mathcal{BAT}$s and follow from the fact that after progression some new information from SSAs can be added to initial theory which spoils its component properties. We only need to provide a combination of an initial theory with a set of SSAs appropriate for this purpose.

Example 3 (Decomposability lost under progression) *Consider a basic action theory $\mathcal{D}$ with $\{F, A, c_1, c_2\} \subseteq \mathtt{sig}\,(\mathcal{D})$, where F is a ternary fluent, A is a binary action function, and c_1, c_2 are object constants. Let the theory $\mathcal{D}_{ss}$ consist of the single axiom*

$$F(x, y, do(a, s)) \leftrightarrow\ (a = A(x, y) \vee F(x, y, s)\ ),$$

and let the initial theory $\mathcal{D}_{S_0}$ consist of two formulas $Taut(c_1)$ and $Taut(c_2)$ uniform in S_0, which are tautological sentences in signature $\{c_1\}$ and $\{c_2\}$, respectively. Clearly, $\mathcal{D}_{S_0}$ is $\varnothing$–decomposable theory. On the other hand, progression $\mathcal{D}_{S_\alpha}$ of $\mathcal{D}_{S_0}$ wrt action $\alpha = A(c_1, c_2)$ is equivalent to the theory consisting of the ground atom $F(c_1, c_2, do(\alpha, S_0))$. This can be verified following Definition 5 directly, or by Proposition 5, since $\mathcal{D}$ is local-effect. Anyway, it is easy to check that $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$ (and $\mathcal{D}_{S_\alpha}$, as well) is not Δ–decomposable theory (for any Δ).

For a signature Δ, with $S_0 \in \Delta$, and a unary action $A(c)$, we now give an example of a local-effect BAT $\mathcal{D}$ with an initial theory $\mathcal{D}_{S_0}$ Δ–decomposable into finite Δ–inseparable components, such that progression $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$ of $\mathcal{D}_{S_0}$ wrt $A(c)$ (with term S_α substituted with S_0) is finitely axiomatizable and Δ–decomposable, but the decomposition components are no longer Δ–inseparable.

Example 4 (Δ–inseparability is lost if a fluent is in Δ) *Consider a BAT $\mathcal{D}$ with $\{F, P, Q, R, A, b\} \subseteq \mathtt{sig}\,(\mathcal{D})$, where F is a fluent, P, Q, R are predicates, A is an action function, and b is an object constant. Let $\Delta = \{F, R, S_0\}$ and define sub-theories of $\mathcal{D}$ as follows:*

- $\mathcal{D}_{ss} = \{F(x, do(a, s)) \leftrightarrow\ (a = A(x) \wedge P(x) \wedge Q(d)\ \vee\ F(x, s)\ )\}$;
- $\mathcal{D}_{S_0} = \mathcal{D}_1 \cup \mathcal{D}_2$, *with*
  - $\mathcal{D}_1 = \{Taut(F, R, S_0, b), \neg F(x, S_0)\}$, *where b is a constant and $Taut(F, R, S_0, b)$ is a tautological formula in the signature $\{F, R, S_0, b\}$, uniform in S_0,*
  - $\mathcal{D}_2 = \{P(x) \rightarrow \exists y(R(x, y) \wedge P(y)),\ \neg F(x, S_0)\}$.

Let $\Delta = \{F, R, S_0\}$. By the syntactic form, $\mathcal{D}_{S_0}$ is Δ–decomposable: we have $\mathcal{D}_{S_0} = \mathcal{D}_1 \cup \mathcal{D}_2$, $\mathtt{sig}\,(\mathcal{D}_1) \cap \mathtt{sig}\,(\mathcal{D}_2) = \Delta$, $\mathtt{sig}\,(\mathcal{D}_1) \setminus \Delta = \{b\}$, and $\mathtt{sig}\,(\mathcal{D}_2) \setminus \Delta = \{P\}$. It is also easy to check that $\mathcal{D}_1$ and $\mathcal{D}_2$ are Δ–inseparable.

Note that $\mathcal{D}_{ss} \models F(x, do(A(c), S_0)) \leftrightarrow (x = c) \wedge P(c) \wedge Q(d) \vee F(x, S_0)$, the result of substitution of ground action $\alpha = A(c)$ and situation constant S_0 into SSA. As $\mathcal{D}_{S_0} \models \neg F(x, S_0)$, we have $\mathcal{D}_{ss} \cup \mathcal{D}_{S_0} \models F(c, do(A(c), S_0)) \leftrightarrow P(c) \wedge Q(d)$; denote the last formula by φ.

By Proposition 5 it is easy to verify that the union of $\{Taut(F, R, S_0, b)\}$ and $\mathcal{D}'_2 = (\mathcal{D}_2 \setminus \{\neg F(x, S_0)\}) \cup \{\varphi,\ x \neq c \rightarrow \neg F(x, do(A(c), S_0))\}$ is a progression ($\mathcal{D}_{S_\alpha}$) of $\mathcal{D}_{S_0}$ wrt $A(c)$. By the syntactic form, $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$ is Δ–decomposable theory. On the other hand, we have $\varphi \models F(c, do(A(c), S_0)) \rightarrow P(c)$, thus $\mathcal{D}'_2(S_0/S_\alpha) \models \{F(c, S_0) \rightarrow \exists y R(c, y),\ \ F(c, S_0) \rightarrow [\exists y \exists z R(c, y) \wedge R(y, z)], \ldots\}$, and hence $\mathcal{D}'_2(S_0/S_\alpha)$ entails the same infinite set of formulas, where c is replaced by an existentially quantified variable. By compactness, it is not finitely axiomatizable by formulas of first order logic in signature Δ and it is not hard to verify that $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$ can not have a decomposition into Δ–inseparable components.

We note that the example is based on the observation about finite non-axiomatizablity used in the literature on DLs (e.g., see Section 3.2 in (Lutz and Wolter 2009)). There is a plenty of flexibility to formulate similar examples with help of non-tautological formulas which syntactically "bind" symbols F, R, b, S_0 in theory $\mathcal{D}_1$. The given example motivates the following definition.

Definition 6 (Fluent–free signature) *A signature Δ is called fluent–free if no fluent is contained in Δ.*

To formulate the theorem below we let $\mathcal{F}$ denote the set of all fluents. Essentially, the conditions of the theorem are designed to guarantee componentwise computation of progression for a decomposable initial theory. Let a finite set $\mathcal{D}_{ss}$ of the SSAs be syntactically divided into the union of $|I|$ sub-theories sharing some fluent-free signature Δ_1 (that may include actions, static predicates, and object constants). Also, let the initial theory $\mathcal{D}_{S_0}$ be Δ_2–decomposable, for a fluent-free signature Δ_2, into $|J|$ components, and the sub-theories of $\mathcal{D}_{ss}$ be aligned with the components of $\mathcal{D}_{S_0}$ via syntactic occurrences of fluents.

Theorem 1 (Preservation of components in local-effect $\mathcal{D}$) *Let $\mathcal{D}$ be a local-effect $\mathcal{BAT}$, with $\mathcal{D}_{S_0}$ an initial theory in first-order logic. Let Δ_1, Δ_2 be fluent-free signatures, $do \notin \Delta_1$, and $\alpha = A(\bar{c})$, be a ground action term. Denote $\Delta = \Delta_1 \cup \Delta_2 \cup \{c_1, \ldots, c_k\}$, if $\bar{c} = \langle c_1, \ldots, c_k \rangle$, and suppose the following:*

- $\mathtt{sig}(\mathcal{D}_{ss}) \cap \mathcal{F} \subseteq \mathtt{sig}(\mathcal{D}_{S_0})$*;*
- *$\mathcal{D}_{ss}$ is the union of theories $\{D_i\}_{i \in I}$, with $\mathtt{sig}(D_n) \cap \mathtt{sig}(D_m) \subseteq \Delta_1 \cup \{do\}$ for all $n, m \in I \neq \varnothing$, $n \neq m$;*
- *$\mathcal{D}_{S_0}$ is Δ_2–decomposable into finite components $\{D'_j\}_{j \in J}$ uniform in S_0;*
- *for every $i \in I$, there is $j \in J$ such that $\mathtt{sig}(D_i) \cap \mathtt{sig}(\mathcal{D}_{S_0}) \subseteq \mathtt{sig}(D'_j)$.*

Then $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$ is Δ–decomposable. If the components $\{D'_j\}_{j \in J}$ are pairwise Δ–inseparable, then so are the components of $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$ in the corresponding decomposition.

We note that the proof of the theorem uses Proposition 5 and the component properties of forgetting from Section 3. SSAs can be grouped into $|I|$ components by drawing a graph with fluent names as vertices, and an edge from the fluent on the left-hand-side of each SSA going to each fluent occurring on the right-hand-side of the same SSA. Similarly, it is easy to check the last condition in Theorem 1 that guarantees alignment of groups of axioms in SSAs with decomposition components of $\mathcal{D}_{S_0}$. In the above conditions, observe that if an action A occurs in active position of SSAs from two different sub-theories of $\mathcal{D}_{ss}$, then computing progression may involve forgetting in two corresponding components of $\mathcal{D}_{S_0}$ and potentially cause occurrence of common Δ_1–symbols in the components of progression. A practically important class of $\mathcal{BAT}$s for which this interference can be avoided is described in the corollary below. Note the first condition in the corollary which yields that every action mentioned in $\mathcal{BAT}$ can have effect on fluents only from one component of $\mathcal{D}_{ss}$.

Corollary 2 (Strong preservation of components) *For every ground action term $\alpha = A(\bar{c})$, where $\bar{c} = \langle c_1, \ldots, c_k \rangle$, in the conditions and notations of Theorem 1, if:*

- *no action function is in Δ_1,*
- *whenever A is in active position in an SSA for a fluent F and $F \in \mathtt{sig}(D'_j)$ for some $j \in J$, we have $\{c_1, \ldots, c_k\} \subseteq \mathtt{sig}(D'_j)$,*

then $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$ is Δ_2–decomposable into Δ_2–inseparable components.

The corollary obviously remains true if the first condition is replaced with the simple requirement: $\Delta_1 \subseteq \Delta_2$.

Example 1 (continuation). Note that the $\mathcal{BAT}$ considered in the example satisfies the conditions of the corollary with signatures $\Delta_1 = \varnothing$ and $\Delta_2 = \{Block, S_0\}$. The theory $\mathcal{D}_{ss}$ is a union of two theories, with the intersection of signatures equal to $\{do\}$. As already noted in the example, the initial theory $\mathcal{D}_{S_0}$ is Δ_2–decomposable into Δ_2–inseparable components. Now, consider the ground action $\alpha = move(A, B, C)$. By Corollary 1 and Proposition 5, in order to compute the theory $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$ (the progression of $\mathcal{D}_{S_0}$ wrt α, with the term S_α substituted with S_0), it suffices to forget the ground atoms $On(A, B, S_0)$ and $Clear(C, S_0)$ in the first decomposition component of $\mathcal{D}_{S_0}$ and update it with the ground atoms $On(A, C, S_0)$ and $Clear(B, S_0)$. The second component of $\mathcal{D}_{S_0}$ remains unchanged. One can check that $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$ is the union of the following theories:

$\varphi \wedge \psi \wedge (x \neq C) \rightarrow Clear(x, S_0)$
$\psi \rightarrow Block(x)$
$Block(B) \wedge Block(C) \wedge On(A,C,S_0) \wedge \neg On(A,B,S_0)$
$Clear(A, S_0) \wedge Clear(B, S_0) \wedge \neg Clear(C, S_0)$

and

$(Top(x, S_0) \vee Inheap(x, S_0)) \rightarrow \neg Block(x)$
$\exists x\ Block(x)$,

where φ and ψ, respectively, stand for

$(x \neq B) \wedge \neg \exists y\ ((y \neq A \vee x \neq B) \wedge On(y, x, S_0))$,
$(x = A) \vee \exists y\ ((x \neq A \vee B \neq y) \wedge On(x, y, S_0))$.

The theory $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$ is Δ_2–decomposable by the syntactic form and there is no need to compute a decomposition again after progression. Corollary 2 guarantees that the obtained components are Δ_2–inseparable and that we can compute progression for arbitrary long sequences of actions while preserving decomposability of $\mathcal{D}_{S_\alpha}(S_0/S_\alpha)$ and inseparability of its components.

5 Discussion

As a rule, decomposing a given theory wrt a signature happens to be of the same complexity as entailment in the underlying logic. The tasks of computing Δ–decompositions and checking for inseparability is a related, but different ongoing research topic, e.g., addressed in the literature on description logics, see (Konev et al. 2010; 2009) presenting widely-applicable results. We have noted natural sufficient conditions for the two component properties of theories, decomposability and inseparability, to be preserved under progression of local-effect theories. This has required a new understanding of progression and the related notion of forgetting wrt modularity of theories. Given a decomposition of the initial theory into inseparable components, the rest of the conditions in the main results, Theorem 1 and Corollary 2, are purely syntactical, easy to check, and natural to hold, judging from experience of formalizing composite domains in the situation calculus. The important observation behind these results is that in order to compute progression of an initial theory wrt an action having effects only on fluents from one decomposition component, it suffices to compute forgetting only in this component.

Acknowledgements The first author was supported by the German Research Foundation within the Transregional Collaborative Research Centre SFB/TRR 62 “Companion-Technology

for Cognitive Technical Systems", the Russian Academy of Sciences (Grant No. 15/10), and the Siberian Division of the Russian Academy of Sciences (Integration Project No. 3). Both authors would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Dept. of Computer Science of the Ryerson University for providing partial financial support.

References

Amir, E. 2000. (De)composition of situation calculus theories. In Kautz, H. A., and Porter, B. W., eds., *AAAI/IAAI*, 456–463. AAAI Press / The MIT Press.

Amir, E. 2002. Projection in decomposed situation calculus. In Fensel, D.; Giunchiglia, F.; McGuinness, D. L.; and Williams, M.-A., eds., *KR*, 315–326. Morgan Kaufmann.

Cook, S. A., and Liu, Y. 2003. A complete axiomatization for blocks world. *J. Log. Comput.* 13(4):581–594.

Craig, W. 1957. Three uses of the herbrand-gentzen theorem in relating model theory and proof theory. *J. Symb. Log.* 22(3):269–285.

Craig, W. 2008. The road to two theorems of logic. *Synthese* 164(3):333–339.

Fox, D., and Gomes, C. P., eds. 2008. *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*. AAAI Press.

Grüninger, M.; Hahmann, T.; Hashemi, A.; Ong, D.; and Özgövde, A. 2012. Modular first-order ontologies via repositories. *Applied Ontology* 7(2):169–209.

Gu, Y., and Soutchanski, M. 2008. Reasoning about large taxonomies of actions. In Fox and Gomes (2008), 931–937.

Gu, Y., and Soutchanski, M. 2010. A description logic based situation calculus. *Ann. Math. Artif. Intell.* 58(1-2):3–83.

Inclezan, D., and Gelfond, M. 2011. Representing biological processes in modular action language ALM. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*. AAAI.

Kakas, A. C.; Michael, L.; and Miller, R. 2011. Modular-$\mathcal{E}$ and the role of elaboration tolerance in solving the qualification problem. *Artif. Intell.* 175(1):49–78.

Konev, B.; Lutz, C.; Walther, D.; and Wolter, F. 2009. Formal properties of modularisation. In Stuckenschmidt, H.; Parent, C.; and Spaccapietra, S., eds., *Modular Ontologies*, volume 5445 of *Lecture Notes in Computer Science*. Springer. 25–66.

Konev, B.; Lutz, C.; Ponomaryov, D.; and Wolter, F. 2010. Decomposing description logic ontologies. In Lin, F.; Sattler, U.; and Truszczynski, M., eds., *KR*. AAAI Press.

Kourousias, G., and Makinson, D. 2007. Parallel interpolation, splitting, and relevance in belief change. *J. Symb. Log.* 72(3):994–1002.

Lin, F., and Reiter, R. 1994. Forget it! In *Proceedings of the AAAI Fall Symposium on Relevance*, 154–159.

Lin, F., and Reiter, R. 1997. How to progress a database. *Artificial Intelligence* 92:131–167.

Liu, Y., and Lakemeyer, G. 2009. On first-order definability and computability of progression for local-effect actions and beyond. In Boutilier, C., ed., *IJCAI*, 860–866.

Liu, Y., and Levesque, H. J. 2005. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In Kaelbling, L. P., and Saffiotti, A., eds., *IJCAI*, 522–527. Professional Book Center.

Lutz, C., and Wolter, F. 2009. Mathematical logic for life science ontologies. In Ono, H.; Kanazawa, M.; and de Queiroz, R. J. G. B., eds., *WoLLIC*, volume 5514 of *Lecture Notes in Computer Science*, 37–47. Springer.

Lutz, C., and Wolter, F. 2010. Deciding inseparability and conservative extensions in the description logic el. *J. Symb. Comput.* 45(2):194–228.

McCarthy, J., and Hayes, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence*, volume 4. Edinburgh University Press, Reprinted in (McCarthy 1990). 463–502.

McCarthy, J. 1990. *Formalization of common sense: papers by John McCarthy edited by V. Lifschitz*. Norwood, N.J.: Ablex.

Pirri, F., and Reiter, R. 1999. Some contributions to the metatheory of the situation calculus. *Journal of the ACM* 46(3):325–364.

Ponomaryov, D. 2008. On decomposibility in logical calculi. *Bulletin of the Novosibirsk Computing Center* 28:111–120.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. The MIT Press.

Soutchanski, M., and Yehia, W. 2012. Towards an expressive practical logical action theory. In Voronkov, A., ed., *Turing-100*, volume 10 of *EPiC Series*, 307–325. EasyChair.

Vassos, S., and Levesque, H. J. 2008. On the progression of situation calculus basic action theories: Resolving a 10-year-old conjecture. In Fox and Gomes (2008), 1004–1009.