Reasoning About Property Preservation in Adaptive Case Management

Rik Eshuis
Richard Hull
Mengfei Yi

Beta Working Paper series 484

BETA publicatie	WP 484 (working paper)
ISBN	
ISSN	
NUR	804
Eindhoven	August 2015

Reasoning About Property Preservation in Adaptive Case Management

Rik Eshuis[1], Richard Hull[2], and Mengfei Yi[1]

[1] School of Industrial Engineering, Eindhoven University of Technology, Netherlands
[2] IBM T J Watson Research Center, USA

Abstract. Adaptive Case Management (ACM) has emerged as a key BPM technology for supporting unstructured business process, and has been used to support flexible services orchestration. A key problem in ACM is that case schemas need to be changed to best fit the case at hand. Such changes are ad-hoc, and may result in schemas that do not reflect the intended logic or properties. This paper presents a formal approach for reasoning about which properties of a case schema are preserved after a modification, and describes change operations that are guaranteed to preserve certain properties. The Case Management model used here is a variant of the Guard-Stage-Milestone model for declarative business artifacts. Applicability is illustrated using a real-life example.

1 Introduction

Case management has been introduced to support knowledge intensive business processes, which are organized around data artifacts [28, 8, 24]. Case management often needs to support flexible business processes that are performed by knowledge workers. So case management schemas must be easy to change. Adaptive Case Management (ACM) has been proposed as umbrella term for flexible case management [22]. Case Management has been applied in many knowledge-worker driven application areas, including fraud detection, healthcare, education, and social work, and has also been used as a basis to support flexible services orchestration to enable collaboration between enterprises (e.g., [17, 19]).

Designing case management models is hard. The presence of business rules may make it difficult to assess and predict the behavior specified in a case management model or schema. However, changing case management schemas is even harder. Unwanted behavior such as logical errors can be easily introduced by changing a case management model. More generally, a change could have undesirable side effects. Therefore, certain user-defined properties should be preserved in the changed schema.

This paper studies conditions under which case management schemas can be changed while preserving specified properties. We use the Guard-Stage-Milestone (GSM) model; GSM schemas declaratively specify life-cycles of business artifacts. The meta-model underlying the OMG standard Case Management Model and Notation (CMMN) [3] is based on GSM. In this paper we use a restricted variant of the GSM model, called Fully Acyclic GSM, to enable a focus on the key ideas and the development of informative and useful results. We leave for future research the generalization of the approach to richer variants of GSM.

The paper makes three fundamental contributions. First, we develop a precise definition for testing the preservation of properties. This is based on the notion of *conditional emulatability*, which allows to specify a condition under which executions of one GSM schema can be imitated by executions of a second GSM schema, including having exactly the same behavior on selected output attributes. Second, we develop a general-purpose "*Lifting Lemma*". Speaking intuitively, this provides a mechanism for isolating changes to a "local area" in a GSM schema. The designer can prove preservation properties at the local level, and then apply the Lifting Lemma to infer the preservation of properties at the "global level". The Lifting Lemma can also be used to specify best practices for schema change operations, which ensure a form of modularity and guarantee the preservation of selected properties. And third, we use the Lifting Lemma to show how *key change operations* can be defined so as to guarantee the preservation of certain properties. Importantly, the theoretical work is motivated by examples arising in a real-world application.

The remainder of this paper is structured as follows. Section 2 introduces the problem of changing GSM schemas based on a real-world example, and illustrates change operations that preserve specified properties. Section 3 formally introduces the GSM model used in this paper. Section 4 develops the Lifting Lemma, and Section 5 illustrates applications of the Lifting Lemma by defining general-purpose change operations that preserve selected properties. Section 6 describes related work, and Section 7 offers brief conclusions.

2 Motivation

To introduce the problem of variability, we consider an example based on a real-world process from an international technology company, which has offices in different geographic regions of the world. In the process, business criteria for partner contracts are assessed. Each region has its own flavor of the process.

Example 2.1: The base process, called here BCA^{base}, is used for the main region and has the following activities (see Figure 1). First, data is gathered needed to perform the assessment. Next, two activities are performed in parallel as a pre-check. The credit is checked to ensure that the credit limit of the partner is still valid. In parallel, the past performance of the partner is evaluated and checked. If both checks are successful, the pre-check succeeds and a detailed check is performed, which may either succeed or fail. If the pre-check has succeeded within three weeks, a bonus is paid to the team managing the deal.

Fig. 1 shows the lifecycle part of the GSM schema for this process. The lifecycle contains stages (rounded rectangles) which represent the business activities (in this paper, these are essentially (atomic) tasks that are not explicilty modeled within the GSM schema). Guards (diamonds) specify under which condition work in a stage is launched. Milestones (circles) represent business objectives that are achieved by stages to which they are attached or by important events. Guards and milestones have sentries (business rules) that specify when they are executed; these are shown in Tables 1 and 2. Sentries implicitly specify dependencies between stages and milestones: for instance, the sentry of the guard of stage Credit Check states that the stage is opened if milestone IDGS has been

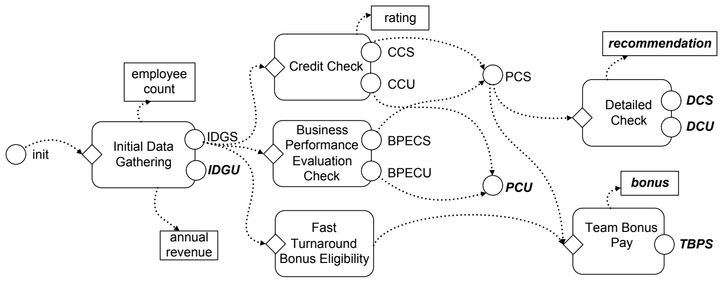

Fig. 1. Main Business Criteria Assessment process (BCA^{base})

Stage	**Guard**
Initial Data Gathering	init
Credit Check	IDGS
Business Performance Evaluation Check	IDGS
Detailed Check	PCS
Fast Turnaround Business Eligibility	IDGS
Team Bonus Pay	C:Fast Turnaround Business Eligibility $\wedge$ fast_turnaround $\wedge$ PCS

Table 1. Stages and guards for BCA^{base} in Fig. 1

achieved, so the guard of Credit Check depends on IDGS. The dependencies are graphically depicted using dashed arrows in Fig. 1. (Our diagramatic convention does not explictly indicate how multiple milestones are combined in a sentry, e.g., the sentry for PCS; please refer to the tables.) Rectangles represent data attributes. A dashed line from a stage to a data attribute indicates that the stage computes a value for the data attribute. To compare different GSM schemas, we make use of output attributes, depicted in bold italics, which can be milestones or data attributes. Some attributes are not shown (spez., fast_turnaround computed by Fast Turnaround Business Eligibility and BP_good computed by Business Performance Evaluation Check).

The behavior of GSM schemas is driven by event occurrences, which are typically the result of completion of a stage execution. In response to an event occurrence, a B(usiness)-step is taken, in which as many sentries as possible are applied. For instance, suppose that in some "snapshot", i.e., the state of an artifact instance at some time during its execution, the milestone BPECS is true and stages Credit Check and Fast Turnaround Bonus Eligibility are the only open stages. If stage completion event C:Credit Check now occurs with value 9 for rating, then milestone CCS gets achieved. The milestone PCS also gets achieved, and also stage Detailed Check is opened (and thus, the external activity associated with that stage is started). At this point no further sentries can be applied, the B-step is finished, and the new snapshot has been computed. (See also Example 3.7 below.) □

We next present three variations on this example.

Milestone	**Full Name**	**Sentry**
IDGS	Initial Data Gathering Successful	C:Initial Data Gathering $\wedge \ldots$
IDGU	Initial Data Gathering Unsuccessful	C:Initial Data Gathering $\wedge \ldots$
CCS	Credit Check Successful	C:Credit Check $\wedge$ rating ≥ 8
CCU	Credit Check Unsuccessful	C:Credit Check $\wedge$ rating < 8
BPECS	Business Performance Evaluation Check Successful	C:Business Performance Evaluation Check $\wedge$ BP_good
BPECU	Business Performance Evaluation Check Unsuccessful	C:Business Performance Evaluation Check $\wedge \neg$ BP_good
PCS	Pre-checks Successful	CCS $\wedge$ BPECS
PCU	Pre-checks Unsuccessful	CCU $\vee$ BPECU
DCS	Detailed Check Successful	C:Detailed Check $\wedge \ \ldots$
DCU	Detailed Check Unsuccessful	C:Detailed Check $\wedge \ \ldots$
TBPS	Team Bonus Pay Successful	C:Team Bonus Pay

Table 2. Milestones for BCA^{base} in Fig. 1

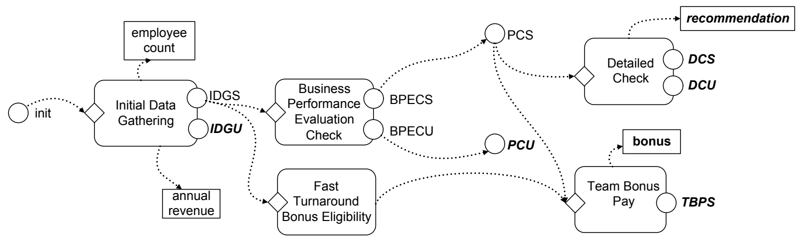

Fig. 2. Process BCA^{del} resulting after applying change of Example 2.3

Example 2.2: In another region, the business performance of a partner is evaluated and checked only if the partner has more than 300 employees. The GSM schema of Fig. 1 is changed as follows (the other sentries are not changed):

– The guard of stage Business Performance Evaluation Check becomes IDGS $\wedge$ employee_count ≥ 300.
– Milestone PCS can be achieved via extra sentry CCS$\wedge$employee_count < 300.

Now the question arises how the change affects cases. We would like to assert that for partners with 300 or more employees, the new GSM schema emulates the behavior of the old GSM schema, and the old GSM schema emulates that of the new, so for the same cases, the same output results in both schemas. ('Emulates' is defined precisely in Definition 4.4 below.) For partners with less than 300 employees, this assertion does not hold. In particular, it may be that a company with say 290 employees and a poor performance is accepted under the new schema but rejected under the old schema. Example 5.3 will illustrate how the formalism and results of this paper can be applied to prove these properties. □

Example 2.3: Consider again the base process BCA^{base} of Example 2.1. In yet another region, the credit of the partner is not checked. Schema BCA^{base} is

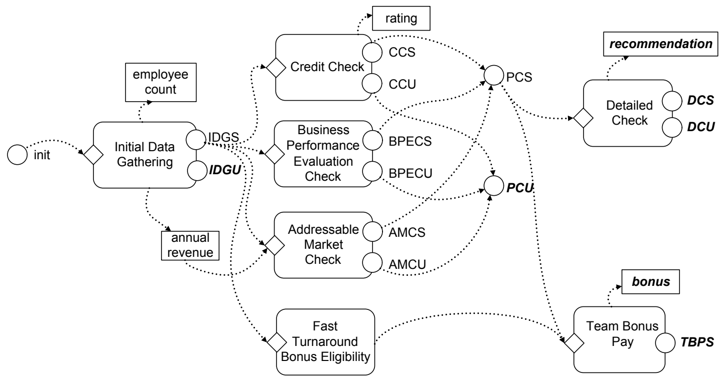

Fig. 3. Process BCA^{ins} after applying change of Example 2.4

changed by deleting stage Credit check and milestones CCS and CCU, as visualized in BCA^{del} in Fig. 2. The sentries of milestones PCS and PCU need to change as follows (the other sentries are not changed):

– The sentry of milestone PCS becomes BPECS.
– The sentry of milestone PCU becomes BPECU.

To characterize the change, we would like to assert that for cases under the old schema for which the credit check was successful, the new schema emulates the old schema. For cases of partners for which the credit check was unsuccessful in Fig. 1 there is a difference: for those cases the detailed check can be performed as in Fig. 2. This example will be revisited in Examples 4.5 and 4.11. $\square$

Example 2.4: Consider again the base process BCA^{base}. In a fourth region, the market addressed by the partner is assessed. Stage Addressable Market Check is inserted with milestones AMCS (Addressable Market Check Successful) and AMCU (Addressable Market Check Unsuccessful); see BCA^{ins} in Fig. 3. The sentries need to change as follows (the other sentries are not changed):

– The guard of stage Addressable Market Check becomes $\mathsf{IGDS} \wedge \mathsf{annual\_revenue} \geq \$500K$;
– The sentry of milestone PCS is replaced with two sentries: $\mathsf{CCS} \wedge \mathsf{BPECS} \wedge \mathsf{AMCS}$ and $\mathsf{CCS} \wedge \mathsf{BPECS} \wedge \mathsf{annual\_revenue} < \$500K$.
– The sentry of milestone PCU becomes $\mathsf{CCU} \vee \mathsf{BPECU} \vee \mathsf{AMCU}$.

The change assertion is that for cases in which the annual revenue is lower than $\$500K$, the old schema emulates the new schema and vice versa. Also for cases in which the annual revenue is higher or equal to $\$500K$ *and* the milestone AMCS gets achieved, the old and the new schema emulate each other. This will be revisited in Example 5.12. $\square$

In the remainder of this paper, we develop formal machinery that precisely defines the impact of each change on the GSM schema and also characterizes which properties of cases are preserved when a change is applied. We revisit these examples to illustrate the salient points.

3 The Formal GSM Model

This section presents formal definitions for the variant of GSM used in this paper. This includes a specific notion of "executions" of a GSM schema, that will be important in our reasoning about property preservation. It is assumed that the reader is familiar with the basic aspects of the formal definitions of GSM (e.g., as in [16, 6, 9]).

The development here imposes a family of restrictions on the GSM variants of, e.g., [16, 9], to enable the development of interesting theoretical properties concerning schema evolution. Speaking intuitively, the key restrictions and variations for GSM studied in this paper are as follows. (See below for more detail.) The first two restrictions are primarily cosmetic and do not significantly impact the expressive power of GSM schemas, and the third is a variant studied in previous papers. The other ones are more impactful, and enable a form of monotonicity the enables the results about property preservation developed in this paper. These are inspired by the Case Management model of [28].

1. *Explicit modeling of "output attributes:* This is done to streamline notions of equivalence between schemas.
   Flat stage hierarchy, i.e., no nesting of stages.
2. *Single artifact type and single artifact instance.*
3. *Sentries without events are triggered when they change value.* This semantics is also used in [9].
4. *Stage attributes not permitted in sentries.* (Stage completion events are modeled and can trigger sentries. Also, a stage's completion can be tested by testing whether its output attributes have assigned values.)
5. *Acyclicity of the dependency graph.*
6. *Unique attribute writer*: For each data attribute there is exactly one stage that can assign values to that attribute.
7. *Strict sentry satisfaction*: Intuitively, this ensures that a sentry is not assigned a value until values have been assigned to a set of relevant attributes (see below).
8. *No terminating sentries for stages, no invalidating sentries for milestones*: This means that the sole mechanism for aborting the execution of a stage is a roll-back.
9. *Restricted form of rollback*, that is compliant with the dependency graph.

As will be seen, the combination of these restrictions will also imply the following:

10. *Assign once*: Except in the case of roll-back, each attribute starts with null value, and is assigned a non-null value at most once during execution.

Generalization and adaptation of these results to richer variants of GSM, and to the full GSM model, are left for future research.

These assumptions enable a streamlined approach for the formal definitions of GSM schema and operational semantics. Because there is no stage hierarchy, in this paper we are able to blur the distinction between stages and the tasks that they contain.

We assume three infinite disjoint sets of names, for *data attributes*, for *milestones*, and for *stages*. Each data attibute a has a type $type(a)$ which is scalar (e.g., string, character, integer, float, etc.), or is a set of records of scalars. Milestones can be used as attributes with type Boolean. Both data attributes and milestones may take the unassigned (or null) value (denoted $\perp$).

We assume a condition language $\mathcal{C}$ that includes fixed predicates over scalars (e.g., '$\leq$' over integers or floats), and Boolean connectives. Quantification and testing set membership is supported for working with the set-valued attributes. The condition formulas may involve stage, milestone, and data attributes. All attributes start with undefined value ($\perp$). Milestones will take the value *True* if one of their sentries go true. Stages will take the value *True* at the time when they complete. (This is a variation on the traditional behavior of stage attributes.)

A *sentry* ψ has one of the three forms: "φ", "$\mathsf{C}{:}S$", or "$\mathsf{C}{:}S \wedge \varphi$", where φ is a condition formula ranging over the attributes of Γ. Here "$\mathsf{C}{:}S$" is called the *completion event* for stage S. Also, $\mathsf{C}{:}S$ (if present) is the *completion event* for ψ and φ (if present) is the *formula* for ψ. Sentries having the first form are called *eventless*, and sentries having the latter two forms are called *event-based*.

Definition 3.1: A GSM *schema* is a 5-tuple $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{out}, sen, sig)$ where:

1. Att_d is a finite set of data attributes.
2. Att_m is a finite set of milestone attributes.
3. Att_S is a finite set of stage attributes.
4. $m_{\text{start}} \in Att_m$ is called the *start milestone*. It is used as a mechanism for launching an execution of Γ.
5. $Att_{out} \subseteq Att_d \cup Att_m$ is the set of *output* attributes for Γ. This set is also denoted as $out(\Gamma)$.
6. The *sentry assignment sen* is a function from $Att_S \cup (att_m - \{m_{\text{start}}\})$ to sets of sentries with formulas in the condition language $\mathcal{C}$ ranging over Att, and such that if there is a completion event $\mathsf{C}{:}S$ then $S \in Att_S$. Each element of set $sen(v)$ for $v \in Att_S \cup Att_m$ is called a *sentry* of v.
7. The *signature assignment sig* is a function from Att_S to pairs (I, O) of finite sets of attributes from Att_d, i.e., $sig : Att_S \to \mathcal{P}^{\text{fin}}(Att_d) \times \mathcal{P}^{\text{fin}}(Att_d)$. If $sig(S) = (I, O)$, then we denote I as $sig_{in}(S)$, called the *input* of S, and denote O as $sig_{out}(S)$, called the *output* of S.

Definition 3.2: Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{out}, sen, sig)$ be a GSM schema. The *dependency graph* of Γ, denoted $DG(\Gamma)$, is a directed graph (V, E) where

1. The set of vertices $V = Att_S \cup Att_m$.
2. For $m \in Att_m$ and $v \in V$, $(m, v) \in E$ if the attribute m occurs in the sentry $sen(v)$ of (milestone or stage) v.
3. For $S \in Att_S$ and $v \in V$, $(S, v) \in E$ if some attribute in $sig_{out}(S)$ occurs in a sentry in $sen(v)$.
4. For $S, S' \in Att_S$, $(S, S') \in E$ if $sig_{out}(S) \cap sig_{in}(S') \neq \emptyset$, i.e, if some output of S is an input of S'.

Schema Γ is *Fully Acyclic* if $DG(\Gamma)$ is a directed acyclic graph. In this case we say that Γ is an *FA-GSM schema*.

Note that for a GSM schema Γ, there is no incoming edge for the start milestone in $DG(\Gamma)$. This is because the start milestone has no sentry.

We comment further on differences between the GSM variant used here and the variants of previous work, e.g., [6, 9]. If Γ is FA-GSM, then what about the polarized dependency graph of Γ as defined in the earlier work? First, because there are no invalidating sentries for milestones, nor terminating sentries for stages, each node in $PDG(\Gamma)$ has positive polarity. Second, because the dependency graph of Γ is acyclic, so is the polarized dependency graph. In particular, each FA-GSM schema is a GSM schema in the sense of the earlier work. Further, the equivalence theorem for B-steps developed there also holds for FA-GSM schemas.

Definition 3.3: For a GSM schema $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{out}, sen, sig)$ a *snapshot* is a mapping σ from Att into values of appropriate type (where some attributes may be assigned the null value $\perp$). For milestone and stage attributes, the only permitted values are $\perp$ and *True*.

In the GSM model used here, milestone and stage attributes will never take the value *False*. This is because such attributes will remain undefined until they become true. We now present the definition of *strict satisfaction* of a sentry for a stage or milestone.

Definition 3.4: Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{out}, sen, sig)$ be a GSM schema. Let μ be a sentry for miletone m, and let φ be the formula of μ. Given a snapshot σ of Γ (where some attributes may have undefined value), φ is *strictly satisfied* by σ, denoted $\sigma \models^{strict} \varphi$, if σ is non-null for each attribute A occurring in φ, and if φ is satisfied by σ.

Now let φ be the formula of a sentry for a stage S. For snapshot σ of Γ, φ is *strictly satisfied* for S by σ, denoted $\sigma \models_S^{strict} \varphi$, if (a) σ is non-null for each attribute in $sig_{in}(S)$, (b) σ is non-null for each attribute occurring in φ, and (c) φ is satisfied by σ.

In particular, if $\sigma \models_S^{strict} \varphi$, then each input attribute for S is defined, and so S can be launched. In this paper we focus on strict satisfaction, and refer to this simply as "satisfaction".

Remark 3.5: In the above semantics, the formula $\neg(x = 1)$ will evaluate to *False* if x is undefined. Under another form of semantics for logics that include undefined attribute values, the formula will take the value of *True* for the case where x is undefined. Note also that under the above semantics, a formula $x = x$ will evaluate to false if x has null value, and will evaluate to true of x has a non-null value. $\Box$

The notion of *B-step* for a FA-GSM schema Γ and snapshot σ is defined as in [6, 9]. Further, it can be verified that the basic equivalence results from [6] apply to FA-GSM schemas. In particular, B-steps computed using the incremental semantics satisfy the Church-Rosser property, i.e., two B-steps that start with the same snapshot and same triggering event will have the same final snapshot and same generated events. In this paper we generally use the incremental semantics when studying a B-step. A formalism for studying the properties of executions according to the incremental semantics is presented next.

Definition 3.6: Let $\mathit{\Gamma} = (\mathit{Att} = \mathit{Att}_d \cup \mathit{Att}_m \cup \mathit{Att}_S, m_{\text{start}}, \mathit{Att}_{out}, \mathit{sen}, \mathit{sig})$ be an FA-GSM schema. An *execution* of $\mathit{\Gamma}$ is a sequence

$$\xi \;=\; (\sigma_{init}, \sigma_0, \alpha_1, \beta_1, \sigma_1, \ldots, \alpha_n, \beta_n, \sigma_n)$$

where

1. σ_{init} is a snapshot of $\mathit{\Gamma}$ with all attributes having null-value except for m_{start}, which has value *True*.
2. For each $i \in [1..n]$,
   (a) for each $j \in [2..n]$ with $j \neq i$, $\mathit{stage}(\beta_i) \neq \mathit{stage}(\beta_j)$, where $\mathit{stage}(S(c_1, \ldots, c_n)) = S$ for each expression having the form $S(c_1, \ldots, c_n)$ where $S \in \mathit{Att}_S$.
   (b) σ_i is a snapshot of $\mathit{\Gamma}$.
   (c) If $i \in [2..n]$, then σ_i is the resulting snapshot of the the B-step starting from σ_{i-1} and incorporating the completion of stage S with payload $c_1, \ldots, c_p$ where $\beta_i = S(c_1, \ldots, c_p)$. If $i \in [2..n-1]$, then α_{i+1} is the set of stages lauched during that snapshot.
   (d) For the case of $i = 0$, σ_0 is the snapshot resulting from a B-step applied to the initial snapshot σ_{init} where m_{start} has transitioned from $\perp$ to *True*, and α_1 is the set of stages launched by that B-step.

The set of executions of $\mathit{\Gamma}$ is denoted $\mathit{Exec}(\mathit{\Gamma})$

The above execution ξ is *terminal* if the B-step resulting from the application of event C:β_n launches no stages, and if for each stage S that was launched, there is a β_j with $\mathit{stage}(\beta_j) = S$ (that is, each launched stage eventually completes). The set of terminal executions is denoted $\mathit{TermExec}(\mathit{\Gamma})$.

In an execution ξ having the form as above, it can be shown that for each $i \in [1..n]$,

1. each stage in α_i does not appear in $\cup_{j=1}^{i-1} \alpha_j$.
2. β_i is an expression of form $S(c_1, \ldots, c_p)$ where S is a stage name in $\cup_{j=1}^{i} \alpha_j$, and $c_1, \ldots, c_p$ are values for the attributes in $\mathit{sig}_{out}(S)$.

In the case of a terminal execution $\sigma_{init}, \sigma_0, \alpha_1, \beta_1, \ldots, \sigma_n$, the set of stages mentioned in $\{\beta_1, \ldots, \beta_n\}$ will equal the set $\cup_1^n \alpha_i$. Also, no sentry will be true in σ_n, i.e., no sentry can be fired to form a B-step from σ_n. (This is true in part because we are using the "becomes true" semantics for eventless sentries.) Thus, a terminal execution cannot be extended, and corresponds intuitively to a complete execution of one instance of $\mathit{\Gamma}$.

Example 3.7: We illustrate the notion of execution by revisiting Example 2.1 and the B-step described there. Snapshots are denoted here by listing all milestones that are true, all stages that are open, and the value of each defined data attribute. In each execution of BCA^{base}, $\sigma_0 = \{\mathsf{Initial\ Data\ Gathering}\}$. After that stage completes, we might arrive at σ_1 that additionally has milestone IDGS true, and each of Credit Check, Business Performance Evaluation Check, and Fast

Turnaround Bonus Eligibility open. Also, α_2 holds these three stage names. The next steps of the execution might be as follows.

$$\begin{aligned}
\beta_2 &= \mathsf{C{:}Business\ Performance\ Evaluation\ Check}\\
\sigma_2 &= \{\mathsf{init}, \mathsf{IDGS}, \mathsf{BPECS},\\
&\qquad \mathsf{Credit\ Check}, \mathsf{Fast\ Turnaround\ Bonus\ Eligibility},\\
&\qquad \mathsf{employee\_count} : 1200,\ \mathsf{annual\_revenue} : \$700\text{K},\ \mathsf{BP\_good} : \mathit{True}\}\\
\alpha_3 &= \emptyset\\
\beta_3 &= \mathsf{C{:}Credit\ Check}\\
\sigma_3 &= \{\mathsf{init}, \mathsf{IDGS}, \mathsf{BPECS}, \mathsf{CCS}, \mathsf{PCS},\\
&\qquad \mathsf{Fast\ Turnaround\ Bonus\ Eligibility}, \mathsf{Detailed\ Check},\\
&\qquad \mathsf{employee\_count} : 1200,\ \mathsf{annual\_revenue} : \$700\text{K},\\
&\qquad\quad \mathsf{rating} : 9,\ \mathsf{BP\_good} : \mathit{True},\}\\
\alpha_4 &= \{\mathsf{Detailed\ Check}\}
\end{aligned}$$

The B-step of Example 2.1 occurs from β_3 to σ_3. □

4 Reasoning about GSM executions

This section develops tools for reasoning about GSM executions, including comparing the executions supported by different FA-GSM schemas. The first subsection introduces the notion of *stage i/o assignments*, used to formally study the possible behaviors of stage executions. The second subsection defines *conditional emulation*, which provides the basis for formally comparing the behaviors of FA-GSM schemas. And the third subsection presents the *Lifting Lemma*.

4.1 Stage i/o assignments

A primary goal of this paper is to study the preservation of properties when transforming an FA-GSM schema Γ^1 into a related FA-GSM schema Γ^2. To accomplish this we study properties of elements of $\mathit{Exec}(\Gamma^1)$ vis-a-vis elements of $\mathit{Exec}(\Gamma^2)$. *Non-determinism* in executions of an FA-GSM Γ may lead to different outcomes for the same input, which complicates a fair comparison among executions of different schemas. There are two ways that non-determinism arises:

Different stage outputs: Since many stages correspond to human activities, the outputs may vary due to a variety of factors that are not explicitly available in the snapshot that launched the stage containing that stage.

Different stage completion timing: Because sentries may include stage completion events, there may be "race" conditions under which a sentry does or does not fire. For example, consider sentry $\psi = \mathsf{C}{:}S \wedge \varphi$. Suppose that in a particular execution ξ stage S completes before all variables in φ have become defined. Then ψ can never be triggered in ξ. In contrast, if S completes after all variables in φ have become defined then ψ might trigger in ξ.

To enable "apples to apples" comparisons of executions of Γ and Γ', we shall use conditions that accomodate these two causes of non-determinism.

The next definition allows us to focus on pairs of executions for which all shared stages have the same behavior. In essence, this enables us to assume that the stages are "deterministic" for a particular comparison. (For this definition, recall that there is no stage nesting, and so we can blur the distinction between a stage and the task that it contains.)

Definition 4.1: Given FA-GSM schema $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{out}, sen, sig)$, a *stage i/o assignment* is a function τ with domain Att_S such that for each $S \in Att_S$, $\tau[S] : sig_{in}(S) \rightarrow sig_{out}(S)$, that is, $\tau[S]$ is a function whose signature matches the signature of S in Γ.

An execution $\xi = \sigma_{init}, \sigma_0, \alpha_1, \beta_1, \ldots, \sigma_n$ of Γ is *compliant* with τ if for each $i \in [1..n]$, if $\beta_i = \mathsf{C}{:}S(c_1, \ldots, c_p)$ then the values $c_1, \ldots, c_p$ correspond to the output of $\tau(S)$ applied to values that σ_{i-1} assigns to the input attributes of S. The set of executions of Γ that are compliant with τ is denoted as $Exec(\Gamma, \tau)$.

Example 4.2: Let IDG denote stage Initial Data Gathering of BCA^{base}, and BPEC denote Business Performance Evaluation Check. In one stage i/o assignment τ_{ABC} for the *ABC* company, we might have

$$\begin{aligned} \tau_{\mathsf{ABC}}[\mathsf{IDG}](\mathsf{employee\_count}) &= 1200 \\ \tau_{\mathsf{ABC}}[\mathsf{IDG}](\mathsf{annual\_revenue}) &= \$\ 500\text{K} \\ \tau_{\mathsf{ABC}}[\mathsf{BPEC}](\mathsf{BP\_good}) &= \mathit{True} \end{aligned}$$

Because the evaluations of business performance may be subjective, a different stage i/o assignment τ'_{ABC} might arise, with $\tau'_{\mathsf{ABC}}[\mathsf{IDG}] = \tau_{\mathsf{ABC}}[\mathsf{IDG}]$ but $\tau'_{\mathsf{ABC}}[\mathsf{BPEC}](\mathsf{BP\_good}) = \mathit{False}$. □

The following result states that if two executions of Γ are compliant with the same stage i/o assignment, and it the order of stage completions is the same, then they are identical in all other ways as well. In other words, the full range of nondeterminism in GSM executions can be controlled by holding the stage behaviors and the relative timing of stage completion fixed.

Lemma 4.3: Let Γ be an FA-GSM schema and τ a stage i/o assignment for Γ. Let $\xi^q = \sigma_0, \alpha_1^q, \beta_1^q, \ldots, \sigma_{n_q}^q$ be a terminal execution in $Exec(\Gamma, \tau)$ for $q \in [1, 2]$. Suppose further that the ordering of stage completions in ξ^1 is identical to the ordering of stage completions in ξ^2. Then $\xi^1 = \xi^2$, and in particular, the values of the final snapshots of ξ^1 and ξ^2 on $out(\Gamma)$ are identical.

Proof: (Sketch) Recall that in GSM, for a snapshot σ and event $\mathsf{C}{:}S(c_1, \ldots, c_n)$, there is at most one result of computing a B-step on σ and $\mathsf{C}{:}S(c_1, \ldots, c_n)$. The result now follows from a straightforward induction. □

As an aside, we note that if all sentries in Γ are eventless, then the above lemma remains true if the condition on orderings of stage completions is dropped.

4.2 Conditional Emulation

In the general case, we shall be looking at a pair Γ^1, Γ^2 of FA-GSM schemas and attempting to compare elements of $TermExec(\Gamma^1)$ with elements of $TermExec(\Gamma^2)$.

We typically focus on executions that satisfy a condition, e.g., in the case of Example 2.2, Ω = "employee count ≥ 300". We then demonstrate that executions of one schema that satisfy the condition can be emulated by executions of the other, e.g., for each execution of BCA^{mod} that satisfies Ω there is a corresponding execution of BCA^{base} that behaves identically on output attributes PCU, DCS, and recommendation (see Example 5.3 below).

In the sequel, if f is a function over domain D, and $C \subseteq D$, then $f|_C$ denotes the restriction of f to C.

Suppose now that $\Gamma^i = (Att^i = Att^i_d \cup Att^i_m \cup Att^i_S, {m_{\text{start}}}^i, Att^i_{out}, sen^i, sig^i)$ for i in [1,2]. Suppose further that τ^i is a stage i/o assignment for Γ^i, i in [1,2]. Then τ^1 and τ^2 are *compatible* if $\tau^1|_{Att^1_S \cap Att^2_S} = \tau^2|_{Att^1_S \cap Att^2_S}$.

Let Γ^1, Γ^2 be as above. As suggested above, we shall work with conditions Ω over the union $Att^1 \cup Att^2$, in order to focus on executions of Γ^1 or Γ^2 of interest. For a snapshot σ^1 over Γ^1, σ^1 *satisfies* Ω with existential extension, denoted $\sigma^1 \models^{ex} \Omega$, if there is some extension σ of σ^1 to include all attributes of Ω not in Att^1, such that $\sigma \models^{strict} \Omega$. (For the current paper we require strict satisfaction of the conditions Ω; variations of this can also be used.)

We now define the notion of "conditional emulatability", which enables us to compare the behavior of pairs of schemas with regards to selected attributes.

Definition 4.4: Let $\Gamma^i = (Att^i = Att^i_d \cup Att^i_m \cup Att^i_S, {m_{\text{start}}}^i, Att^i_{out}, sen^i, sig^i)$ be an FA-GSM schema for i in [1,2], and let $\mathcal{A} \subseteq Att^1 \cap Att^2$, and let Ω be a condition over $Att^1 \cup Att^2$. Then Γ^1 *emulates* Γ^2 under Ω, denoted $\Gamma^1 \rightharpoonup_{\Omega,\mathcal{A}} \Gamma^2$, if the following holds. If

1. τ^2 is a stage i/o assignment for Γ^2;
2. $\xi^2 \in Exec(\Gamma^2)$ is a (possibly non-terminal) τ^2-compliant execution with final snapshot σ^2; and
3. $\sigma^2 \models^{ex} \Omega$

then

1. there exists a stage i/o assignment τ^1 for Γ^1 that is compatible with τ^2, and
2. there exists a τ^1-compliant execution $\xi^1 \in Exec(\Gamma^1)$ with final snapshot σ^1,
3. such that $\sigma^1|_{\mathcal{A}} = \sigma^2|_{\mathcal{A}}$.

We write $\Gamma^1 \rightleftharpoons_{\Omega,\mathcal{A}} \Gamma^2$ if $\Gamma^1 \rightharpoonup_{\Omega,\mathcal{A}} \Gamma^2$ and $\Gamma^2 \rightharpoonup_{\Omega,\mathcal{A}} \Gamma^1$.

Example 4.5: Recall BCA^{base} (Example 2.1) and BCA^{del} (Example 2.3). Let

1. $\mathcal{A}$ = {PCS, PCU}
2. Ω = "Rating = 9"

We illustrate now how it can be shown that $\Gamma^1 \rightleftharpoons_{\Omega,\mathcal{A}} \Gamma^2$. For the $\rightharpoonup$ direction, fix stage i/o assignment τ^2 for Γ^2. We focus here on executions ξ^2 of Γ^2 where IDGS is satisfied. In those cases, the only τ^1 that extends τ^2 and enables satisfaction of Ω will have τ^1[Credit Check](Rating) = 9. For this τ^1, the stage Credit Check will execute and return Rating with value 9 and trigger the milestone CCS. Thus, an execution ξ^1 compliant with τ^1 can be constructed from ξ^2 by inserting the launch and completion of Credit Rating sometime in between the satisfaction of IDGS and satisfaction of CCS. Emulation in the other direction is straightforward to show. □

4.3 The Lifting Lemma

The Lifting Lemma will enable us to infer emulatability in terms of output attributes, i.e., at a "global level", based on emulatability in terms of selected milestone attributes, i.e., at a "local level".

To state the lifting lemma we need to be able to talk about the areas where schemas Γ^1, Γ^2 differ.

Definition 4.6: Let $\Gamma^i = (Att^i = Att^i_d \cup Att^i_m \cup Att^i_S, m_{\text{start}}{}^i, Att^i_{out}, sen^i, sig^i)$ be an FA-GSM schema, let $\Delta^i \subseteq Att^i_m \cup Att^i_S$, for i in [1,2]. Then Δ^1, Δ^2 is a *change pair* for Γ^1, Γ^2 if $Att^1 - \Delta^1 = att^2 - \Delta^2 = \mathcal{A}$ and $sen^1|_{\mathcal{A}} = sen^2|_{\mathcal{A}}$. In this case, both Δ^1 and Δ^2 are called *change sets*.

That is, Δ^1, Δ^2 is a change pair for Γ^1, Γ^2 if the two schemas are identical except for the milestones and stages in the delta's.

Next, we introduce the notion of "fence" that allows us to create a separation between a change set and an output attribute.

Definition 4.7: Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{out}, sen, sig)$ be an FA-GSM schema, let $\Delta \subset Att_m \cup Att_S$, and let $\mathcal{O} \subseteq Att_{out}$. A set $\mathcal{F} \subseteq Att_m$ is a *fence* between Δ and $\mathcal{O}$ if for each pair $\delta \in \Delta, o \in \mathcal{O}$ and each path ρ from δ to o in $DG(\Gamma)$ there is some $m \in \mathcal{M}$ on path ρ.

Speaking intuitively, if $\mathcal{F}$ is a fence between Δ and $\mathcal{O}$, and if certain "race" conditions do not hold, then the values assigned to $\mathcal{O}$ will not be impacted by the behavior in the Δ area. In this sense, the fence $\mathcal{F}$ "protects" the set $\mathcal{O}$ of output attributes from the set Δ. The next definition identifies the "race" conditions that need to be avoided (see Example 4.9 below).

Definition 4.8: Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{out}, sen, sig)$ be an FA-GSM schema, $\mathcal{F} \subseteq Att_m$ a set of milestones in Γ, and $v \in Att_m \cup Att_S$. Then v is *completion independent* modulo $\mathcal{F}$ if for each stage $S \in Att_S$ and each path ρ from S to v, if there is a node w on ρ with a sentry of form "C:$S \ldots$", then there is a node $f \in \mathcal{F}$ that lies between w and v in ρ.

We note that since $w \neq f$, w has the effect of transforming the C:S into an S, i.e., from time-specific to persisting.

Example 4.9: In BCA^{base}, with the exception of bonus and TBPS, all output attributes are completion independent modulo $\{PCS\}$. In contrast, bonus and TBPS are not, because of the completion event C:Fast Turnaround Bonus Eligibility in the guard for stage Team Bonus Pay. □

We now have the Lifting Lemma, which states that under certain conditions, if Γ^1 emulates Γ^2 for the elements of a fence, then Γ^1 also emulates Γ^2 for output attributes that are downstream from that fence. The proof, omitted, is based on splicing of executions.

Lemma 4.10: (Lifting Lemma) Let $\Gamma^i = (Att^i = Att^i_d \cup Att^i_m \cup Att^i_S, m_{\text{start}}{}^i, Att^i_{out}, sen^i, sig^i)$ be an FA-GSM schema for i in [1,2]. Suppose that:

1. Δ^1, Δ^2 is a change pair for Γ^1, Γ^2.
2. $\mathcal{O} \subseteq out(\Gamma^1) \cap out(\Gamma^2)$.
3. $\mathcal{F}$ is a fence between Δ^i and $\mathcal{O}$ in Γ^i for i in [1,2].
4. $\mathcal{O}$ is completion independent modulo $\mathcal{F}$ in Γ^i, for i in [1,2].
5. Ω is a condition over $Att^1 \cup Att^2$.
6. $\Gamma^1 \rightharpoonup_{\Omega,\mathcal{F}} \Gamma^2$.

Then $\Gamma^1 \rightharpoonup_{\Omega,\mathcal{O}} \Gamma^2$.

We next apply the Lifting Lemma to the example of deletion from Section 2.

Example 4.11: Recall Example 4.5, and the property $\mathsf{BCA}^{base} \rightleftharpoons_{\Omega,\mathcal{A}} \mathsf{BCA}^{del}$, where $\mathcal{F} = \{\mathsf{PCS}, \mathsf{PCU}\}$ and Ω = "rating = 9". Let Δ^1 = {Credit Check, CCS, CCU, PCS, PCU} and Δ^2 = {PCS, PCU}. Then Δ^1, Δ^2 is a change pair for Γ^1, Γ^2. It is straightforward to verify that $\mathcal{F}$ is a fence between these change sets and the output attributes $\mathcal{O}$ = {IDGU, PCU, recommendation, DCS, DCU}. Thus, by the Lifting Lemma, $\Gamma^1 \rightleftharpoons_{\Omega,\mathcal{O}} \Gamma^2$. Intuitively, this states that Γ^1, Γ^2 have identical behavior on $\mathcal{O}$, if the rating attribute is assumed to have value 9. There are no guarantees with regards to the attribute bonus), because of a possible race condition involving the completion of Fast Turnaround Bonus Eligibility, which occurs in the sentry for Team Bonus Pay.

However, note that bonus and TBPS have a completion dependency on Fast Turnaround Bonus Elibibility that is not blocked by $\mathcal{F}$. As a result, the Lifting Lemma does not apply to those attributes. Indeed, it is possible to construct an example execution ξ^1 of Γ^1 where Team Bonus Pay is not launched, but in the corresponding execution ξ^2 of Γ^2 this stage would launch. Intuitively, this can happen if in ξ^1, Business Performance Evaluation Check completes before Fast Turnaround Bonus Eligibility, and Credit Check completes after Fast Turnaround Bonus Eligibility. In particular, in ξ^1, PCS becomes true only after Fast Turnaround Bonus Eligibility, and so the guard for Team Bonus Pay will never go true. (With this particular example, an alternative $\xi^{2'}$ can be constructed to achieve an emulation, but in the general case there might be other stages whose launching would depend on the timing of when Business Performance Evaluation Check completes.)

Note that if the completion event C:Fast Turnaround Bonus Eligibility in the guard for Team Bonus Pay were dropped, then Term Bonus Pay, TBPS, and bonus would be completion independent modulo $\mathcal{F}$, and so the Lifting Lemma would apply to them. □

5 Property Preserving Schema Modifications

This section presents operators for modifying FA-GSM schemas that guarantee the preservation of various properties. The operators focus on sentry modification, and on deletions and insertions of stages and milestones. The proofs about property preservation rely on the Lifting Lemma. Examples from Section 2 are used to illustrate the results developed here.

We begin with a useful observation that is a very straightforward consequence of the Lifting Lemma. Before making the observation, we need the following: the notion of "shadow" of a change set Δ. Intuitively, the shadow is the set of milestones, stages, and data attributes that are "downstream" of nodes in Δ in the graph $DG(\Gamma)$.

Definition 5.1: Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{out}, sen, sig)$ be an FA-GSM schema and $\Delta \subseteq Att_m \cup Att_S$. The *shadow* of Δ in Γ, denoted $shadow(\Delta, \Gamma)$ is $\{v \in Att_m \cup Att_S \mid \exists \delta \in \Delta$ and a path in $DG(\Gamma)$ from δ to $v\}$ $\cup$ $a \in sig_{out}(S) | S \in Att_S$ and $\exists \delta \in \Delta$ and a path in $DG(\Gamma)$ from δ to $S\}$.

Let Δ^1, Δ^2 be a change pair for FA-GSM schemas Γ^1, Γ^2. It is easily shown that $shadow(\Delta^1, \Gamma^1) = shadow(\Delta^2, \Gamma^2)$.

Proposition 5.2: Let $\Gamma^i = (Att^i = Att^i_d \cup Att^i_m \cup Att^i_S, {m_{\text{start}}}^i, Att^i_{out}, sen^i, sig^i)$ for i in [1,2], and let Δ^1, Δ^2 be a change pair for Γ^1, Γ^2. Let $\mathcal{A} = shadow(\Delta^1, \Gamma^1) = shadow(\Delta^2, \Gamma^2)$, and let $\mathcal{O} = (Att^1_{out} \cup Att^2_{out}) - \mathcal{A}$. Then $\Gamma^1 \rightleftharpoons_{True,\mathcal{O}} \Gamma^2$.

Proof: (Sketch) To apply the Lifting Lemma in this case, choose the fence $\mathcal{F}$ to be $\emptyset$. Each node $o \in \mathcal{O}$ satisfies the conditions concerning paths from Δ^i to o, because there are no such paths. □

We next examine a simple form of sentry modification.

Example 5.3: Consider BCA^{base} from Example 2.1 and BCA^{mod} from Example 2.2. Recall that BCA^{mod} is formed from BCA^{base} by modifying the sentry on Business Performance Evaluation Check, to skip launching of that stage if the client has < 300 employees, and adding a sentry for milestone PCS. Let Ω = "employee_count $\geq$ 300". A case-by-case argument can be used to show that $\mathsf{BCA}^{base} \rightleftharpoons_{\Omega,\{\mathsf{PCS},\mathsf{PCU}\}} \mathsf{BCA}^{mod}$. Now let

- $\mathcal{O}$ = {IDGU, PCU, recommendation, DCS, DCU}.
- Δ^1 = {Business Performance Evaluation Check, BPECS, BPECU, PCS}.
- Δ^2 = {PCS}.

Similar to Example 4.11, it is easily verified that $\mathcal{F}$ = {PCS, PCU} is a fence for Δ^i and $\mathcal{O}$, for i in [1,2]. Further, $\mathcal{O}$ is completion-independent modulo $\mathcal{F}$. The Lifting Lemma now implies that $\Gamma^1 \rightleftharpoons_{\Omega,\mathcal{O}} \Gamma^2$. We comment now on a possible extension of the Lifting Lemma that could be used to compare the behavior of BCA^{base} and BCA^{mod} on companies with < 300 employees. Assume here that Business Performance Evaluation Check produces one Boolean data attribute BP_good, which is assigned *True* if the stage returns a positive evaluation, and is assigned *False* otherwise. In essence, the construction of BCA^{mod} is implicitly assuming that all such companies will have BP_good set to *True*. One way to capture this is to extend the condition language to include properties of stage i/o assignments, and set Ω' = "employee_count $\geq 300 \wedge T$[Business Performance Evaluation Check](BP_good) = *True*", where T is a variable that ranges over stage i/o assignments. □

We consider the challenge of generalizing the above example, to find a general-purpose approach for modifying a schema to (a) reflect changes to the sentries of one node, and (b) preserve behavior as much as possible? Several aspects make the preceding example "easy" to work with: (i) only one sentry is modified, (ii) the modification involves adding a condition using conjunction, (iii) the new condition is based on a data attribute whose defining stage precedes the affected sentry in $DG(\mathsf{BCA}^{base})$. Although not done here, a generalization of the approach can be developed, that permits adding conditions (with conjunction) to multiple sentries of a stage or milestone. In this case, identifying the preserved properties can be accomplished inductively, based on modifying one sentry at a time.

5.1 Deletion

This subsection develops constructions for deleting milestones and stages from FA-GSM schemas. Similar to the examples of Section 2, the focus is on enabling the deletions while maximizing emulatability.

We begin by describing the construction for deleting a single milestone. We shall use two notational conventions. The first is for substitutions in sentries: given a sentry ψ, an attribute z, and a formula φ, $\psi[z/\varphi]$ denotes the result of replacing all occurrences of z in ψ by (φ). The second is a manipulation on sentries called *completion-event removal*: For a sentry of form $\psi = \mathsf{C}{:}S \wedge \varphi$, define $cer(\psi)$ to be $S \wedge \varphi$. Notice that ψ will be true for the single B-step where stage S completes, whereas $cer(\psi)$ will be true for that B-step and all subsequent B-steps. If ψ is eventless, then $cer(\psi) = \psi$.

The following definition specifies implicitly an algorithm for deleting a milestone while preserving all output behaviors.

Definition 5.4: Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{out}, sen, sig)$ be an FA-GSM schema, m a milestone of Γ, and $M = \{\psi_1, \ldots, \psi_q\}$ the set of sentries of m in Γ. The *deletion* of m from Γ, denoted $del(\Gamma, m)$, is the FA-GSM schema constructed from Γ in the following way. Suppose that v is a stage or milestone in Γ, that χ is a sentry for v, and that m occurs in χ. Then replace χ in Γ with a set of sentries

$$N = \{\chi[m/cer(\psi_p)] \mid p \in [1, q]\}$$

Finally, delete m from the set of attributes of Γ.

Intuitively, in the construction of schema $del(\Gamma, m)$ occurrences of m in sentries are replaced by “macro-expansions” of m. It is straightforward to verify that the result of the construction is an FA-GSM schema.

(Completion event removal is performed to handle situations where the target sentry itself has a completion event.)

Lemma 5.5: Let Γ be an FA-GSM schema and m a milestone of Γ^1. Let $\mathcal{F}$ be the set of stages and milestones of Γ whose sentries are modified to create $del(\Gamma, m)$. Then $\Gamma \rightleftharpoons_{True,\mathcal{F}} del(\Gamma, m)$.

Proof: (Sketch) One aspect of the proof is to show that for a sentry ψ, $cer(\psi)$ takes the value *True* for all snapshots after ψ has triggered, and also that the behavior of $cer(\psi)$ mimics ψ, in terms of triggering behavior. The other aspect involves showing how executions of Γ can be transformed into executions of $del(\Gamma, m)$ with the same behaviors on $\mathcal{F}$. □

Deleting a stage S from an FA-GSM schema $\mathit{\Gamma}$ is similar to deleting a milestone, in terms of performing "macro-expansions" in selected sentries. However, there are three complications. First, something must be done about the values of the data attributes produced by S. In the approach taken here, we assume that a vector $\overrightarrow{c}$ of constants is used to serve as default values. Second, speaking intuitively, we must ensure that the attribute values in $\overrightarrow{c}$ are not available for use until after S would have completed; the approach taken here follows the pattern used for deleting milestones. And third, we must address sentries χ that have form $\mathsf{C}{:}S \wedge \varphi$; for these we essentially replace $\mathsf{C}{:}S$ with the sentries that launch S.

Definition 5.6: Let $\mathit{\Gamma} = (\mathit{Att} = \mathit{Att}_d \cup \mathit{Att}_m \cup \mathit{Att}_S, m_{\mathrm{start}}, \mathit{Att}_{out}, \mathit{sen}, \mathit{sig})$ be an FA-GSM schema, S a stage of $\mathit{\Gamma}$, and $M = \{\psi_1, \ldots, \psi_q\}$ the set of sentries of m in $\mathit{\Gamma}$. Let $\overrightarrow{a} = \mathit{sig}_{out}(S)$ and let $\overrightarrow{c}$ be a vector of constants having types that match $\overrightarrow{a}$. The *deletion* of S from $\mathit{\Gamma}$ using $\overrightarrow{c}$ for $\overrightarrow{a}$, denoted $\mathit{del}(\mathit{\Gamma}, S, \overrightarrow{a}/\overrightarrow{c})$, is an FA-GSM schema constructed from $\mathit{\Gamma}$ in the following way. Suppose that v is a stage or milestone in $\mathit{\Gamma}$, that χ is a sentry for v, and that χ includes $\mathsf{C}{:}S$ and/or includes one or more attribute from $\overrightarrow{a}$. Then replace χ with a set of sentries

$$N = \{\chi[\mathsf{C}{:}S/\psi_p, \overrightarrow{a}/\overrightarrow{c}] \wedge \mathit{cer}(\psi_p) \mid p \in [1, q]\}.$$

Finally, delete S from Att_S.

Example 5.7: To illustrate the above construction, consider a variation BCA^{del}_{var} of BCA^{del}, in which only the stage Credit Check is deleted, but milestones CCS and CCU are to be retained. In this case, the sentry of CCS will become "IDGS $\wedge\ 9 \geq 8$", and the sentry of CCU will become "IDGS $\wedge 9 < 8$". $\Box$

The following lemma establishes the key property preservation properties of the stage deletion construction. (The proof is similar to that of the preceding lemma, and is omitted.)

Lemma 5.8: Let $\mathit{\Gamma} = (\mathit{Att} = \mathit{Att}_d \cup \mathit{Att}_m \cup \mathit{Att}_S, m_{\mathrm{start}}, \mathit{Att}_{out}, \mathit{sen}, \mathit{sig})$ be an FA-GSM schema, S a stage of $\mathit{\Gamma}$, and $\overrightarrow{c}$ a vector of constants having the types of $\overrightarrow{a} = \mathit{sig}_{out}(S)$. Let $\mathcal{F}$ be the set of stages and milestones whose sentries are modified in $\mathit{del}(\mathit{\Gamma}, S, \overrightarrow{c})$. Let $\mathit{\Omega}$ be "$\overrightarrow{a} = \overrightarrow{c}$. Then $\mathit{\Gamma} \rightleftharpoons_{\mathit{\Omega},\mathcal{F}} \mathit{del}(\mathit{\Gamma}, m)$.

The preceding lemmas are now generalized to provide conditions on property preservation when a full fragment is deleted from an FA-GSM schema. Suppose that $\mathit{\Gamma} = (\mathit{Att} = \mathit{Att}_d \cup \mathit{Att}_m \cup \mathit{Att}_S, m_{\mathrm{start}}, \mathit{Att}_{out}, \mathit{sen}, \mathit{sig})$ is an FA-GSM schema, and let $\mathcal{X}$ be a set of stages and/or milestones in $\mathit{\Gamma}$. For each stage S in $\mathcal{X}$, let $\overrightarrow{a}^S = \mathit{sig}_{out}(S)$ and let $\overrightarrow{c}^S$ be a set of constants with the same types. Let $\overrightarrow{a}$ denote a listing of all $\overrightarrow{a}^S$ for stages $S \in \mathcal{X}$, and define $\overrightarrow{c}$ similarly.

Now let $x_1, \ldots, x_n$ be a listing of the elements of $\mathcal{X}$. Let $\mathit{\Gamma}' = \mathit{del}(x_1, \mathit{del}(x_2, \ldots, \mathit{del}(x_n, \mathit{\Gamma}, \overrightarrow{a}^{x_n}/\overrightarrow{c}^{x_n}), \ldots, \overrightarrow{a}^{x_2}/\overrightarrow{c}^{x_2}), \overrightarrow{a}^{x_1}/\overrightarrow{c}^{x_1})$, where $\overrightarrow{a}^{x_i}/\overrightarrow{c}^{x_i}$ is empty if x_i is a milestone. It can be shown that the deletion operation satisfies a Church-Rosser property, and so different orderings for $\mathcal{X}$ will yield equivalent FA-GSM schemas. We define the *deletion* of $\mathcal{X}$ from $\mathit{\Gamma}$, denoted $\mathit{del}(\mathit{\Gamma}, \mathcal{X}, \overrightarrow{a}/\overrightarrow{c})$, to be one of these equivalent schemas. The following result can be shown using an induction based on Lemmas 5.5 and 5.8.

Theorem 5.9: Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{out}, sen, sig)$, $\mathcal{X}$, $\overrightarrow{d}$ and $\overrightarrow{c}$ and $\Gamma' = del(\Gamma, \mathcal{X}, \overrightarrow{d}/\overrightarrow{c})$ be as above. Let $\mathcal{F}$ be the collection of all stages and milestones in Γ' whose sentries have been modified, and let Ω be the formula $\overrightarrow{d} = \overrightarrow{c}$. Then $\Gamma \rightleftharpoons_{\Omega,\mathcal{F}} \Gamma'$. Furthermore, if $\mathcal{O} \subseteq Att_{out}$ is completion-independent modulo $\mathcal{F}$, then $\Gamma \rightleftharpoons_{\Omega,\mathcal{O}} \Gamma'$.

5.2 Insertion

This subsection studies property preservation in the context of insertions to an FA-GSM schema Γ. Speaking intuitively, the emphasis here is on enabling the designer to insert one or several stages and milestones, while ensuring that the global impact of the insertion is minimized "when things go right". This approach was followed in the construction of schema BCA^{ins} from BCA^{base} in Section 2: for each execution of BCA^{ins} where milestone AMCS goes true there is guaranteed to be a corresponding execution of BCA^{base} that produces the same outcome.

The following definition is provided to talk about "bulk" insertions.

Definition 5.10: Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{out}, sen, sig)$ be an FA-GSM schema. An *insertable fragment* for Γ is a tuple $\Delta = (Att^\Delta = Att_d^\Delta \cup Att_m^\Delta \cup Att_S^\Delta, Att_{out}^\Delta, sen^\Delta, sig^\Delta)$ that satisfies the following conditions:

1. Att_d^Δ is a set of data attributes names, disjoint from Att_d.
2. Att_m^Δ is a set of milestone names, disjoint from Att_m.
3. Att_S^Δ is a set of stage names, disjoint from Att_S.
4. $Att_{out}^\Delta \subseteq Att_m^D elta \cup att_d^\Delta$.
5. sen^Δ is a mapping from $Att_m^\Delta \cup Att_S^\Delta$ into sets of sentries, where each sentry can refer to elements of $Att \cup Att^\Delta$ and completions of elements of $Att_S \cup Att_S^\Delta$.
6. sig^Δ is a mapping from stages in Att_S^Δ, such that for each $S \in Att_S^\Delta$,
   (a) $sig_{in}^\Delta(S)$ and $sig_{out}^\Delta(S)$ are finite sets of attribute names.
   (b) $sig_{in}^\Delta(S) \subseteq Att_d \cup (\cup\{sig_{out}^\Delta(S') \mid S' \neq S, S \in Att_S^\Delta\})$
   (c) $sig_{out}^\Delta(S) \cap Att_d = \emptyset$, i.e., the stages in Δ produce all new data attributes.
7. The dependency graph produced by merging Γ and Δ is acyclic.

The *insertion* of Δ into Γ, denoted $ins(\Gamma, \Delta)$ is the tuple $(Att' = (Att_d \cup Att_d^\Delta) \cup (Att_m \cup Att_m^\Delta) \cup (Att_S \cup Att_S^\Delta), m_{\text{start}}, Att_{out} \cup Att_{out}^\Delta, sen \cup sen^\Delta, sig \cup sig^\Delta)$

Given Γ, Δ as above, it is straightforward to verify that $ins(\Gamma, \Delta)$ is an FA-GSM schema.

To enable modular insertions, and to facilitate straightforward reasoning about the impact of an insertion, a best practice is to include as part of Δ one or more milestones that are used to indicate the "success" or "failure" of a case with regards to the inserted activity. (More refined kinds of milestones can also be imagined). The following result assumes there is a single "success" milestone $m_{success}$ in Δ; generalizations are left to the reader. The result follows easily from the Lifting Lemma.

Theorem 5.11: Let $\Gamma = (Att = Att_d \cup Att_m \cup Att_S, m_{\text{start}}, Att_{out}, sen, sig)$ be an FA-GSM schema, and let $\Delta = (Att^{\Delta} = Att^{\Delta}_d \cup Att^{\Delta}_m \cup Att^{\Delta}_S, Att^{\Delta}_{out}, sen^{\Delta}, sig^{\Delta})$ be an insertable fragment that includes a milestone $m_{success}$. Suppose that $\mathcal{F} \subseteq Att_m$ is a family of milestones in Γ, and suppose that Γ' is the result of modifying $ins(\Gamma, \Delta)$ by replacing each sentry μ of a milestone in $\mathcal{F}$ by $\mu \wedge m_{success}$. Let Ω = "$m_{success}$". Finally, let $\mathcal{O} \subseteq Att_{out}$ be completion-independent modulo $\mathcal{F}$ in Γ'. Then $\Gamma \rightharpoonup_{\Omega,\mathcal{O}} \Gamma'$.

Example 5.12: In the schema BCA^{ins} of Example 2.4, with regards to the above theorem, the milestone AMCS plays the role of $m_{success}$, the set {PCS} plays the role of $\mathcal{F}$, and the set {recommendation, DCS, DCU} plays the rols of $\mathcal{O}$. In this case, the theorem tells us that for each execution of BCA^{ins} for which AMCS goes true, there is a corresponding execution of the base schema BCA^{base} with the same outcomes on $\mathcal{O}$. What about failure of Addressable Market Check? In this case a variant of Theorem 5.11 can be formulated, based on a milestone m_{fail}. In the example, AMCU would play role of m_{fail}, {PCU} would play the roles of both $\mathcal{F}$ and $\mathcal{O}$. □

6 Related work

We discuss the literature on changes in process models for activity-centric business process management and case management.

In the context of *activity-centric BPM*, change operations have been proposed [29]. Different correctness criteria have been identified in the literature to assess which changes are allowed so that cases can be migrated properly from an old to a new schema [26]. A particular focus has been on ensuring that when the execution of a BP instance starts on one schema and migrates to another one while in flight, the final BP instance corresponds to an execution of the new schema. In our approach, we study a novel form of correctness, which focuses on preservation of schema properties, defined in terms of emulatability of one shema by another one. A form of unconditional emulatability was studied in connection with declarative artifact-centric business processes in [4]. That work was in an abstract setting; in contrast the results here are tied to a practical Case Management model, and motivated by a real-world use case.

Applying changes to activity-centric process models leads to variety of related process models that needs to be managed properly [10–12, 27]. Configurable workflow models [10, 27] manage adaptations for activity-centric process models. Configurable workflow models contain configurable elements that can be skipped or blocked. This way, different workflow models can be generated from the same configurable workflow model.

Business process families relate feature models, introduced for managing variability in software product lines, to business process models [11]. There the main focus is on finding inconsistencies between selected features and the generated process model realizing those features.

The Provop (Process Variants by Options) approach uses groups of atomic change operations, called options, to generate different process models from a base process model [12]. Different strategies for defining base models are discussed.

Case management originates from industry, including, e.g., [28] and work on business artifacts, e.g., [24]. Recent overview works include [8, 15, 22]. Case management is related to the more general concept of data-centric business process management, which studies how activity-centric processes can be made more data-aware [24, 18, 25, 20] to improve their flexibility. This includes work on declarative artifact-centric models, including GSM [5, 6] and declarative process models for case management [14].

Though the problem of change has been recognized as central to case management [15], in particular adaptive case management [21], it has not been widely studied. Mukkalama et al. [23] study change in DCR Graphs, a declarative formalism for case management. They define basic change operations that add and remove behavior, but their operations are aimed at a micro-level, so removing atomic elements from schema. In our approach, we study also the impact of adding and removing larger fragments, so at a macro level. They focus on logical correctness and the use of automated verification techniques, whereas we develop tests for property preservation that can be checked at a syntactic level.

Motahari et al. [21] present a framework and prototype implementation that supports adaptive case management in social enterprises. The framework supports change, but does not address preservation of properties across changes.

There has been active research on verification for artifact-centric BPM models (e.g., [2, 7, 1, 13]). That work could be also be applied to reason about preservation of properties of case management schemas during evolution. The approach in the current paper uses syntactic conditions rather than semantic ones, and would thus be subsantially easier easier to deploy and maintain than a verification-based approach.

7 Conclusion

This paper studies schema modifications in the context of a varient of the Guard-Stage-Milestone (GSM) model for Case Management. The main contributions of this paper are (i) a precise definition for testing the preservation of properties through the use of conditional emulatability; (ii) the development of a general-purpose "Lifting Lemma" which allows a variety of approaches to achieve and/or prove property preservation; and (iii) the specification of operators to perform schema manipulations that are guaranteed to preserve certain properties. The theoretical work is motivated by examples arising in a real-world application.

The research here can be extended in several directions, including the following: (a) extend results to more general kinds of GSM schema; (b) extend results to other Case Management and BPM models ([28] is a natural first candidate, and also the OMG CMMN standard [3]); (c) develop algorithms for schema modifications other than deletion and insertion, that preserve specified properties; (d) generalize to support adaptation of schemas for cases that are "in-flight". and (e) develop approaches to apply the theoretical results developed here in practical settings.

References

1. F. Belardinelli, A. Lomuscio, and F. Patrizi. Verification of GSM-based artifact-centric systems through finite abstraction. In *Proc. Intl. Conf. on Service-Oriented Computing (ICSOC)*, pages 17–31, 2012.
2. K. Bhattacharya, C. E. Gerede, R. Hull, R. Liu, and J. Su. Towards formal analysis of artifact-centric business process models. In *Proc. Int. Conf. on Business Process Management (BPM)*, pages 288–304, 2007.
3. BizAgi and others. Case Management Model and Notation (CMMN), v1, May 2014. OMG Document Number formal/2014-05-05, Object Management Group.
4. D. Calvanese, G. D. Giacomo, R. Hull, and J. Su. Artifact-centric workflow dominance. In *Proc. Intl. Conf. on Service Oriented Computing (ICSOC)*, 2009.
5. D. Cohn and R. Hull. Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes. *IEEE Data Eng. Bull.*, 32(3):3–9, 2009.
6. E. Damaggio, R. Hull, and R. Vaculín. On the equivalence of incremental and fixpoint semantics for business artifacts with guard-stage-milestone lifecycles. *Information Systems*, 38:561–584, 2013.
7. A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic verification of data-centric business processes. In *Proc. Intl. Conf. on Database Theory (ICDT)*, 2009.
8. C. Di Ciccio, A. Marrella, and A. Russo. Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches. *J. Data Semantics*, 4(1):29–57, 2015.
9. R. Eshuis, R. Hull, Y. Sun, and R. Vaculín. Splitting GSM schemas: A framework for outsourcing of declarative artifact systems. *Inf. Syst.*, 46:157–187, 2014.
10. F. Gottschalk, W. M. P. van der Aalst, M. H. Jansen-Vullers, and M. L. Rosa. Configurable workflow models. *Int. J. Cooperative Inf. Syst.*, 17(2):177–221, 2008.
11. G. Gröner, M. Boskovic, F. S. Parreiras, and D. Gasevic. Modeling and validation of business process families. *Inf. Syst.*, 38(5):709–726, 2013.
12. A. Hallerbach, T. Bauer, and M. Reichert. Capturing variability in business process models: the provop approach. *Journal of Software Maintenance*, 22(6-7):519–546, 2010.
13. B. B. Hariri, D. Calvanese, G. D. Giacomo, A. Deutsch, and M. Montali. Verification of relational data-centric dynamic systems with external services. In *Proc. Intl. Symp. Principles of Database Systems*, pages 163–174, 2013.
14. T. T. Hildebrandt, R. R. Mukkamala, and T. Slaats. Designing a cross-organizational case management system using dynamic condition response graphs. In *Proc. IEEE EDOC 2011*, pages 161–170. IEEE Computer Society, 2011.
15. S. Huber, A. Hauptmann, M. Lederer, and M. Kurz. Managing complexity in adaptive case management. In *Proc. S-BPM ONE 2013*, pages 209–226, 2013.
16. R. Hull, E. Damaggio, R. D. Masellis, F. Fournier, M. Gupta, F. H. III, S. Hobson, M. Linehan, S. Maradugu, A. Nigam, P. Sukaviriya, and R. Vaculín. Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In *Proc. of the 5th ACM Int. Conf. on Distributed Event-Based Systems, DEBS, USA*, pages 51–62, 2011.
17. R. Hull, N. C. Narendra, and A. Nigam. Facilitating workflow interoperation using artifact-centric hubs. In *ICSOC/ServiceWave*, pages 1–18, 2009.
18. V. Künzle and M. Reichert. Philharmonicflows: towards a framework for object-aware process management. *Journal of Software Maintenance*, 23(4):205–244, 2011.
19. L. Limonad, D. Boaz, R. Hull, R. Vaculín, and F. T. Heath. A generic business artifacts based authorization framework for cross-enterprise collaboration. In *SRII Global Conference*, pages 70–79, 2012.
20. A. Meyer, L. Pufahl, D. Fahland, and M. Weske. Modeling and enacting complex data dependencies in business processes. In *Proc. BPM 2013*, pages 171–186, 2013.

21. H. R. Motahari Nezhad, C. Bartolini, S. Graupner, and S. Spence. Adaptive case management in the social enterprise. In *Proc. ICSOC 2012*, pages 550–557, 2012.
22. H. R. Motahari Nezhad and K. D. Swenson. Adaptive case management: Overview and research challenges. In *IEEE CBI 2013*, pages 264–269. IEEE, 2013.
23. R. R. Mukkamala, T. T. Hildebrandt, and T. Slaats. Towards trustworthy adaptive case management with dynamic condition response graphs. In *Proc. EDOC 2013*, pages 127–136, 2013.
24. A. Nigam and N. S. Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3):428–445, 2003.
25. G. Redding, M. Dumas, A. H. M. ter Hofstede, and A. Iordachescu. A flexible, object-centric approach for business process modelling. *Service Oriented Computing and Applications*, 4(3):191–201, 2010.
26. S. Rinderle, M. Reichert, and P. Dadam. Correctness criteria for dynamic changes in workflow systems - a survey. *Data Knowl. Eng.*, 50(1):9–34, 2004.
27. M. Rosemann and W. M. P. van der Aalst. A configurable reference modelling language. *Inf. Syst.*, 32(1):1–23, 2007.
28. W. M. P. van der Aalst, M. Weske, and D. Grünbauer. Case handling: a new paradigm for business process support. *Data Knowl. Eng.*, 53(2):129–162, 2005.
29. B. Weber, M. Reichert, and S. Rinderle-Ma. Change patterns and change support features - enhancing flexibility in process-aware information systems. *Data Knowl. Eng.*, 66(3):438–466, 2008.