Specifying and Validating Data-Aware Temporal Web Service Properties

Sylvain Hallé[†], *Member, IEEE,* Roger Villemaire[‡], *Affiliate Member, IEEE*, Omar Cherkaoui[‡], *Member, IEEE*

***Abstract*—Most works that extend workflow validation beyond syntactical checking consider constraints on the sequence of messages exchanged between services. These constraints are expressed only in terms of message names and abstract away their actual data content. We provide examples of real-world "data-aware" web service constraints where the sequence of messages and their content are interdependent. To this end, we present CTL-FO$^+$, an extension over Computation Tree Logic that includes first-order quantification on message content in addition to temporal operators. We show how CTL-FO$^+$ is adequate for expressing data-aware constraints, give a sound and complete model checking algorithm for CTL-FO$^+$ and establish its complexity to be PSPACE-complete. A "naïve" translation of CTL-FO$^+$ into CTL leads to a serious exponential blow-up of the problem that prevents existing validation tools to be used. We provide an alternate translation of CTL-FO$^+$ into CTL where the construction of the workflow model depends on the property to validate. We show experimentally how this translation is significantly more efficient for complex formulæ and makes model checking of data-aware temporal properties on real-world web service workflows tractable using off-the-shelf tools.**

***Index Terms*—Web services, software/program verification, model checking, temporal logic**

I. Introduction

There exists a large number of web service orchestration tools available over the Internet. Since the format of all input and output messages is publicized by service providers, these tools allow a syntactical validation of the service invocations in a workflow. This "first generation" of web service technologies concentrates on single request-response patterns of messages. However, correctness at the syntactical level does not give a complete picture of the necessary conditions for a successful composition [1]. Nothing prevents a service from sending to a peer syntactically valid messages in a sequence that prevents an actual composition from taking place. *Operating guidelines*, *conversation specification*, *user contract*, *protocol of interaction*, *web service choreography* are various terminologies referring to a common, twofold concern: the use of a formal language to express and advertise the *protocol* imposed on the use of a service, and the development of a methodology to ensure compliance.

A wide consensus exists to the effect that specification of these constraints is beyond the expressive power of existing standards and available design tools. A "second generation" of web service technologies has given rise to a variety of standards taking into account the sequence of message exchanges allowed by a service. The SOAP Service Description Language (SSDL) [2] is a notable example of this approach. Classical temporal languages such as the Linear Temporal Logic (LTL), the Computation Tree Logic (CTL) or the π-calculus have been suggested as appropriate notations for expressing sequential dependencies between messages.

In Section II, we briefly review related work and show why solutions based on traditional temporal logics are not adequate for the validation task at hand: most efforts still treat messages as atomic units represented by their names; they are not "data-aware". In this paper, we argue that "data-awareness" of protocol specifications is a fundamental part of ensuring workflow correctness. We provide in Section III examples of real-world web service scenarios where both the sequence of messages and their content are interdependent.

In Section IV, we present an extension of the popular Computation Tree Logic (CTL) that introduces first-order quantification on values of message elements, called CTL-FO$^+$, as an appropriate formal language for the expression of temporal constraints on web service invocations. Contrarily to the classical temporal formalisms used in most web service validation approaches, CTL-FO$^+$ retains the full temporal flexibility of CTL, while allowing to refer to the content of messages inside the temporal properties. We provide in Section V an algorithm for the model checking of CTL-FO$^+$ formulæ on a given workflow model and show it is PSPACE-complete. This result places CTL-FO$^+$ on a par, complexity-wise, with the Linear Temporal Logic (LTL) used by widely accepted tools like SPIN [3].

An *explicit* translation of CTL-FO$^+$ back into CTL model checking consists in repeating a formula such as $\mathbf{AG}\,(a = x \rightarrow \mathbf{AF}\,b = y)$ for every possible combination of static values of x and y. Any such transformation results in an exponential blow-up of the original problem. VI, we present a reduction of CTL-FO$^+$ to CTL that modifies the translation of a workflow into a finite-state system using the concept of "freeze quantification": the construction of the service model becomes dependent on the property to validate. In Section VII, we compare this freeze quantification approach with the more straightforward *explicit quantification* suggested above. Although both translations are ultimately exponential, we empirically demonstrate that freeze quantification is several orders of magnitude more efficient. We illustrate our claim by using an off-the-shelf tool, the NuSMV [4] model checker, to validate constraints on sample web service workflows. We conclude that our methodology brings validation of data-aware properties within reach of existing tools.

[†]University of California, Santa Barbara, CA 93106-5110; e-mail: shalle@acm.org. [‡]Université du Québec à Montréal, C.P. 8888, Succ. Centre-Ville, Montréal, Canada H3C 3P8; e-mail: villemaire.roger@uqam.ca, cherkaoui.omar@uqam.ca.

II. Related Work and Existing Solutions

Existing web service validation approaches can be classified into three categories, corresponding to the degree of data-awareness they exhibit. We illustrate each of these categories in the simple example of Figure 1. We consider a web service workflow which receives from some partner a message labelled "a" that contains an integer value. If this received value is 0, the service returns a message "b" with value 9. If the received value is not 0, the service returns a message "c" that increments the received value by 1. The $\gg$ symbol means "the next message".

A. Propositional Workflows, Propositional Properties

A first step is to use classical automata-theoretic constructions or model checking tools and languages to model the behaviour of a web service and its interaction with other services. This is exemplified in Figure 1a. The messages are considered *atomic*: their actual data content is abstracted away. We call such a model *propositional*, since the external behaviour of web services is represented by the transmission or reception of messages that are identified by propositional letters standing for their names.

This entails that the choice between sending message "b" and message "c", since it depends on message content, is seen as non-deterministic by the model. For the same reason, the behavioural properties of the service can only be expressed in terms of message names; we call them *propositional* properties. Therefore, neither of the two formulæ at the bottom of Figure 1a is always true on the modelled workflow.

Conversation specification [5] is an example of sequence of intertwined messages received and sent by multiple agents. Message Sequence Charts (MSC) are modelled into finite state processes by [6]. A similar approach has been done with use of the BPE-calculus and the Concurrency Workbench (CWB) [7] and Petri nets [8]. [9] tackles the formal specification of a protocol of interaction between services expressed as a pattern of messages. These works have been dubbed "data-agnostic" solutions [10]. Currently, some problems, such as local enforceability of global constraints [9], have only been studied in this context.

B. Data-aware Workflows, Propositional Properties

A refinement over the previous solutions is to consider that the actual data exchanged in the messages of a web service can actually influence the control flow of that service: the workflow model becomes "data-aware". This refinement is illustrated in Figure 1b: for example, the choice between sending message "b" or "c" is now unambiguous and determined by the value inside message "a".

This category constitutes the bulk of formal web services models. [11] models web service compositions by finite-state systems and studies them from the angle of synchronicity; it takes the content of variables and message parts into account by extending the original message alphabet. [12] models web services in Propositional Dynamic Logic (PDL) and is interested in generating automated compositions between services. [13] proposes a restricted BPEL semantics for which it is possible to automatically generate the composition of tasks. In [14], the controllability of a business process is studied; the *operating guidelines* of a process P is the automaton that includes as its subgraphs all the possible controllers of P. [15] proposes techniques to extract a behavioural specification from a BPEL process and verify it with model checking techniques.

Other works present automated tools for the validation of the properties. [16] formalizes BPEL web service workflows using a language called CHISEL which is then transformed into LOTOS for automated validation. Multi-agent web services are modelled in [17] using a custom protocol language called MAP which is then translated into SPIN models and model-checked. A process algebra approach is used in [18] to model web service choreographies using the Calculus of Communicating Systems (CCS). [19] uses a formal language called Tropos and validates properties in NuSMV [4]. Finally, in [20], model checking of LTL formulæ expressed in Promela on BPEL specifications is attempted using SPIN [3]. The approach is extended in [21] and constitutes the basis of the Web Service Analysis Tool (WSAT). VERBUS [22] is another tool that translates a web service workflow into a finite-state structure. Finally, [23] studies the two-phase commit protocol and models it using the Temporal Logic of Actions (TLA$^+$).

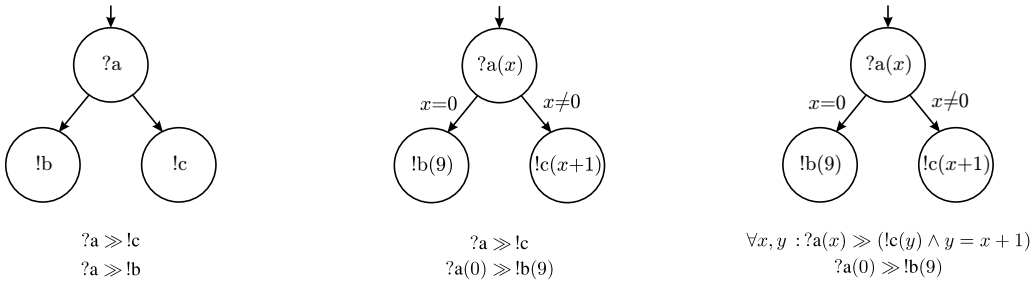
Although these works take data into account when modelling the web services' interactions, this data does not play a role when expressing the properties. The temporal formulæ are still propositional. Actual data content can be referred statically: in Figure 1b, a(0) and b(9) are simply modelled as two new message names. It is not possible, however, to compare the values inside two different messages except by explicitly stating their value.

C. Data-aware Workflows, Data-aware Properties

A further extension consists of allowing quantification on data inside temporal properties, making them "data-aware" as is shown in Figure 1c. Knowledge about the internal workflow generally remains unchanged with respect to the previous category; however, the properties can now express that when "c" is sent, it contains the value of "a" incremented by 1.

In [10], [24], extensions to the temporal logics CTL and LTL, respectively called CTL-FO and LTL-FO, are introduced. These logics include a form of first-order quantification on data. The model presented is very rich: it contains a *database* represented as a variable set of first-order predicates; however this richness is achieved at the price of complexity. The problem of model checking a CTL-FO formula φ on a web service $\mathcal{W}$ (as defined in [24]) is undecidable. The problem of model checking a formula φ without any quantification is in CO-NEXPTIME if the formula is propositional CTL. We show in this paper how a simpler modelling of the services, coupled with a more expressive logic than CTL-FO, is sufficient for model checking important data-aware properties in real-world scenarios. Theorem 2 will demonstrate that CTL-FO$^+$ model checking is PSPACE-complete, a considerably lower complexity.

The validation of interacting databases communicating through Tree Pattern Queries is studied in [25]. Tree Pat-



(a) Propositional workflow, propositional properties

(b) Data-aware workflow, propositional properties

(c) Data-aware workflow, data-aware properties

Figure 1. Workflow modelling with various degrees of data-awareness

tern Queries are tree skeletons used to fetch values inside a structured database, which can then be sent to a remote requester. Temporal formulæ can be expressed in an extension of LTL called Tree LTL. Although this work considers infinite domains, in counterpart it prevents the use of negation, and its model checking becomes undecidable if existential quantification is allowed in TPQs.

The Artifact Behavioral Specification Language (ABSL) [26] is another extension of CTL that includes a form of first-order quantification. However, ABSL is developed in a context of artifact-centric business processes and is suited to express properties of *intra*-artifact behaviours, not *inter*-message constraints; the optimality of the ABSL model checking algorithms also remains to be shown. In the same vein, the language ALBERT [27] provides a way of expressing assertions on the runtime state of a BPEL process by referring to its internal variables and calls to external services. Operators "forall" and "exists" are mentioned in the language syntax and can be used to fetch and memorize elements.

We shall stress that although some of these works give formal complexity-theoretic bounds to the algorithms they present, very few provide proof-of-concept implementations of the kind presented in this paper; as far as we could look, this work is the first to perform an empirical analysis and present actual model checking times of data-aware properties in various scenarios. This step should not be overlooked, as determining the theoretical complexity of a language does not give a complete picture of its capabilities. For example, the decidability of ABSL for finite domains is obtained in [26] by reducing it to classical CTL; the reduction uses *explicit quantification*. Explicit quantification is also used in [27] to reduce ALBERT to classical CTL. As we shall see in Section VII, explicit quantification is only appropriate for very simple cases. The alternate translation that we provide in Section VI, despite being in the same complexity class, performs orders of magnitude faster for most of the properties we considered.

III. Data-Aware Constraints in Web Service Scenarios

To measure the importance of data-awareness in web service workflow validation, we introduce two representative real-world scenarios where temporal constraints on messages arise for a variety of reasons: *technical* constraints stem from the physical or logical nature of the resources involved in the operations, while *policy* constraints deal with business logic and may include membership restrictions, QoS requirements or security. We proceed to show that a number of these constraints are data-aware temporal properties.

A. User-controlled Lightpaths

The User-Controlled Lightpath (UCLP) research project [28] develops an environment that allows end users to self provision and dynamically reconfigure optical networks. To this end, network resources from a specific provider, called Lightpath Objects (LPOs) are virtualized and exposed to the end user as a web service. Each provider gives access to its resources in an Articulated Private Network (APN) via an LPO-factory web service from which LPOs can be controlled and consumed. Each LPO is identified by a unique ID, and the UCLP operations usually manipulate these IDs.

For example, two adjacent LPOs can be *concatenated*; the result of the concatenation operation is an LPO that is considered as one single link. This operation takes as input the ID of some LPOs $i_1, \dots i_n$ and returns a new LPO i corresponding to the result of the concatenation. A simplified version of the `concatenateRequest` message structure is shown below:

```
<message>
  <operation>concatenateRequest</operation>
  <LPO-ID>i1</LPO-ID>
  <LPO-ID>i2</LPO-ID>
  ...
  <LPO-ID>in</LPO-ID>
</message>
```

An LPO's bandwidth can also be *partitioned* into links of equal bandwidth. For instance an OC-3 LPO (155.52 Mbps) can be partitioned into three OC-1 LPOs (51.84 Mbps). Furthermore, as before, during the partition's life-time the original LPO cannot be used in other operations. A request is therefore composed of two elements: i is the ID of the LPO to partition, and b is the bandwidth of the desired fragments. The response from this request returns a list of LPO-IDs resulting from this partition.

Suppose a small UCLP resource provider wants to limit the management overhead of its LPOs; it might want to avoid over-partitioning or over-concatenating its links by imposing that no LPO be involved in more than one operation, either as an input or as an output. Therefore, any LPO will appear at most once:

Choreography Constraint 1. *Every LPO-ID present in a message cannot appear in any future message.*

This constraint clearly has nothing to do with the semantics of the partition operation, but rather with some additional business logic imposed by one particular service provider. Constraints can also arise for technical reasons. For example, the semantics of the concatenate operation supposes that the LPOs to be concatenated are adjacent (i.e. they have exactly one common end). Therefore, although it would be syntactically perfectly valid, it does not make sense to take two LPOs originating from the same partition operation and attempt to concatenate them, as these two LPOs are actually the same end-to-end connection:

Choreography Constraint 2. *If two LPOs are the result of the same partition response, they cannot be involved together in the same concatenate request.*

Many more data-aware constraints can be extracted from this scenario; see for example [29].

B. E-commerce Online Shop

We next consider an e-commerce scenario, adapted from [30], where a shop offers users to buy products through a web service interface. This general context is appropriate to represent many requirements of e-commerce applications.

An external buyer (which can be a human interfacing through a web portal, or another web service acting on behalf of some customer) first logs into the system by providing a user name. The shop offers a discount if a user connects with the commitment to buy at least one product, which is signalled with the `commitToBuy` element. The shop responds to the login with a `loginConfirmation`, providing a unique ID for the session. The user can then retrieve the product list, and ask for more information about each product, such as its price and available quantity, by sending a `getProductDetails` message; the shop replies with a `productDetails` message listing the information for each product. The customer can buy products; this is done by first placing a `buyOrder` message, listing the name and desired amount of each products to be bought:

<message>
  <action>buyOrder</action>
  <product>
    <name> s_1 </name>
    <amount> a_1 </amount>
  </product>
    ...
  <product>
    <name> s_1 </name>
    <amount> a_1 </amount>
  </stock>
</message>

The shop checks the availability of each product and returns a `orderConfirm` with a bill identifier. The last step is for the customer to complete the transaction by proceeding to a cash transfer. This is achieved by providing an account number. The transfer can be done for multiple buy orders at the same time. Alternatively, instead of a cash transfer, a `cancelTransaction` message listing some bill-ids can be sent to revoke these transactions before payment. All these operations can be intertwined.

One can verify that a user who commits to buy actually does so for at least one product before the end of the transaction. This involves the correlation of data elements inside three different messages, as the following constraint shows:

Choreography Constraint 3. *There exists a product p appearing in some* `buyOrder` *message whose bill ID i, returned in some* `orderConfirm` *message, eventually appears in a payment confirmation.*

This choreography specification is indeed "data-aware", because the sequence of messages and their content are interdependent. Again, this scenario lends itself to numerous other data-aware constraints; see [31].

IV. A Data-Aware Temporal Logic

Temporal logics are commonly used in model checking for describing behavioural properties of systems. However, classical temporal formalisms are propositional, and Section II has shown how these languages are only partially appropriate to the modelling and validation of data-aware properties. In this section, we introduce CTL-FO$^+$, an extension of the classical temporal logic CTL.

A. Workflow Modelling

The logic is defined in relation to a suitable model of a web service workflow. In the present context, this representation should take into account the actual messages that are exchanged. In addition, the values of the internal variables used in the original process, since they can influence the control flow, and hence the messages that can be sent or received, should also be kept.

To simplify the presentation, we shall first assume that the states explicitly represent the content of flat XML messages formed of an unordered list of elements; this presentation will be generalized in Section V-D. To this end, we define a set Π of parameters and a set Ω of values that are used to represent the content of the messages. We define a special symbol # that stands for "no-value". Couples of parameters and values form a message element:

Definition 1 (Message elements)**.** *The set of defined message elements is $\mathcal{E}_d = \Pi \times (\Omega \cup \{\#\})$. We also consider the set of undefined message elements, which is the singleton $\mathcal{E}_u = \{(\#, \#)\}$. A* message element *is a member of $\mathcal{E} = \mathcal{E}_d \cup \mathcal{E}_u$.*

The concept of message element closely parallels the structure of a (flat) XML message. The parameters stand for the tag names, while the values represent the data inside the tag. For that reason, the definition of a message element excludes

the possibility that a value has no corresponding parameter. A message is simply an ordered sequence of message elements:

Definition 2 (k-messages)**.** *Let k be a positive integer, and for $i < k$, define $D_i = \{(e_1, \ldots, e_k) : e_i \in \mathcal{E}_u \wedge e_{i+1} \in \mathcal{E}_d\}$. The set of k-messages is defined as $M_k = \mathcal{E}^k \setminus (\bigcup_{i=1}^{k-1} D_i)$.*

Note that this representation does not allow nested tags, and imposes an upper bound k on the number of elements inside a single message. Empty elements are simply ignored; we impose the restriction that all undefined elements be grouped at the end of the k-uple, and therefore one (flat) XML message maps to exactly one message of M_k.

A workflow messaging model is a standard Kripke structure whose states represent the values of each of the internal variables and the message that is being sent or received in that state. That message can be the empty k-message $((\#, \#), \ldots, (\#, \#))$, indicating that no message is either received or sent in that particular state of the model.

Definition 3 (Workflow messaging model)**.** *Let k be a positive integer, Π be a set of parameter names, Ω be a set of value names, $\mathcal{P} = \{p_1, \ldots, p_k\}$ a set of parameter variables, $\mathcal{V} = \{v_1, \ldots, v_k\}$ a set of value variables, $\mathcal{I}$ be a set of internal variables. A* workflow messaging model *is a Kripke structure $\mathcal{M}_k = (S, I, R, L)$ such that:*

- *S is a set of states*
- *$I \subseteq S$ is a set of initial states*
- *$R \subseteq S^2$ is a transition relation over the states*
- *$L = (S \times (\mathcal{P} \cup \mathcal{V} \cup \mathcal{I})) \rightarrow (\Pi \cup \Omega)$ is a labelling function such that for every $s \in S$, $((L(s, p_1), L(s, v_1)), \ldots, (L(s, p_k), L(v_k)))$ is a k-message*

We further suppose that L uniquely identifies every state; that is, there does not exist states $s_0, s_1 \in S$ such that $L(s_0, \alpha) = L(s_1, \alpha)$ for every $\alpha \in \mathcal{P} \cup \mathcal{V} \cup \mathcal{I}$.

A path $\pi = s_0 s_1 \ldots$ is a sequence of states in S such that $(s_i, s_{i+1}) \in R$ for every $i \geq 0$. A workflow messaging model can be seen as a generalized construction of a classical *Moore machine* [32], where the symbol to be printed in a state is replaced with the k-message encoded by the values of state variables in $\mathcal{P}$ and $\mathcal{V}$. Any path in the system corresponds to a possible sequence of messages in a service interaction. Properties about message sequences become properties on sequences of states that can then be expressed using temporal logic.

The translation of a business process into a workflow messaging model is outside the scope of this paper; in the spirit of [24], we assume it is given. There exists numerous works providing formal models from various input notations: UML Message Sequence Charts [33], CRESS [16], BPEL [20], [34]–[36]. Some of these tools even manage exception handling and compensation procedures as alternate flows.

B. Syntax and Semantics of CTL-FO$^+$

The Computation Tree Logic with Full First-order Quantification (CTL-FO$^+$) is is aimed at describing sequentialities in a finite-state system while allowing full quantification over data; it is an extension of the well-known temporal logic CTL [37]. CTL and a related logic called LTL are the most commonly used languages for describing sequentialities in finite-state systems. All major model checking tools, such as SPIN [3] and NuSMV [4], verify temporal formulæ expressed in one of these logics. The reader is referred to [37] for a deeper coverage of CTL and other temporal logics.

Formulæ are built from Boolean variables and the constants *true* and *false* using the classical connectors: $\wedge$ (and), $\vee$ (or), $\rightarrow$ (implies) and $\neg$ (not). CTL-FO$^+$ further provides *temporal operators*, taken directly from CTL, that can be used on top of traditional propositional logic formulæ to specify the temporal conditions. *Universal* operators assert properties about all executions starting from the current state. The first of these operators is **AG**, which means "globally". For example, the formula $\mathbf{AG}\,\varphi$ means that formula φ is true in every state of every execution starting at the current state. The operator **AF** means "eventually"; the formula $\mathbf{AF}\,\varphi$ is true whenever for all executions, φ holds for some future state. The operator **AX** means "next"; it is true whenever φ holds in any possible next state of the current state. Finally, the **AU** operator means "until"; the formula $\mathbf{A}\,\varphi\,\mathbf{U}\,\psi$ is true if, in any execution sequence, φ holds for all states until ψ holds. *Existential* operators, designated by **EG**, **EF**, **EX** and **EU**, are defined in the same way as their universal equivalents, except that the condition holds only for some instead of all possible sequences.

We extend the expressiveness of the traditional CTL by adding first-order quantification. The resulting language has the following formal syntax and semantics.

Definition 4 (Syntax)**.** *The language CTL-FO$^+$ (Computation Tree Logic with Full First-order Quantification) is obtained by closing CTL under the following construction rules: Let x and y be variables or constants; let let φ and ψ be CTL-FO$^+$ formulæ, x_i be a free variable in φ, $p \in \Pi$ be a parameter name; then $x = y$, $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, $\mathbf{AG}\,\varphi$, $\mathbf{EG}\,\varphi$, $\mathbf{AF}\,\varphi$, $\mathbf{EF}\,\varphi$, $\mathbf{AX}\,\varphi$, $\mathbf{EX}\,\varphi$, $\mathbf{A}\,\varphi\,\mathbf{U}\,\psi$, $\mathbf{E}\,\varphi\,\mathbf{U}\,\psi$, $\exists_p x_i : \varphi$ and $\forall_p x_i : \varphi$ are CTL-FO$^+$ formulæ.*

Definition 5 (Semantics)**.** *Let $\mathcal{M}_k$ be a workflow messaging model, and $s \in S$ be a state. For $p \in \Pi$, let $Dom_s(p) = \{L(s, v_i) : L(s, p_i) = p, 1 \leq i \leq k\}$, and c_1 and c_2 be constants. Let $X = \{x_1, \ldots, x_n\}$ be the set of variables in φ and $\nu : X \rightarrow \Omega \cup \{\#\}$ a valuation that maps each variable to a possible value. By extension, ν maps any constant c to itself. We denote by $\nu[a/x_j]$ the valuation that agrees with ν on every $x_i \in X$ with the exception of x_j for which it returns a. We say the triplet $\mathcal{M}_k, s, \nu$ satisfies the CTL-FO$^+$ formula φ, and write $\mathcal{M}_k, s, \nu \models \varphi$ if and only if it follows the rules given in Table I. By extension, we write $\mathcal{M}_k \models \varphi$ if all initial states $s \in I$ of $\mathcal{M}_k$ and the empty valuation $\nu(x) = \#$ for all $x \in X$ are such that $\mathcal{M}_k, s, \nu \models \varphi$.*

The set of operators $\neg$, $\vee$, **AF**, **EX**, **EU**, and $\exists$ is called an *adequate* set of connectives in that all other operators can be derived from a combination of them with the following identities: $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$, $\mathbf{EF}\,\varphi \equiv \mathbf{E}\,[true\,\mathbf{U}\,\varphi]$, $\mathbf{AX}\,\varphi \equiv \neg\mathbf{EX}\,\neg\varphi$, $\mathbf{AG}\,\varphi \equiv \neg\mathbf{E}\,[true\,\mathbf{U}\,\neg\varphi]$, $\mathbf{A}\,[\varphi\,\mathbf{U}\,\psi] \equiv \neg(\mathbf{E}\,[(\neg\varphi)\,\mathbf{U}\,\neg(\varphi \vee \psi)] \vee \mathbf{EG}\,\neg\psi)$, $\forall_p x_i : \varphi \equiv \neg(\exists_p x_i : \neg\varphi)$.

$\mathcal{M}_k, s, \nu \models x_i = x_j$	$\Leftrightarrow$	$\nu(x_i)$ is equal to $\nu(x_j)$
$\mathcal{M}_k, s, \nu \models \exists_p x_i : \varphi$	$\Leftrightarrow$	there exists $a \in Dom_s(p)$ such that $\mathcal{M}_k, s, \nu[a/x_i] \models \varphi$

Table I
FORMAL SEMANTICS OF CTL-FO$^+$. THE SEMANTICS FOR THE BOOLEAN CONNECTIVES AND TEMPORAL OPERATORS IS IDENTICAL TO CTL'S

This result is classical [38].

Without loss of generality, we assume that CTL-FO$^+$ formulæ φ with n quantified variables are *well-named*: each variable is quantified only once and in the order $x_1, x_2, \ldots, x_n$. Every CTL-FO$^+$ formula can be transformed by a simple renaming of its variables to a well-named formula. Then, the valuations ν used in the previous semantics can be restricted to *ordered* valuations, which define variables progressively in the order $x_1, x_2, \ldots, x_n$. An ordered t-valuation is an ordered valuation for which exactly the first t variables $x_1, x_2, \ldots, x_t$ have been defined. We then define $p_i \in \Pi$ as the parameter name appearing in the quantification of variable x_i in φ.

CTL-FO$^+$ is reminiscent of [39] which introduces a logic called EQCTL that extends CTL by allowing existential quantification over *state variables*. EQCTL is not closed under negation; therefore, universal quantification cannot be obtained. CTL-FO$^+$ quantifies over *values* and is is closer to true first-order quantification. Furthermore, the model checking of EQCTL is NP-complete, while we show later that model checking in CTL-FO$^+$ is in a higher complexity class. A closer work is QCTL [40] which extends CTL by including first-order quantification and monadic second-order quantification over arbitrary *algebraic data structures*; such expressiveness is not required in our case. Finally, CTL-FO$^+$ can freely mix temporal and data quantification without restriction. This is an extension over the logic CTL-FO defined in [24], which does not allow formulæ containing temporal operators to be existentially quantified.

CTL-FO$^+$ is expressive enough to model a wide range of existing notations. In particular, it can be used to express safety, fairness, and liveness properties, and sequentiality properties in UML Sequence Diagrams. The three operators of the *Let's Dance* choreography language ("precedes", "inhibits" and "weak-precedes") [9] can be mapped into CTL-FO$^+$, as well as all the existence and relation formulæ of the DecSerFlow language [41], and the portion of BPMN and BEMN which has been formalised into temporal logic [42]. It can also express properties that are beyond any of these languages, such as all data-aware constraints. By taking time as an additional global variable, the logic also expresses metric temporal properties such as time windows (B happens at least/most k seconds after A, where k is a constant) and time filters (B happens at most x seconds after A, where x is fetched from a message in the conversation); see [43].

C. Formalizing Web Service Properties

The values of the variables appearing in a CTL-FO$^+$ formula are quantified according to specific parts of the XML message that is received or sent in the current state of the system. A quantifier like $\forall_{\text{LPO-ID}} x$ therefore means "for all values x of elements named LPO-ID in the current message". This form of quantification is sufficient, since when referring to message data, it is never necessary to quantify over all values of all elements in the message; rather, we normally want to quantify for all values of a specific element name. As an example, Choreography Specification 1 becomes the following CTL-FO$^+$ formula:

Formal Choreography Constraint 1.

$$\mathbf{AG}\,(\forall_{\textit{LPO-ID}}\, x_1 : \mathbf{AXAG}(\forall_{\textit{LPO-ID}}\, x_2 : x_1 \neq x_2))$$

It is important to remark that quantification only refers to values occurring *in the current message*. Variables x_1 and x_2 both quantify over LPO-ID elements. If quantification did not depend on the current message, the previous formula would always be false, as any value c bound to x_1 would also be admissible for x_2, making the assertion $x_1 \neq x_2$ false at least once. The previous formula rather states that at any time in any execution of the script, for any LPO-ID x_1 appearing in a message, then from now on in any *future* message, any LPO-ID x_2 is different from x_1. Hence, it will be true exactly when no LPO-ID appears more than once in any execution, which is consistent with Choreography Specification 1.

In a similar way, Choreography Specification 2 becomes the following, more complex CTL-FO$^+$ formula:

Formal Choreography Constraint 2.

$$\begin{aligned}&\mathbf{AG}\,(\forall_{\textit{operation}}\, x_1 : x_1 = \textit{concatenateRequest} \rightarrow\\ &\forall_{\textit{LPO-ID}}\, x_2 : \mathbf{AX}\ \mathbf{AG}\ (\forall_{\textit{operation}}\, x_3 :\\ &(x_3 = \textit{partitionRequest} \vee x_3 = \textit{concatenateRequest})\\ &\rightarrow \forall_{\textit{LPO-ID}}\, x_4 : x_2 \neq x_4))\end{aligned}$$

This formula states that at any time in any execution of the script, if the operation x_1 of the message is concatenateRequest, then for every LPO-ID x_2 appearing in this message, we have that for every future message whose operation element value x_3 is partitionRequest or concatenateRequest, any value x_4 for its LPO-ID is different from x_2. In other words, once an LPO has been concatenated, no further partition or concatenation involves this LPO, which is indeed equivalent to Choreography Constraint 2.

As a last example, Choreography Constraint 3 becomes the following CTL-FO$^+$ formula in seven variables; remark that in this case, existential quantification is necessary:

Formal Choreography Constraint 3.

$$\begin{aligned}&\mathbf{AF}\,(\exists_{\textit{action}}\, x_1 : x_1 = \textit{buyOrder} \wedge\\ &\exists_{\textit{product/name}}\, x_2 : \mathbf{AF}\,(\exists_{\textit{action}} x_3 :\\ &(x_3 \neq \textit{orderConfirm} \wedge (\exists_{\textit{product/name}} x_4 : \exists_{\textit{bill-ID}} x_5 :\\ &\mathbf{AF}\,(\exists_{\textit{action}} x_6\ \exists_{\textit{bill-ID}} x_7\\ &x_6 = \textit{confirmPayment} \wedge x_7 = x_5)))))\end{aligned}$$

V. VALIDATING CTL-FO$^+$ PROPERTIES

In this section, we show how CTL-FO$^+$ formulæ can be actually validated on a web service workflow by presenting a model checking algorithm. The complexity of this algorithm

Procedure CHECK($a = b$, ν)
  If $\nu(a) = \nu(b)$
    Return S
  Else
    Return $\emptyset$
  End if
End procedure

Procedure CHECK($\exists_p x : \varphi$, ν)
  $N := \emptyset$
  For each $s \in S$
    For each $a \in Dom_s(p)$
      If $s \in$ CHECK$(\varphi, \nu[a/x])$
        $N := N \cup \{s\}$
      End if
    End for
  End for
  Return N
End procedure

Table II
THE RECURSIVE MODEL CHECKING PROCEDURE FOR CTL-FO$^+$

is then established and discussed. In particular, we show that CTL-FO$^+$ model checking is a problem as tractable as the LTL model checking problem that is widely used in the industry. Finally, we show that any web service model that uses a data-aware workflow, but *propositional* properties cannot efficiently simulate data-awareness.

A. Model Checking CTL-FO$^+$

The model checking for CTL-FO$^+$ formulæ is derived from the classical CTL model checking algorithm and is presented in Table II. Given a valuation ν, the procedure CHECK performs by structural recursion on the CTL-FO$^+$ formula φ and consists in forming recursively the set of states s such that $\mathcal{M}_k, s, \nu \models \varphi$. Ground equality testing is evaluated by comparing the valuation of each side. If the main operator of the subformula to check is a Boolean connective or a temporal modality, the algorithm is identical to the model checking of a CTL formula defined in [44].

Differences arise when the main operator of the formula is an existential quantifier, $\exists x : \varphi(x)$. In such a case, the algorithm successively applies the model checking of $\varphi(a)$, for a in the domain of x, and keeps states which satisfy at least one $\varphi(a)$. Finally, model checking of ground terms is composed of equality testing. Depending on the valuation and the terms to be compared, either the entire model satisfies it if the assertion is true, or no state satisfies it if the assertion is false.

A workflow messaging model $\mathcal{M}_k$ satisfies the global CTL-FO$^+$ formula φ if and only if all its initial states are in the set returned by CHECK(φ, ν), with ν the empty valuation.

Theorem 1. CHECK *is sound and complete.*

Proof: The proof is done by structural induction on the formula φ. Base case: ground formulæ are equality testings and the result is direct. Induction step: suppose that CHECK is sound and complete for every formula of length less than ℓ. Let φ be a formula of length ℓ. For Boolean connectives and temporal operators, the procedure CHECK is identical to CTL's [44] and its soundness and completeness are assumed. The remaining case not covered is that of existential quantification. The lines 4-6 of the procedure CHECK$(\exists_p x : \varphi, \nu)$ are such that a state s is added to set N if and only if there exists at least one value $a \in Dom_s(p)$ such that $s \in$ CHECK$(\varphi, \nu[a/x])$. By the induction hypothesis, this is the case if and only if $\mathcal{M}_k, s, \nu[a/x] \models \varphi$. Since this loop is repeated for every $s \in S$, at the end of the procedure we have that $s \in N =$ CHECK$(\exists_p x : \varphi, \nu)$ if and only if there exists $a \in Dom_s(p)$ such that $\mathcal{M}_k, s, \nu[a/x] \models \varphi$. By Definition 5, this in turn is equivalent to $\mathcal{M}_k, s, \nu \models \exists_p x : \varphi$. ■

The model checking of CTL-FO$^+$ requires a workflow messaging model with finite domains. If the original process produces a finite number of values, then these values can be directly used to build the workflow messaging model. If the process manipulates a potentially infinite domain, a finite abstraction of the infinite space must first be applied before building the model. For example, since atoms in CTL-FO$^+$ are only equality tests, the original domain can be replaced by a finite number of symbolic values representing different equivalence classes; among other possible approaches, we also mention finite sampling of an infinite set by random selection of values [27], and abstraction to Boolean values and progressive refinement of the model [45]. However, as explained in Section IV-A, the translation between a business process and a workflow messaging model is outside the scope of this paper; therefore any mapping to the actual values of the original process is assumed to be taken care of independently.

B. CTL-FO$^+$ Model Checking is PSPACE-complete

We now establish the complexity of model checking CTL-FO$^+$ formulæ and show that data-aware properties cannot be modelled effectively by propositional properties.

Theorem 2. *Let φ be a CTL-FO$^+$ formula and $\mathcal{M}_k$ be a workflow messaging model. Determining whether $\mathcal{M}_k \models \varphi$ is PSPACE-complete.*

Proof: We first show that the model checking problem is PSPACE-hard by reducing the quantified Boolean formula problem (QBF), known to be PSPACE-complete [46], to CTL-FO$^+$ model checking. A quantified Boolean formula φ is of the form $Q^1 x_1 Q^2 x_2 \ldots Q^n x_n \varphi$, where Q^i is either the existential ($\exists$) or the universal ($\forall$) quantifier and the x_i are Boolean variables (their domain is $\{0, 1\}$). Consider the workflow messaging model $\mathcal{M}_2$ consisting of a single state s (which is also the initial state), a transition relation $\{(s, s)\}$ and where the 2-message in state s is $((p, 0), (p, 1))$ for some dummy parameter name p. Then, by rewriting the quantifiers Qx_i in the original QBF to $Q^i_p x_i$, φ becomes a CTL-FO$^+$ formula φ' where each variable x_i has domain $Dom_s(p) = \{0, 1\}$. Therefore, φ is satisfiable if and only if $\mathcal{M}_2 \models \varphi'$.

The second step consists in showing that the procedure CHECK is in PSPACE. Each recursive call receives as arguments a subformula bounded by the size of the original formula φ and a valuation ν whose cardinality is fixed. Depending on the case to be considered, each call uses at

most two subsets of S during its computation, and returns a subset of S. Therefore, the space consumed by each recursive call is linear in the size of both the formula and the structure. Since the number of calls is bounded by the length of the formula, this algorithm is polynomial in the size of the CTL-FO$^+$ formula and the transition system. Remark that the PSPACE class of decision problems only requires polynomial use of *memory space*; the algorithm is clearly exponential with respect to time. ∎

The PSPACE-completeness result places CTL-FO$^+$ model checking for finite domains in the same complexity class as model checking of an LTL formula [37]. LTL is a temporal logic widely used in the industry, for example in conjunction with the SPIN model checker, and many works with *propositional* properties mentioned in Section II-B use LTL as their language for expressing constraints on message sequences. Therefore, although CTL-FO$^+$ allows to fully access the data content of the messages, its model checking problem has an equivalent complexity to many other existing solutions that do not provide data-aware temporal capabilities.

C. Simulating Data-awareness with Propositional Properties

Studying the complexity of the CTL-FO$^+$ model checking algorithm can teach us more. Since the domain for each variable is finite, it is possible to use the semantics of Definition 5 and convert each quantifier into a conjunction or a disjunction of a finite number of terms. The resulting expression is a plain CTL formula where all references to data are static; we call this approach *explicit quantification*. In turn, this expansion of a CTL-FO$^+$ formula φ into a propositional CTL formula φ' is exponential in the number of quantifiers, since each quantified subformula must be repeated once for each possible value in the domain. However, the model checking algorithm of a CTL formula in in P: it has a worst-case running time linear in the size of the formula to check, and linear in the size of the Kripke structure [38]. Therefore, using CTL model checking on φ' takes exponential time, which is no worse than the runtime of CTL-FO$^+$ model checking on φ.

One might then think that CTL-FO$^+$ is simply CTL with an additional level of syntactic sugar, and that data-aware workflows with *propositional* properties, as described in Section II-B, are already sufficient to model any data-aware property by simply extending the message alphabet. However, this is not the case; the following theorem shows that it is highly unlikely that any polynomial reduction of CTL-FO$^+$ to CTL exists.

Theorem 3. *If there exists a polynomial reduction of CTL-FO$^+$ model checking to CTL model checking, then P = NP.*

Proof: A polynomial reduction of CTL-FO$^+$ model checking to CTL model checking entails that for every workflow messaging model $\mathcal{M}_k$ and every CTL-FO$^+$ formula φ, there exists a Kripke structure K' and a CTL formula φ' such that $\mathcal{M}_k \models \varphi$ if and only if $K' \models \varphi'$. Moreover, the size of K' and φ are respectively polynomial in the sizes of $\mathcal{M}_k$ and φ. Since CTL-FO$^+$ model checking is PSPACE-complete and CTL model checking is in P, we have PSPACE $\subseteq$ P. The result follows since P $\subseteq$ NP $\subseteq$ PSPACE. ∎

Therefore, unless P = NP, any attempt at using data-aware workflows with propositional properties to model data-aware properties will either blow the size of the formulæ or the size of the model by an exponential factor. In other words, CTL-FO$^+$ is exponentially more succinct than any propositional modelling of data-awareness.

Furthermore, with explicit quantification, the translation of constraints into temporal logic becomes tightly coupled with the actual script on which it has to be checked. This is because the translation of the quantifiers shown depends on the values occurring in the script. It is, however, unrealistic that a service provider advertises its constraints in such a manner: one would have to know in advance all possible values occurring in scripts prepared by third-parties to include them in the large disjunction. Finally, we suspect standard model checkers such as NuSMV [4] to easily handle systems with very large state spaces and reasonably short temporal formulæ, but to be far less efficient for checking exponentially long formulæ. The experimental results in Section VII will confirm this intuition.

D. Generalization to Nested Message Elements

Up to now, messages in the workflow messaging model were represented explicitly as flat, unordered lists of element-value pairs. The quantifier $\exists_p x$ constrains the domain of x to the values of p elements in the current message. For a given CTL-FO$^+$ formula φ, one can compute the set of element names $\{p_1, \ldots, p_n\}$ occurring in a quantifier. Since the p_i are the only message parts that are accessed, all message elements outside this set are irrelevant to φ; hence, the workflow messaging model only needs to encode a projection of the actual messages onto the set $\{p_1, \ldots, p_n\}$.

This construction can be generalized by replacing the p_i with any static expression ϵ_i which can fetch a set of values inside a message; in particular, ϵ_i can be an XPath expression specifying branches of a particular form, such as $p_1/p_2/\ldots/p_n$, where the p_i are static element names. In this case, the workflow messaging model only needs to encode a projection of the actual messages onto the set of XPath expressions $\{\epsilon_1, \ldots, \epsilon_n\}$. Therefore, the workflow messaging model can be used to represent access to nested message elements reachable by standard, static path expressions.

VI. An Efficient Reduction of CTL-FO$^+$ to CTL

Theorem 3 indicates that in fact, *any* attempt to use standard CTL model checkers to validate data-aware workflow properties is "doomed" to an exponential blow-up of the original problem, and not only the explicit quantification method suggested above.

Nevertheless, in this section, we show an alternate translation of the CTL-FO$^+$ model checking problem to CTL. In this approach, the original CTL-FO$^+$ formula is transformed into a CTL formula, but the original workflow messaging model is also transformed by adding new state variables. These additional variables are used to "freeze" the value of a state variable at some point in the execution for future reference; consequently the transformation technique used is

called "freeze quantification". It has been originally developed in [47] for timed transitions systems.

We proceed in two steps: first, we show how to convert a workflow messaging model $\mathcal{M}_k$ and a CTL-FO$^+$ formula φ with n variables into a *freeze workflow messaging model* $\widehat{\mathcal{M}}_k^n$; then, we show how a CTL-FO$^+$ formula φ can be translated to a CTL formula $\widehat{\varphi}$ and show that φ is true for $\mathcal{M}_k$ if and only if $\widehat{\varphi}$ is true for $\widehat{\mathcal{M}}_k^n$, thereby reducing the problem of CTL-FO$^+$ model checking to CTL model checking.

The number of states in $\widehat{\mathcal{M}}_k^n$ is exponential with respect to the number of states in $\mathcal{M}_k$ and hence the reduction is still in EXPTIME; however, the original CTL-FO$^+$ formula φ is transformed into a CTL formula whose size is *linear* with respect to φ. We shall see in Section VII that for this reason, this reduction performs much better than the explicit quantification approach for complex formulæ.

A. Transforming a Kripke Structure

Let $\mathcal{M}_k = (S, I, R, L)$ be a workflow messaging model defined over parameters Π and values Ω, with sets of state variables $\mathcal{P}$, $\mathcal{V}$ and $\mathcal{I}$ defined as previously. We will build a *freeze* workflow messaging model $\widehat{\mathcal{M}}_k^n = (\widehat{S}, \widehat{I}, \widehat{R}, \widehat{L})$ by including an additional set of state variables $\mathcal{F} = \{\widehat{\nu}_1, \ldots, \widehat{\nu}_n\}$, called the "freeze" variables, intended to capture the value of some part of a message at a given point in the execution of the workflow. Intuitively, the $\widehat{\nu}_i$ will be used to represent inside the workflow messaging model the possible ordered valuations ν of the variables $x_1, \ldots, x_n$ in the original CTL-FO$^+$ formula.

The labelling function L is extended to the freeze variables and is defined as $\widehat{L} : (\widehat{S} \times (\mathcal{P} \cup \mathcal{V} \cup \mathcal{I} \cup \mathcal{F})) \to (\Pi \cup \Omega)$. For $0 \leq t \leq n$, we define the set $\Omega_t^n \subset (\Omega \cup \{\#\})^n$ such that $(\widehat{\nu}_1, \ldots, \widehat{\nu}_n) \in \Omega_t^n$ if and only if $\widehat{\nu}_i \neq \#$ for $1 \leq i \leq t$ and $\widehat{\nu}_i = \#$ otherwise. The set Ω_t^n contains all possible ordered t-valuations.

The set of system states $\widehat{S}$ and the behaviour of the labelling function $\widehat{L}$ on this set are defined as follows:

Definition 6 (Set of system states, labelling)**.** *Let $s \in S$ be a state of $\mathcal{M}_k$ and t be an integer such that $0 \leq t \leq n$. Then $\widehat{s} \in \widehat{S}$ is a state of $\widehat{\mathcal{M}}_k^n$ if and only if:*

- $L(s, \alpha) = \widehat{L}(\widehat{s}, \alpha)$ *for every* $\alpha \in \mathcal{P} \cup \mathcal{V} \cup \mathcal{I}$
- $(\widehat{L}(\widehat{s}, \widehat{\nu}_1), \ldots, \widehat{L}(\widehat{s}, \widehat{\nu}_n)) \in \Omega_t^n$.

This definition entails that the relation between S and $\widehat{S}$ is surjective: for every state $s \in S$ there exists multiple "copies" $\widehat{s}_\nu \in \widehat{S}$ that agree on the labelling function for s for every state variable of $\mathcal{M}_k$, and for which the freeze variables encode every possible ordered valuation ν of the x_i. Therefore, for every $0 \leq t \leq n$ and every $\nu \in \Omega_t^n$, the sets $\widehat{S}_\nu = \left\{ \widehat{s}_\nu : \widehat{L}(\widehat{s}_\nu, \widehat{\nu}_i) = \nu(x_i), 1 \leq i \leq n \right\}$ form a partition of $\widehat{S}$; each $\widehat{S}_\nu$ is a "copy" of S where the $\widehat{\nu}_i$ encode one specific ordered valuation, ν.

The initial states of $\widehat{\mathcal{M}}_k^n$ are the copies of the initial states of $\mathcal{M}_k$ where the $\widehat{\nu}_i$ encode the empty valuation.

Definition 7 (Initial states)**.** *Let $s \in S$ be a state of $\mathcal{M}_k$ and $\widehat{s}_\nu \in \widehat{S}$ be a state of $\widehat{\mathcal{M}}_k^n$ such that $\nu \in \Omega_0^n$. Then $s \in I$ if and only if $\widehat{s}_\nu \in \widehat{I}$.*

The transition relation $\widehat{R}$ is defined as the union of two relations, $\widehat{R}_w$ and $\widehat{R}_f$. The transitions contained in the first part, $\widehat{R}_w$, reproduce in each partition $\widehat{S}_\nu$ the original transition relation R. They are called *workflow transitions*.

Definition 8 (Transition relation: workflow transitions)**.** *Let $s, s' \in S$, $\nu \in \Omega_t^n$ for some $0 \leq t \leq n$ and $\widehat{s}_\nu, \widehat{s}'_\nu \in \widehat{S}_\nu$. Then $(s, s') \in R$ if and only if $(\widehat{s}_\nu, \widehat{s}'_\nu) \in \widehat{R}_w$.*

The transitions contained in the second part, $\widehat{R}_f$, simulate the definition of a new variable into the valuation.

Definition 9 (Transition relation: freeze transitions)**.** *Let $s \in S$. Let $\nu \in \Omega_t^n$ for some $0 \leq t \leq n$, $\nu' \in \Omega_{t+1}^n$ such that $\nu(x_i) = \nu'(x_i)$ for $1 \leq i \leq t$. Then $(\widehat{s}_\nu, \widehat{s}_{\nu'}) \in \widehat{R}_f$ if and only if $\nu'_{t+1} \in Dom_s(\pi_{t+1})$.*

These are called *freeze transitions*, since the workflow messaging model switches between two copies $\widehat{s}_\nu$, $\widehat{s}_{\nu'}$ of the same original state $s \in S$. Thus, the action of the original workflow messaging model is suspended while a variable $\widehat{\nu}_i$ takes a value. We say that $\widehat{\mathcal{M}}_k^n$ is in a *freezing phrase* when it advances to its next state through a freeze transition.

Following the semantics of CTL-FO$^+$, the domain of each freeze variable is dependent on the message part on which they are defined: the definition imposes that in state s, if the variable that takes a value is $\widehat{\nu}_i$, then this value must be from $Dom_s(\pi_{i+1})$.

The actual value assigned to either of these special variables in each state is non-deterministic among all possible values in $Dom_s(\pi_{i+1})$. In addition, each variable may or may not take a value –that is, variables can stay undefined. However, once a variable has taken a definite value, it keeps this value for the remainder of the execution trace. Finally, any number of freeze transitions can be taken before resuming the execution of $\mathcal{M}_k$ by taking again a workflow transition. This entails that any number of variables can be assigned in a freezing phase, provided that they are assigned in lexicographical order and that their domain is not empty for that state.

Definitions 6 to 9 completely specify $\widehat{\mathcal{M}}_k^n$ from $\mathcal{M}_k$. It is important to remark that $\widehat{\mathcal{M}}_k^n$ also depends on the CTL-FO$^+$ formula to check, φ, but only in the number of variables and the parameters π_i on which each x_i is quantified.

B. Converting a CTL-FO$^+$ Formula

Once a workflow messaging model $\mathcal{M}_k$ has been translated into a freeze workflow messaging model $\widehat{\mathcal{M}}_k^n$, the CTL-FO$^+$ formula on $\mathcal{M}_k$ can be translated into a standard CTL formula on $\widehat{\mathcal{M}}_k^n$.

We first define a class of auxiliary formulæ γ_t^n, called the *guards*. Intuitively, γ_t^n describes the fact that the variables $\widehat{\nu}_1, \ldots, \widehat{\nu}_n$ encode an ordered t-valuation in Ω_t^n.

Definition 10 (Guard)**.** *Let t, n be positive integers such that $0 \leq t \leq n$. The* guard *γ_t^n is the CTL formula:*

$$\gamma_t^n = \left(\bigwedge_{i=1}^{t} \widehat{\nu}_i \neq \# \right) \wedge \left(\bigwedge_{i=t+1}^{n} \widehat{\nu}_i = \# \right)$$

It can be observed that by definition, we have that γ_t^n holds in a state $\widehat{s} \in \widehat{S}$ if and only if $(\widehat{L}(\widehat{s}, \widehat{\nu}_1), \ldots, \widehat{L}(\widehat{s}, \widehat{\nu}_n)) \in \Omega_t^n$.

We define a linear embedding ω_t^n of CTL-FO$^+$ into CTL formulæ which performs by structural induction on the original CTL-FO$^+$ formula φ. In the same way as the semantics of CTL-FO$^+$ depends on the valuation of the variables ν, the translation ω_t^n depends on t, the number of variables whose value is already defined. Therefore, $\omega_t^n(\varphi)$ returns the CTL translation of φ, given that t out of n variables are already defined.

Let φ_1 and φ_2 be CTL-FO$^+$ subformulæ, c_1, c_2 be constants in Ω, t, n be integers defined as above, $p \in \Pi$ be a parameter name and the $x_1, \ldots x_n$ be the n quantified variables in the CTL-FO$^+$ formula φ. Translating the Boolean connectives and the ground equality testings is direct.

$$\omega_t^n(c_1 = c_2) \equiv c_1 = c_2 \quad (1)$$
$$\omega_t^n(x_i = c_1) \equiv \widehat{\nu}_i = c_1 \quad (2)$$
$$\omega_t^n(x_i = x_j) \equiv \widehat{\nu}_i = \widehat{\nu}_j \quad (3)$$
$$\omega_t^n(\neg\varphi_1) \equiv \neg\omega_t^n(\varphi_1) \quad (4)$$
$$\omega_t^n(\varphi_1 \vee \varphi_2) \equiv \omega_t^n(\varphi_1) \vee \omega_t^n(\varphi_2) \quad (5)$$

The translation of the CTL temporal operators requires more work; we explain them one by one. The semantics of the **EX** operator requires that there exists one execution path in $\mathcal{M}_k$ for which the next state respects some property. In $\widehat{\mathcal{M}}_k^n$, not all possible execution paths are admissible; remember that in *freeze* transitions $(\widehat{s}_0, \widehat{s}_1) \in \widehat{R}_f$ the states $\widehat{s}_0$ and $\widehat{s}_1$ are two copies of the same original state in $\mathcal{M}_k$, and do not represent an actual progression of the execution of $\mathcal{M}_k$. Therefore, the next states reached through these freeze transitions are not "real" next states of the execution and must be discarded. Only next states reached through workflow transitions $(\widehat{s}_0, \widehat{s}_1) \in \widehat{R}_f$ must be considered. These states can be characterized by the fact that the $\widehat{\nu}_i$ encode a t-valuation which, by Definition 9, must be the same as that in $\widehat{s}_0$. Hence, only next states that verify both γ_t^n and $\omega_t^n(\varphi)$ are valid candidates. This yields the following equation:

$$\omega_t^n(\mathbf{EX}\,\varphi_1) \equiv \mathbf{EX}\,(\gamma_t^n \wedge \omega_t^n(\varphi_1)) \quad (6)$$

A similar adaptation must be done to preserve the semantics of the **AF** operator. In the original semantics, $\mathbf{AF}\,\varphi$ requires that every execution path in $\mathcal{M}_k$ starting from the current state is such that there exists a state that verifies φ. Again, not all paths must be considered: states accessible through freeze transitions must be discarded. The only paths that must fulfil $\mathbf{F}\,\varphi$ are those which do not take a freeze transition:

$$\omega_t^n(\mathbf{AF}\,\varphi_1) \equiv \mathbf{A}\,[\gamma_t^n\,\mathbf{U}\,(\neg\gamma_t^n \vee (\gamma_t^n \wedge \omega_t^n(\varphi_1)))] \quad (7)$$

The translation of the **AF** operator is a generalization of the traditional CTL **AF**, defined as $\mathbf{AF}\,\varphi \equiv \mathbf{A}\,[true\,\mathbf{U}\,\varphi]$; it suffices to replace γ_t^n by $true$ to recover the original definition. Therefore, the guard γ_t^n can be seen as a filter that determines which paths are admissible.

The case of **EU** is adapted following the same principle:

$$\omega_t^n(\mathbf{E}\,\varphi_1\,\mathbf{U}\,\varphi_2) \equiv \mathbf{E}\,[(\gamma_t^n \wedge \omega_t^n(\varphi_1))\,\mathbf{U}\,(\gamma_t^n \wedge \omega_t^n(\varphi_2))] \quad (8)$$

Equation (8) imposes the existence of a path where no freeze transition is taken, by adding the guard γ_t^n to both subformulæ φ_1 and φ_2.

The quantification on variables becomes a quantification on some execution paths. Indeed, a quantifier like $\exists_p x_i : \varphi$ actually means "there exists a value a that variable x_i can take in the current state such that φ holds". According to the Kripke structure $\mathcal{M}_k$ defined previously, this simply amounts to asserting that in the current state, there exists a way for $\widehat{\nu}_i$ of changing from # to some definite value, such that the translation of φ is true. By Definition 9, the only values $\widehat{\nu}_i$ can change to are in $Dom_{\widehat{s}}(\pi_i)$ for s for the current state $\widehat{s}$. This translates as follows:

$$\omega_t^n(\exists_p x_i : \varphi_1) \equiv \mathbf{EX}\,\left(\gamma_{t+1}^n \wedge \omega_{t+1}^n(\varphi_1)\right) \quad (9)$$

Using this embedding, Choreography Specification 2 is recursively translated to the following CTL expression. The translation for **AG** and **AX** has been obtained from the above equations using the classical identities mentioned in Section IV-B.

$$\begin{array}{l}\neg\mathbf{E}\,(\gamma_0^4\,\mathbf{U}\,(\gamma_0^4 \wedge \\ \quad (\neg(\mathbf{AX}\,(\gamma_1^4 \rightarrow (x_1 = \text{partitionResponse} \rightarrow \\ \quad\quad (\mathbf{AX}\,(\gamma_2^4 \rightarrow (\mathbf{AX}\,(\gamma_2^4 \rightarrow \mathbf{A}\,(\gamma_2^4\,\mathbf{U}\,(\gamma_2^4 \vee (\mathbf{AX}\,(\gamma_3^4 \rightarrow \\ \quad\quad\quad (x_3 = \text{concatenateRequest} \rightarrow \\ \quad\quad\quad\quad (\mathbf{EX}\,(\gamma_4^4 \wedge x_2 = x_4))))))))))))))))\end{array}$$

We do not expect data aware constraints to be expressed directly in CTL in such a way. However, the translation from CTL-FO$^+$ to CTL can be automated, and the next theorem shows that the overall construction preserves the validity of the original problem.

Theorem 4. *Let $\mathcal{M}_k$ be a workflow messaging model, $s \in S$ be a state of $\mathcal{M}_k$, φ be a CTL-FO$^+$ formula in n variables, ν be an ordered t-valuation for some $0 \leq t \leq n$. Let $\widehat{\mathcal{M}}_k^n$ be the freeze workflow messaging model built from $\mathcal{M}_k$, $\widehat{s}_\nu \in \widehat{S}$ be a state of $\widehat{\mathcal{M}}_k^n$ such that $\widehat{L}(\widehat{s}_\nu, \widehat{\nu}_i) = \nu(x_i)$ for all $1 \leq i \leq n$ and $L(s, \alpha) = \widehat{L}(\widehat{s}_\nu, \alpha)$ for every $\alpha \in \mathcal{P} \cup \mathcal{V} \cup \mathcal{I}$. Then $\mathcal{M}_k, s, \nu \models \varphi$ if and only if $\widehat{\mathcal{M}}_k^n, \widehat{s}_\nu \models \omega_t^n(\varphi)$.*

Proof: The proof is done by structural induction on φ.

Base case: the three ground equality testings must be verified.

1) $c_1 = c_2$: Suppose $\mathcal{M}_k, s, \nu \models c_1 = c_2$, where c_1 and c_2 are constants. By Definition 5, then c_1 and c_2 are the same. By equation (1), $\omega_t^n(c_1 = c_2) \equiv c_1 = c_2$. Since c_1 and c_2 are the same, then $c_1 = c_2$ is a tautology and $\widehat{\mathcal{M}}_k^n, \widehat{s}_\nu \models c_1 = c_2$. The case where $\mathcal{M}_k, s, \nu \not\models c_1 = c_2$ is shown in the same way.
2) $x_i = c_1$: Same as above, using equation (2).
3) $x_i = x_j$: Same as above, using equation (3).

Induction step: Suppose the equivalence is respected for all formulæ of length less than ℓ. Let φ be a formula of length

ℓ. We must show that the application of ω_t^n in each possible case for φ preserves the satisfiability of the formula.

1) $\neg\varphi'$: By Definition 5, $\mathcal{M}_k, s, \nu \models \neg\varphi'$ if and only if $\mathcal{M}_k, s, \nu \not\models \varphi'$. By the induction hypothesis, $\mathcal{M}_k, s, \nu \not\models \varphi'$ if and only if $\widehat{\mathcal{M}}_k^n, \widehat{s}_\nu \not\models \omega_t^n(\varphi')$. By the classical semantics of CTL, $\widehat{\mathcal{M}}_k^n, \widehat{s}_\nu \not\models \omega_t^n(\varphi')$ if and only if $\widehat{\mathcal{M}}_k^n, \widehat{s}_\nu \models \neg\omega_t^n(\varphi')$. By equation (4), $\neg\omega_t^n(\varphi') \equiv \omega_t^n(\neg\varphi')$.
2) $\varphi' \vee \psi$: By Definition 5, $\mathcal{M}_k, s, \nu \models \varphi' \vee \psi$ if and only if $\mathcal{M}_k, s, \nu \models \varphi'$ or $\mathcal{M}_k, s, \nu \models \psi$. By the induction hypothesis, $\mathcal{M}_k, s, \nu \models \varphi'$ if and only if $\widehat{\mathcal{M}}_k^n, \widehat{s}_\nu \models \omega_t^n(\varphi')$, and $\mathcal{M}_k, s, \nu \models \psi$ if and only if $\widehat{\mathcal{M}}_k^n, \widehat{s}_\nu \models \omega_t^n(\psi)$. By the classical semantics of CTL, this is the case if and only if $\widehat{\mathcal{M}}_k^n, \widehat{s}_\nu \models \omega_t^n(\varphi') \vee \omega_t^n(\psi)$. By equation (4), $\omega_t^n(\varphi') \vee \omega_t^n(\psi) \equiv \omega_t^n(\varphi' \vee \psi)$.
3) $\mathbf{EX}\,\varphi'$: By Definition 5, $\mathcal{M}_k, s, \nu \models \mathbf{EX}\,\varphi'$ if and only if there exists $s' \in S$ such that $(s, s') \in R$ and $\mathcal{M}_k, s', \nu \models \varphi'$. By Definition 8, $(s, s') \in R$ if and only if there exists a state $\widehat{s}'_\nu \in \widehat{S}$ such that $(\widehat{s}_\nu, \widehat{s}'_\nu) \in \widehat{R}$, $L(s', \alpha) = \widehat{L}(\widehat{s}'_\nu, \alpha)$ for every $\alpha \in \mathcal{P} \cup \mathcal{V} \cup \mathcal{I}$, and $\widehat{L}(\widehat{s}'_\nu, \widehat{\nu}_i) = \widehat{L}(\widehat{s}_\nu, \widehat{\nu}_i)$ for every $\widehat{\nu}_i \in \mathcal{F}$; this last condition entails that $\widehat{\mathcal{M}}_k^n, \widehat{s}'_\nu \models \gamma_t^n$. By the induction hypothesis, we know that $\mathcal{M}_k, s', \nu \models \varphi'$ if and only if $\widehat{\mathcal{M}}_k^n, \widehat{s}'_\nu \models \omega_t^n(\varphi')$. By the classical semantics of CTL, the previous two results are equivalent to $\widehat{\mathcal{M}}_k^n, \widehat{s}'_\nu \models \gamma_t^n \wedge \omega_t^n(\varphi')$. This in turn is equivalent to $\widehat{\mathcal{M}}_k^n, \widehat{s}_\nu \models \mathbf{EX}\,(\gamma_t^n \wedge \omega_t^n(\varphi'))$. Finally, $\mathbf{EX}\,(\gamma_t^n \wedge \omega_t^n(\varphi')) \equiv \omega_t^n(\mathbf{EX}\,\varphi')$ by equation (6).
4) $\mathbf{AF}\,\varphi'$: Suppose $\mathcal{M}_k, s, \nu \models \mathbf{AF}\,\varphi'$. By Definition 5, every path $\pi = ss_1s_2\ldots$, is such that $\mathcal{M}_k, s_i, \nu \models \varphi'$ for some i. Alternatively, this is equivalent to the fact that there is no path $\pi = s_1s_2\ldots$ with $s = s_1$ such that $\mathcal{M}_k, s_i, \nu \not\models \varphi'$ for every $i \geq 1$.
By equation (7), $\omega_t^n(\mathbf{AF}\,\varphi') \equiv \mathbf{A}\,[\gamma_t^n\,\mathbf{U}\,(\neg\gamma_t^n \vee \omega_t^n(\varphi'))]$. By the classical CTL semantics, the formula $\mathbf{A}\,[\gamma_t^n\,\mathbf{U}\,(\neg\gamma_t^n \vee \omega_t^n(\varphi'))]$ if true if and only if for every path $\widehat{\pi} = \widehat{s}_1\widehat{s}_2\ldots$ with $\widehat{s}_\nu = \widehat{s}_1$, there exists an $m \geq 1$ such that $\widehat{M}_k, \widehat{s}_i \models \gamma_t^n$ for every $i < m$, and either $\widehat{M}_k, \widehat{s}_m \not\models \gamma_t^n$ or $\widehat{M}_k, \widehat{s}_m \models \omega_t^n(\varphi')$.
Let $\widehat{\pi} = \widehat{s}_\nu\widehat{s}_1\widehat{s}_2\ldots$ be a path. Define $1 \leq m_1 \leq \infty$ such that $\widehat{M}_k, \widehat{s}_{m_1} \models \omega_t^n(\varphi')$, and $\widehat{M}_k, \widehat{s}_i \not\models \omega_t^n(\varphi')$ for every $i < m_1$. Similarly, define $1 \leq m_2 \leq \infty$ such that $\widehat{M}_k, \widehat{s}_{m_2} \not\models \gamma_t^n$, and $\widehat{M}_k, \widehat{s}_i \models \gamma_t^n$ for every $i < m_2$. Three cases must be considered:
   - $m_1 \leq m_2$ and $m_1 < \infty$: then $\widehat{M}_k, \widehat{s}_i \models \gamma_t^n$ for every $i < m_1$, and $\widehat{M}_k, \widehat{s}_{m_1} \models \gamma_t^n \wedge \omega_t^n(\varphi')$.
     Let $m = m_1$, and the path fulfils the definition.
   - $m_1 > m_2$: then $\widehat{M}_k, \widehat{s}_i \models \gamma_t^n$ for every $i < m_2$, and $\widehat{M}_k, \widehat{s}_{m_2} \not\models \gamma_t^n$. Let $m = m_2$, and the path fulfils the definition.
   - $m_1 = m_2 = \infty$: then $\widehat{\pi}$ is a path $\widehat{s}\widehat{s}_1\widehat{s}_2\ldots$ such that $\widehat{\mathcal{M}}_k^n, \widehat{s}_i \models \gamma_t^n$ and $\widehat{\mathcal{M}}_k^n, \widehat{s}_i \not\models \omega_t^n(\varphi')$ for every $i \geq 1$. By Definition 8, $\widehat{\pi}$ is a path in $\widehat{M}_k$ if and only if there exists a path $\pi = ss_1s_2\ldots$ in $\mathcal{M}_k$ such that, for every $i > 1$, $L(s_i, \alpha) = \widehat{L}(\widehat{s}_i, \alpha)$ for every $\alpha \in \mathcal{P} \cup \mathcal{V} \cup \mathcal{I}$ and $(s_{i-1}, s_i) \in R$. By the induction hypothesis, since $\widehat{\mathcal{M}}_k^n, \widehat{s}_i \not\models \omega_t^n(\varphi')$ for every $i \geq 1$, then $\mathcal{M}_k, s_i, \nu \not\models \varphi'$ for every $i \geq 1$. This contradicts the hypothesis that no such path exists in $\mathcal{M}_k$.

   Therefore, all paths in $\widehat{\mathcal{M}}_k^n$ satisfy the condition, and $\widehat{\mathcal{M}}_k^n, \widehat{s}_\nu \models \omega_t^n(\varphi')$.
   Conversely, suppose $\mathcal{M}_k, s, \nu \not\models \mathbf{AF}\,\varphi'$. By Definition 5, there exists a path $\pi = ss_1s_2\ldots$ in $\widehat{M}_k$ such that $\mathcal{M}_k, s_i, \nu \not\models \varphi'$ for every $i \geq 1$. By Definition 8, π is a path in $\mathcal{M}_k$ if and only if there exists a path $\widehat{\pi} = \widehat{s}_\nu\widehat{s}_1\widehat{s}_2\ldots$ such that:
   - $(\widehat{s}_\nu, \widehat{s}_1) \in \widehat{R}$ and for every $j \geq 1$, $(\widehat{s}_j, \widehat{s}_{j+1}) \in \widehat{R}$
   - for every $j \geq 1$ and every $\alpha \in \mathcal{P} \cup \mathcal{V} \cup \mathcal{I}$, $\widehat{L}(\widehat{s}_j, \alpha) = L(s_j, \alpha)$
   - for every $j \geq 1$ and every $\widehat{\nu}_i \in \mathcal{F}$, $\widehat{L}(\widehat{s}_j, \widehat{\nu}_i) = \nu(x_i)$

   Therefore, for every $i \geq 1$, we have $\widehat{M}_k, \widehat{s}_i \models \gamma_t^n$, and by the induction hypothesis, $\widehat{\mathcal{M}}_k^n, \widehat{s}_i \not\models \omega_t^n(\varphi')$. By the classical CTL semantics, we then have that $\widehat{\mathcal{M}}_k^n, \widehat{s}_\nu \not\models \mathbf{A}\,[\gamma_t^n\,\mathbf{U}\,(\neg\gamma_t^n \vee (\gamma_t^n \wedge \omega_t^n(\varphi')))]$.
5) $\mathbf{E}\,[\varphi'\,\mathbf{U}\,\psi]$: By Definition 5, $\mathcal{M}_k, s, \nu \models \mathbf{E}\,[\varphi'\,\mathbf{U}\,\psi]$ holds if and only if there exists a path $\pi = ss_1s_2\ldots$ and an $m \geq 1$ such that $\mathcal{M}_k, s_i, \nu \models \varphi'$ for all $i < m$ and $\mathcal{M}_k, s_m, \nu \models \psi$. By Definition 8, π is a path in $\mathcal{M}_k$ if and only if there exists a path $\widehat{\pi} = \widehat{s}_\nu\widehat{s}_1\widehat{s}_2\ldots$ such that
   - $(\widehat{s}_\nu, \widehat{s}_1) \in \widehat{R}$ and for every $j \geq 1$, $(\widehat{s}_j, \widehat{s}_{j+1}) \in \widehat{R}$
   - for every $j \geq 1$ and every $\alpha \in \mathcal{P} \cup \mathcal{V} \cup \mathcal{I}$, $\widehat{L}(\widehat{s}_j, \alpha) = L(s_j, \alpha)$
   - for every $j \geq 1$ and every $\widehat{\nu}_i \in \mathcal{F}$, $\widehat{L}(\widehat{s}_j, \widehat{\nu}_i) = \nu(x_i)$

   Therefore, every state $\widehat{s}_i$ along $\widehat{\pi}$ is such that $\widehat{M}_k, \widehat{s}_i \models \gamma_t^n$. Moreover, by the induction hypothesis, $\mathcal{M}_k, s_i, \nu \models \varphi'$ for all $i < m$ if and only if $\widehat{M}_k, \widehat{s}_i \models \omega_t^n(\varphi')$ for all $i < m$, and $\mathcal{M}_k, s_m, \nu \models \psi$ if and only if $\widehat{M}_k, \widehat{s}_m \models \omega_t^n(\psi)$. By the classical CTL semantics, this is equivalent to $\widehat{M}_k, \widehat{s}_\nu \models \mathbf{E}\,[(\gamma_t^n \wedge \omega_t^n(\varphi))\,\mathbf{U}\,(\gamma_t^n \wedge \omega_t^n(\psi))]$, and by equation (8), this in turn is equivalent to $\widehat{M}_k, \widehat{s}_\nu \models \omega_t^n(\mathbf{E}\,[\varphi'\,\mathbf{U}\,\psi])$.
6) $\exists_p x_i : \varphi'$: Since φ is well-named, the quantification of variable x_i entails that all variables $x_1, \ldots, x_{i-1}$ are already defined by ν; hence $i = t + 1$ and $p = \pi_{t+1}$. By Definition 5, $\mathcal{M}_k, s, \nu \models \exists_{\pi_{t+1}} x_{t+1} : \varphi'$ if and only if there exists $a \in Dom_s(\pi_{t+1})$ such that $\mathcal{M}_k, s, \nu[a/x_{t+1}] \models \varphi'$; $\nu[a/x_i]$ is the $(t+1)$-valuation that agrees with the ordered t-valuation ν for all x_i with $1 \leq i \leq t$, and which maps x_{t+1} to a.
   By the induction hypothesis, $\mathcal{M}_k, s, \nu[a/x_{t+1}] \models \varphi'$ holds if and only if $\widehat{\mathcal{M}}_k^n, \widehat{s}_{\nu[a/x_i]} \models \omega_{t+1}^n(\varphi')$, where $\widehat{s}_{\nu[a/x_i]} \in \widehat{S}$ is such that $\widehat{L}(\widehat{s}_{\nu[a/x_i]}, \alpha) = L(s, \alpha)$ for every $\alpha \in \mathcal{P} \cup \mathcal{V} \cup \mathcal{I}$, and $\widehat{L}(\widehat{s}_{\nu[a/x_i]}, \widehat{\nu}_j) = \nu[a/x_i](x_j)$ for every $\widehat{\nu}_j \in \mathcal{F}$; moreover, by definition $\widehat{s}_{\nu[a/x_i]}$ is such that $\widehat{\mathcal{M}}_k^n, \widehat{s}_{\nu[a/x_i]} \models \gamma_{t+1}^n$. By Definition 9, this is true if and only if $(\widehat{s}_\nu, \widehat{s}_{\nu[a/x_{t+1}]}) \in \widehat{R}$. By the classical semantics of CTL, this is true if and only if $\widehat{\mathcal{M}}_k^n, \widehat{s}_\nu \models \mathbf{EX}\,(\gamma_{t+1}^n \wedge \omega_{t+1}^n(\varphi'))$, which by equation (9) is equivalent to $\widehat{\mathcal{M}}_k^n, \widehat{s}_\nu \models \omega_t^n(\exists_p x_{t+1} : \varphi')$.

■

Corollary 1. *Let $\mathcal{M}_k$ be a workflow messaging model, φ be a CTL-FO$^+$ formula in n variables, $\widehat{\mathcal{M}}_k^n$ be the freeze workflow messaging model built from $\mathcal{M}_k$, and $\widehat{\varphi} = \omega_0^n(\varphi)$. Then $\mathcal{M}_k \models \varphi$ if and only if $\widehat{\mathcal{M}}_k^n \models \widehat{\varphi}$.*

Proof: From Definition 7, $s \in I$ if and only if $\widehat{s} \in \widehat{I}$, with $L(s,\alpha) = \widehat{L}(\widehat{s},\alpha)$ for every $\alpha \in \mathcal{P} \cup \mathcal{V} \cup \mathcal{I}$, and $\widehat{L}(\widehat{s},\widehat{\nu}_i) = \#$ for every $\nu_i \in \mathcal{F}$. But then by Theorem 4, $\mathcal{M}_k, s, \nu \models \varphi$ if and only if $\widehat{\mathcal{M}}_k^n, \widehat{s} \models \widehat{\varphi}$, with ν the empty valuation. ■

Contrarily to explicit quantification, the freeze quantification approach does not cause an exponential blow-up of the original formula. The embedding ω is linear: that is, if we denote by $|\varphi|$ the length of a CTL-FO$^+$ formula φ, then $|\omega_0^n(\varphi)| \in O(|\varphi|)$. It suffices to remark that each translation rule consumes at least one symbol of the original CTL-FO$^+$ formula and contributes a fixed number of symbols in the resulting CTL formula.

VII. Experimental Results

We conducted a set of experiments that involved the validation of constraints in the scenarios detailed in Section III. This section shows results intended to compare explicit quantification and freeze quantification.

A. Methodology

The goal of these experiments is twofold: first, show that validating web service constraints can be done using the freeze quantification solution presented in this paper; second, exhibit sample properties for which the explicit quantification approach is inadequate.

We defined a workflow messaging model for both the e-commerce and the UCLP scenarios. We first fixed a domain size n, and then populated this domain with symbolic values with names $a_1, \ldots, a_n$, considered different. All message elements in the workflow messaging model took values from this set. According to the fact that atoms in CTL-FO$^+$ are only equality tests, these values are generic (cf. Section V-A). In both scenarios, the initial description of the service consisted of two parts: a finite-state guarded automaton that represented the control-flow of the service, and the structure (name and number of elements) of each message type sent or received by this service (e.g. loginMessage, concatenateRequest, etc.), as shown in Section III. Each state of the guarded automaton was then attached to one of the message types, thereby forming a workflow messaging model.

A PHP script was then used to generate a NuSMV [4] file, taking as paramters: the description of the workflow messaging model (given as above), the CTL-FO$^+$ formula to validate, the desired arity of the messages k, and the size of the value domains n. The script either produced a standard workflow messaging model with an explicitly quantified CTL formula, or a freeze workflow messaging model with a freeze quantified CTL formula.

Adding freeze variables to an existing workflow messaging model requires minimal modifications, that have been automated. In a NuSMV model, for each added freeze variable quantifying over element of name p, it suffices to add two new lines of code stating that: 1) either the variable keeps its value in the next state or 2) the variable is currently undefined and takes in the next state the value of one of the p elements of the current message. Finally, a single new condition on the transition of the guarded automaton is added: if any freeze variable changes its value in the next transition, the state of the guarded automaton does not change. The resulting model is a freeze workflow messaging model; it was then fed into NuSMV, and its running times and file size were measured.

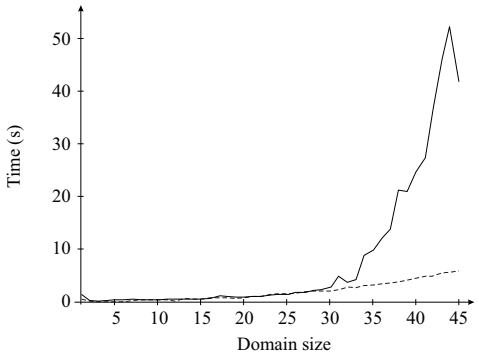


Figure 2. Validation time (in seconds) for Choreography Specification 1 with respect to the size n of the domain, using respectively the explicit quantification (dashed curve) and the freeze quantification approach (solid curve).

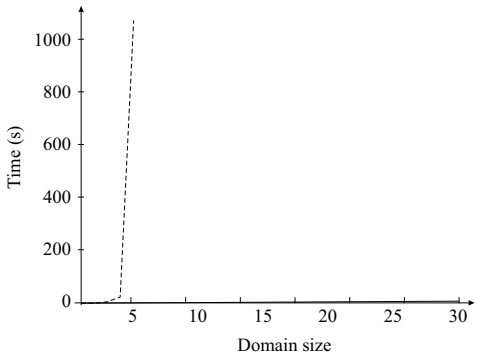


Figure 3. Validation time (in seconds) for Choreography Specification 2 with respect to the size n of the domain, using respectively the explicit quantification (dashed curve) and the freeze quantification approach (solid curve).

B. Results and Discussion

The Figures 2, 3 and 4 present the validation times of the freeze and the explicit quantification approaches for the three formal choreography specifications detailed in Section III, on processes with data domains of size n ranging from 1 to 30 (45 in the case of Figure 2). The formulæ contain respectively 2, 4, and 7 quantifiers. All times have been obtained with NuSMV 2.4.0 on an AMD Athlon 2200+ CPU running under Windows XP (Cygwin). Since NuSMV takes several dozens of seconds only to display the explicitly quantified formulæ, all display from NuSMV was disabled.

We can distinguish two situations. In the case of Property Specification 1 (Figure 2), freeze and explicit quantification are head-to-head until $n = 30$. Explicit quantification then keeps a slower growth than the freeze quantification approach from $n = 30$ to $n = 45$.

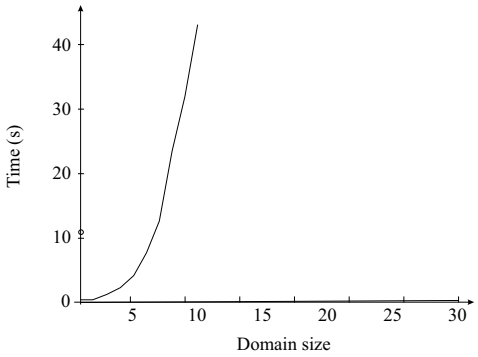


Figure 4. Validation time (in seconds) for Choreography Specification 3 with respect to the size n of the domain, using respectively the explicit quantification (dashed curve) and the freeze quantification approach (solid curve).

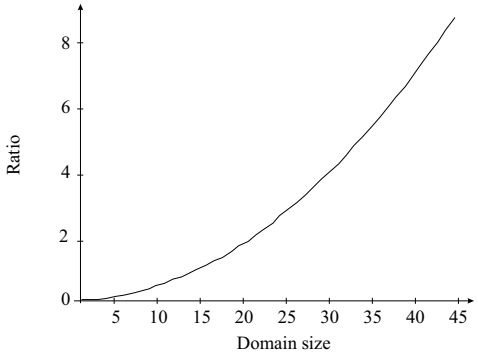


Figure 5. Ratio size of freeze model vs. length of explicit formula for Choreography Specification 1.

In the remaining two figures, this tendency is reversed and freeze quantification performs much better than explicit quantification. For Figure 3, explicit validation times rapidly blow up; NuSMV had to be killed after consuming all available memory for $n = 5$. The last data point is 20 minutes for $n = 4$, while freeze quantification takes less than one second for the same problem. For Figure 3, we could obtain only one data point ($n = 1$) with explicit quantification before it exploded. We stopped evaluating freeze quantification at $n = 10$ when it became clear it outperformed explicit quantification.

We see that explicit quantification performs better with few quantifiers (2 or 3) while freeze quantification becomes more advantageous when the number of quantifiers exceeds that threshold. To highlight the interplay between formula length and system size, we compared the number of states in the *freeze* model the number of symbols in the *explicit* formula. When this ratio increases, as for Choreography Specification 1 (Figure 5), explicit quantification is favored. Otherwise, freeze quantification performs better, as is the case for Choreography Specifications 2 and 3 (Figure 6). An interesting consequence of this observation is that these ratios can be computed beforehand. For any given workflow messaging model and CTL-FO$^+$ property, it is relatively easy to determine functions $f_E(n)$ and $f_F(n)$, returning respectively the size of the explicit CTL formula and the size of the freeze model in terms of the domain size n. It then suffices to study the behaviour of $f_E(n)/f_F(n)$ to determine which method to favor.

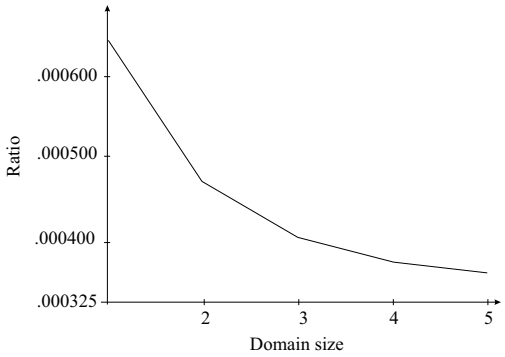


Figure 6. Ratio size of freeze model vs. length of explicit formula for Choreography Specification 2.

In practice, however, explicit quantification fares better only for really simple formulæ, with 3 quantifiers or less. Our examples show that all but one property fits into this category. Therefore, freeze quantification is an important approach to bring a large class of data-aware properties within reach of existing model checking tools.

VIII. Conclusion

In this paper, we have shown how "data-aware" temporal properties can be used to express constraints on the behaviour of a web service composition. These properties enable complex temporal relationships to be expressed, while at the same time allowing full first-order quantification on the content of the messages. We presented real-world scenarios where data-aware properties arise naturally, and showed how existing related work is only partially appropriate for the validation of such properties. To this end, we introduced the logic CTL-FO$^+$, showed its model checking algorithm and studied its complexity. We conclude that model checking data-aware temporal properties is a tractable problem and that any web service model that uses a data-aware workflow, but *propositional* properties cannot efficiently simulate data-awareness. We have demonstrated by empirical testing on processes how a suitable reduction of CTL-FO$^+$ to CTL, using the concept of freeze quantification, can be used to validate them in reasonable time compared to classical approaches.

Acknowledgements

The authors gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada. They thank Jérôme Tremblay and Boubker Ghandour for their technical contribution to this work.

References

[1] G. Meredith and S. Bjorg, "Contracts and types," *Commun. ACM*, vol. 46, no. 10, pp. 41–47, 2003.

[2] S. Parastatidis, J. Webber, S. Woodman, D. Kuo, and P. Greenfield, "SOAP service description language (SSDL)," University of Newcastle, Newcastle upon Tyne, Tech. Rep. CS-TR-899, 2005.

[3] G. J. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley Professional, 2003.

[4] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "NuSMV 2: An opensource tool for symbolic model checking," in *CAV*, ser. Lecture Notes in Computer Science, E. Brinksma and K. G. Larsen, Eds., vol. 2404. Springer, 2002, pp. 359–364.

[5] T. Bultan, X. Fu, R. Hull, and J. Su, "Conversation specification: a new approach to design and analysis of e-service composition," in *WWW*, 2003, pp. 403–410.

[6] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "Model-based analysis of obligations in web service choreography," in *AICT/ICIW*. IEEE Computer Society, 2006, p. 149.

[7] M. Koshkina and F. van Breugel, "Modelling and verifying web service orchestration by means of the concurrency workbench," *ACM SIGSOFT SEN*, vol. 29, no. 5, September 2004.

[8] S. Hinz, K. Schmidt, and C. Stahl, "Transforming BPEL to Petri nets," in *Business Process Management*, W. M. P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, Eds., vol. 3649, 2005, pp. 220–235.

[9] J. M. Zaha, M. Dumas, A. ter Hofstede, A. Barros, and G. Decker, "Service interaction modeling: Bridging global and local views," in *EDOC*. IEEE Computer Society, 2006, pp. 45–55.

[10] A. Deutsch, L. Sui, V. Vianu, and D. Zhou, "Verification of communicating data-driven web services," in *PODS*, S. Vansummeren, Ed. ACM, 2006, pp. 90–99.

[11] R. Kazhamiakin, M. Pistore, and L. Santuari, "Analysis of communication models in web service compositions," in *WWW*, L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, Eds. ACM, 2006, pp. 267–276.

[12] D. Berardi, D. Calvanese, G. D. Giacomo, R. Hull, and M. Mecella, "Automatic composition of transition-based semantic web services with messaging," in *VLDB*, K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-Å. Larson, and B. C. Ooi, Eds. ACM, 2005, pp. 613–624.

[13] Z. Duan, A. J. Bernstein, P. M. Lewis, and S. Lu, "A model for abstract process specification, verification and composition," in *ICSOC*, M. Aiello, M. Aoyama, F. Curbera, and M. P. Papazoglou, Eds. ACM, 2004, pp. 232–241.

[14] N. Lohmann, P. Massuthe, C. Stahl, and D. Weinberg, "Analyzing interacting BPEL processes," in *BPM*, ser. Lecture Notes in Computer Science, S. Dustdar, J. L. Fiadeiro, and A. P. Sheth, Eds., vol. 4102. Springer, 2006, pp. 17–32.

[15] S. Nakajima, "Model-checking of safety and security aspects in web service flows," in *ICWE*, ser. Lecture Notes in Computer Science, N. Koch, P. Fraternali, and M. Wirsing, Eds., vol. 3140. Springer, 2004, pp. 488–501.

[16] K. J. Turner, "Formalising web services," in *FORTE*, ser. Lecture Notes in Computer Science, F. Wang, Ed., vol. 3731. Springer, 2005, pp. 473–488.

[17] C. D. Walton, "Model checking multi-agent web services," p. 8, 2004.

[18] A. Brogi, C. Canal, E. Pimentel, and A. Vallecillo, "Formalizing web service choreographies," *Electr. Notes Theor. Comput. Sci.*, vol. 105, pp. 73–94, 2004.

[19] M. Pistore, M. Roveri, and P. Busetta, "Requirements-driven verification of web services," *Electr. Notes Theor. Comput. Sci.*, vol. 105, pp. 95–108, 2004.

[20] X. Fu, T. Bultan, and J. Su, "Analysis of interacting BPEL web services," in *WWW*, S. I. Feldman, M. Uretsky, M. Najork, and C. E. Wills, Eds. ACM, 2004, pp. 621–630.

[21] ——, "Model checking XML manipulating software," in *ISSTA*, G. S. Avrunin and G. Rothermel, Eds. ACM, 2004, pp. 252–262.

[22] J. Arias-Fisteus, L. S. Fernández, and C. D. Kloos, "Applying model checking to BPEL4WS business collaborations," in *SAC*, H. Haddad, L. M. Liebrock, A. Omicini, and R. L. Wainwright, Eds. ACM, 2005, pp. 826–830.

[23] J. E. Johnson, D. E. Langworthy, L. Lamport, and F. H. Vogt, "Formal specification of a web services protocol," *Electr. Notes Theor. Comput. Sci.*, vol. 105, pp. 147–158, 2004.

[24] A. Deutsch, L. Sui, and V. Vianu, "Specification and verification of data-driven web services," in *PODS*, A. Deutsch, Ed. ACM, 2004, pp. 71–82.

[25] S. Abiteboul, L. Segoufin, and V. Vianu, "Static analysis of active XML systems," in *PODS*, M. Lenzerini and D. Lembo, Eds. ACM, 2008, pp. 221–230.

[26] C. E. Gerede and J. Su, "Specification and verification of artifact behaviors in business process models," in *ICSOC*, ser. Lecture Notes in Computer Science, B. J. Krämer, K.-J. Lin, and P. Narasimhan, Eds., vol. 4749. Springer, 2007, pp. 181–192.

[27] L. Baresi, D. Bianculli, C. Ghezzi, S. Guinea, and P. Spoletini, "Validation of web service compositions," *IET Software*, no. 6, pp. 219–232, 2007.

[28] R. Boutaba, W. Golab, Y. Iraqi, and B. S. Arnaud, "Lightpaths on demand: A web-services-based management system," *IEEE Communications Magazine*, pp. 2–9, July 2004.

[29] S. Hallé, R. Villemaire, O. Cherkaoui, J. Tremblay, and B. Ghandour, "Extending model checking to data-aware temporal properties of web services," in *WS-FM*, ser. Lecture Notes in Computer Science, M. Dumas and R. Heckel, Eds., vol. 4937. Springer, 2007, pp. 31–45.

[30] J. Josephraj, "Web services choreography in practice," 2005, http://www-128.ibm.com/developerworks/webservices/library/ws-choreography/. [Online]. Available: http://www-128.ibm.com/developerworks/webservices/library/ws-choreography/

[31] S. Hallé and R. Villemaire, "Runtime monitoring of message-based workflows with data," in *EDOC*. IEEE Computer Society, 2008, pp. 67–83.

[32] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation, Second Edition*. Addison Wesley, 2000.

[33] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "Model-based verification of web service compositions," in *ASE*. IEEE Computer Society, 2003, pp. 152–163.

[34] C. Ouyang, E. Verbeek, W. M. P. van der Aalst, S. Breutel, M. Dumas, and A. H. M. ter Hofstede, "Formal semantics and analysis of control flow in WS-BPEL," *Sci. Comput. Program.*, vol. 67, no. 2-3, pp. 162–198, 2007.

[35] R. Lucchi and M. Mazzara, "A pi-calculus based semantics for WS-BPEL," *J. Log. Algebr. Program.*, vol. 70, no. 1, pp. 96–118, 2007.

[36] J. Arias-Fisteus, A. Marin, and C. D. Kloos, "VERBUS: A formal model for business process verification," in *IRMA*, May 2004.

[37] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. MIT Press, 2000.

[38] P. Schnoebelen, "The complexity of temporal logic model checking," *Advances in Modal Logic*, vol. 4, pp. 393–436, 2003. [Online]. Available: http://citeseer.ist.psu.edu/schnoebelen03complexity.html

[39] O. Kupferman, "Augmenting branching temporal logics with existential quantification over atomic propositions," in *CAV*, ser. Lecture Notes in Computer Science, P. Wolper, Ed., vol. 939. Springer, 1995, pp. 325–338. [Online]. Available: http://www.cs.huji.ac.il/ ornak/pub.html

[40] A. Rensink, "Model checking quantified computation tree logic," in *CONCUR*, ser. Lecture Notes in Computer Science, C. Baier and H. Hermanns, Eds., vol. 4137. Springer, 2006, pp. 110–125.

[41] W. M. van der Aalst and M. Pesic, "DecSerFlow: Towards a truly declarative service flow language," in *WS-FM*, ser. Lecture Notes in Computer Science, M. Bravetti, M. Núñez, and G. Zavattaro, Eds., vol. 4184. Springer, 2006, pp. 1–23.

[42] M. Brambilla, A. Deutsch, L. Sui, and V. Vianu, "The role of visual tools in a web application design and verification framework: A visual notation for ltl formulae," in *ICWE*, ser. Lecture Notes in Computer Science, D. Lowe and M. Gaedke, Eds., vol. 3579. Springer, 2005, pp. 557–568.

[43] H. Barringer, A. Goldberg, K. Havelund, and K. Sen, "Rule-based runtime verification," in *VMCAI*, ser. Lecture Notes in Computer Science, B. Steffen and G. Levi, Eds., vol. 2937. Springer, 2004, pp. 44–57.

[44] M. R. A. Huth and M. D. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge, England: Cambridge University Press, 2000. [Online]. Available: citeseer.ist.psu.edu/huth99logic.html

[45] T. Ball and S. K. Rajamani, "Boolean programs: A model and process for software analysis," Microsoft Research, Tech. Rep. MSR-TR-2000-14, February 2000.

[46] M. R. Garey and D. S. Johnson, *Computers and intractability, a guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[47] R. Alur and T. A. Henzinger, "A really temporal logic," *J. ACM*, vol. 41, no. 1, pp. 181–204, 1994.

Sylvain Hallé received the BS degree in mathematics from Université Laval in 2002 and the MSc in mathematics and PhD in computer science from Université du Québec à Montréal in 2004 and 2008, respectively. He is currently a postdoctoral research fellow at University of California, Santa Barbara. He received fellowships from the Natural Sciences and Engineering Research Council of Canada (NSERC) in 2005 and Quebec's Research Fund on Nature and Technologies (FQRNT) in 2008. His major research interests include Web applications and formal verification. He is a member of the ACM, the Association for Symbolic Logic, the IEEE, and the IEEE Computer Society. He was co-chair of DDBP 2008, TIME 2008 and DDBP 2009.

Roger Villemaire received the PhD degree from the University of Tübingen in 1988. He was a postdoctoral fellow at McGill University and later at Université du Québec à Montréal (UQAM). He is a professor in the Department of Computer Science at UQAM, which he joined in 1993. His research interests include applications of logic in computer science, in particular formalisms, methods and algorithms which can help to realize reliable computing systems. He was co-chair of TIME 2008 and served on its program committee in 2009. He is a member of the ACM, the Association for Symbolic Logic and the IEEE Computer Society.

Omar Cherkaoui received the PhD degree in computer science from Université de Montréal in 1988. He is a professor in the Department of Computer Science at Université du Québec à Montréal, which he joined in 1984. He has been involved in numerous research partnerships with the industry, including the CANARIE consortium and Cisco Systems. He has co-authored more than 50 peer-reviewed technical publications and books, and two patent disclosures. His research interests include network management and optical networks. He is a member of the IEEE and the IEEE Communications Society. Dr. Cherkaoui is a member of the technical program committees of a dozen conferences, including IM 2003, DSOM 2005, ACON 2006, AICT 2007 and 2008.