Model Checking Unbounded Artifact-Centric Systems

Alessio Lomuscio and Jakub Michaliszyn
Department of Computing, Imperial College London, UK

Abstract

Artifact-centric systems are a recent paradigm for representing and implementing business processes. We present further results on the verification problem of artifact-centric systems specified by means of FO-CTL specifications. While the general problem is known to be undecidable, results in the literature prove decidability for artifact systems with infinite domains under boundedness and conditions such as uniformity. We here follow a different approach and investigate the general case with infinite domains. We show decidability of the model checking problem for the class of artifact-centric systems whose database schemas consist of a single unary relation, and we show that that the problem is undecidable if artifact systems are defined by using one binary relation or two unary relations.

1 Introduction

Artifact-centric systems (ACS) have been put forward as a framework for reasoning about and implementing data-aware business processes (Alonso et al. 2004; Hull 2008; Hull et al. 2011). Artifacts are constructs consisting of *data* and *lifecycles*. The data component is given by means of a relational database, i.e., a set of finite relations with fixed *schema*. The lifecycles describe how the artifacts may interact and evolve over time.

ACS can be programmed via the Guard-Stage-Milestone (GSM) language (Hull et al. 2011). The iHub (Heath III et al. 2013) is a production and execution suite for ACS implemented in GSM. Both GSM and the iHub are designed to help stakeholders encode business interactions intuitively and efficiently.

A problem that naturally arises is whether ACS are correct against specifications. GSMC, a model checking tool for GSM systems, has recently been put forward (Gonzalez, Griesmayer, and Lomuscio 2012) to assess this problem.

While GSMC has shown considerable promise, verifying GSM programs via model checking remains highly problematic. Classical techniques based on model checking (Clarke, Grumberg, and Peled 1999) are typically insufficient as they normally tackle finite-state systems. Yet artifacts exhibit infinitely many different possible configurations. It is therefore important to establish at a fundamental level under what conditions an artifact-system can actually be verified.

The verification problem for ACS specified by quantified temporal specifications is known to be undecidable (Deutsch et al. 2009; Hariri et al. 2011; Belardinelli, Lomuscio, and Patrizi 2011) and considerable research has gone into the exploration of decidable fragments.

Contribution. In this paper we explore novel boundaries in the decidability conditions for the model checking problem of ACS against quantified, temporal specifications. While much of the recent work has focused on the identification of relational properties on the generated models for complete and decidable abstraction procedures, an alternative avenue of investigation consists in considering classes of specifications for the artifacts' transitions. To this end, we here define the logic CARL which we use to model the artifacts' lifecycles. By using CARL to model the artifacts' evolutions, we can express Boolean combinations of the basic relational properties: "**c**heck that the relation is empty, "**a**dd one arbitrary element to the relation", "**r**emove one arbitrary element", and "**l**eave the relation unchanged". We show that model checking ACS described via CARL against reachability specifications is undecidable thereby providing further insights on the difficulty of verifying data-aware systems.

We then focus on restricting the database schema appearing in the ACS. We show that the model checking problem remains undecidable if the schema contain either one relation of arity greater than one, or at least two arbitrary relations. Our undecidability proofs involve the encoding of two-counter machines (Minsky 1967).

Finally, we show that the model checking problem is decidable if we consider only artifacts whose schema consists of a single unary relation and sentence-atomic temporal specifications. We prove this result in three steps. Firstly, we study the expressive power of first-order logic over unary relations using Ehrenfeucht-Fraisse games. Then, we show how to transform an ACS and a specification into an infinite Kripke structure and a CTL specification. Finally, we prove that the resulting infinite Kripke structure can be encoded as a pushdown system. We then define a CTL* specification that the pushdown system satisfies if and only if the ACS satisfies the original FO-CTL specification. Since the former problem is decidable (Bozzelli 2007), we conclude that the latter is decidable as well.

Related Work. Previous research on verifying artifact-centric systems has focused on abstraction techniques where semantical conditions such as "uniformity" and "weak acyclicity" are shown to guarantee decidability under the assumption that systems are bounded, i.e., either the states or the runs do not contain more than a given number of predicates (Belardinelli, Lomuscio, and Patrizi 2012b; Hariri et al. 2013). Differently from these approaches, the investigation here presented makes no assumption on the boundedness of the artifact-systems to be studied.

Model checking artifact-centric systems against a quantified version of LTL was studied in (Deutsch et al. 2009). It was shown that the problem is generally undecidable, but becomes decidable if one considers *guarded artifact systems* and *guarded specifications*, a form of quantification where no variables appear free in a temporal context. A PSPACE model checking algorithm for fixed-arity schemas is also shown in (Deutsch et al. 2009). While the results presented are positive, compared to the classes of artifacts studied here, the expressivity of the fragment studied is limited.

More recently (Hariri et al. 2013) presented results on the model checking problem against specifications given in the first-order extension of the μ-calculus. The problem is undecidable in this setting; however, it is shown that by adding additional restrictions, including forbidding fresh, unbounded data along a run, the problem is decidable. Also in that line, and differently from this paper, systems are assumed to be bounded and sufficient conditions based on acyclicity for boundedness are established.

While all results above focus on complete procedures, in recent work (Belardinelli and Lomuscio 2013) partial model checking procedures against the universal fragment of FO-CTL have been explored. Differently from (Belardinelli, Lomuscio, and Patrizi 2012b) these are given for bounded but possibly non-uniform artifact-centric systems. Conversely, the results established in this paper concern unbounded, but uniform systems.

Scheme of the paper. In Section 2 we provide the background on ACS and relevant concepts, and we define the model checking problem. We observe that the problem is generally undecidable. In Section 3 we investigate the implications of restricting the database schemas and the expressive power of the language used to model the artifacts' lifecycles. We show that under various restrictions the problem remains undecidable. In Section 4 we characterise a scenario in which the model checking problem is decidable and explore its complexity. Taken together the results in the two sections show that the conditions we identify can be interpreted as being maximal for decidability. We conclude in Section 5.

2 Verifying Artifact Systems

Artifacts are operational models of *business process* (Cohn and Hull 2009). Each artifact is composed by two parts: a *data model*, which captures a fragment of the structure of the information relevant to the process, and a *lifecycle model*, which is a specification of their evolution. In this paper we follow the formalisation proposed in (Hariri et al. 2011) and define artifact systems by means of a (relational) database and a set of artifact transitions, which account for the artifact data models and the artifact lifecycles, respectively.

Definition 1 (Database schema). *A* database schema *is a set* $\mathcal{D} = \{P_1/a_1, \dots, P_n/a_n\}$ *of* relation (or predicate) symbols P_i*, each associated with its arity* $a_i \in \mathbb{N}$.

Definition 2 (Database interpretation). *Given a database schema* $\mathcal{D}$*, a* $\mathcal{D}$-interpretation (or $\mathcal{D}$-instance) over an countable interpretation domain U *is a mapping* D *associating each relation symbol* P_i *with a* finite a_i*-ary relation over* U*, i.e.,* $D(P_i) \subseteq U^{a_i}$.

The set of all $\mathcal{D}$-interpretations over a given domain U is denoted by $\mathcal{I}_{\mathcal{D}}(U)$. The *active domain* of D, *adom*(D), is the set of all U-elements occurring in some tuple of some predicate interpretation $D(P_i)$.

2.1 Artifact systems

To model transitions between states of the underlying databases, we use *unprimed* and *primed* relational symbols from $\mathcal{D}$, that refer to relations in the current and the successor state, respectively. Intuitively, given two states (i.e., two $\mathcal{D}$-interpretations) D and D', the operator $\oplus$ constructs a new "joint" interpretation $D \oplus D'$, interpreting unprimed relational symbols in D, and primed in D'.

As we study artifacts systems whose transitions are described by various logics, below we introduce the notion of an *artifact system over a logic* $\mathcal{L}_{\mathcal{D}}$.

Definition 3 ($\mathcal{L}_{\mathcal{D}}$-artifact system). *An* $\mathcal{L}_{\mathcal{D}}$-artifact system *is a tuple* $\mathcal{S} = \langle \mathcal{D}, U, D_0, \mathbb{T} \rangle$*, where:*

- $\mathcal{D} = \{P_1/a_1, \dots, P_n/a_n\}$ *is a database schema;*
- U *is a countable interpretation domain;*
- D_0 *is an* initial database instance*;*
- $\mathbb{T}$ *is a finite set of sentences of the logic* $\mathcal{L}_{\mathcal{D}}$*, called* artifact transitions.

In the definition above we assume that the sentences of the logic $\mathcal{L}_{\mathcal{D}}$ state properties of $D \oplus D'$, i.e., the notion "$D \oplus D' \models_{\mathcal{L}_{\mathcal{D}}} \Phi$" is defined for every sentence Φ of $\mathcal{L}_{\mathcal{D}}$.

The semantics of an artifact system is given in terms of its possible executions, captured by a Kripke structure, whose states are instances of the database schema and whose transitions correspond to executions of artifact transitions.

Definition 4 (Kripke structures). *A Kripke structure is a tuple* $\mathcal{K} = \langle \Sigma, D_0, \tau, \pi \rangle$*, where* Σ *is the* set of states*;* $D_0 \in \Sigma$ *is the* initial state*;* $\tau \subseteq \Sigma \times \Sigma$ *is the* transition relation*;* π *is the labelling function.*

For convenience, we assume a single initial state, but all the results presented below also hold if we allow any finite number of initial states. When the labelling function is irrelevant, we will drop it and represent a Kripke structure as a triple $\mathcal{K} = \langle \Sigma, D_0, \tau \rangle$.

Note that in the definition above we include states not reachable from D_0 by τ. This is not the case below.

Definition 5 (Model of an artifact system). *Given an artifact system* $\mathcal{S} = \langle \mathcal{D}, U, D_0, \mathbb{T} \rangle$ *over* $\mathcal{L}_{\mathcal{D}}$*, the* model of $\mathcal{S}$ *is the Kripke structure* $\mathcal{K}_{\mathcal{S}} = \langle \Sigma, D_0, \tau \rangle$*, where:*

- $\Sigma \subseteq \mathcal{I}_{\mathcal{D}}(U)$ *is the* set of states *reachable from* D_0 *using* τ*;*
- $D_0 \in \Sigma$ *is the* initial state*;*
- τ *is the* transition relation *such that* $\tau(D, D')$ *for some artifact transition* $\Phi \in \mathbb{T}$, $D \oplus D' \models_{\mathcal{L}_{\mathcal{D}}} \Phi$.

We denote the unique model of an artifact system $\mathcal{S}$ by $\mathcal{K}_{\mathcal{S}}$.

Following (Belardinelli, Lomuscio, and Patrizi 2011), we define *runs* on a Kripke structure.

Definition 6 ($\mathcal{K}$-runs)**.** *Given an artifact system* S *and its model* $\mathcal{K} = \langle \Sigma, D_0, \tau \rangle$*, a* $\mathcal{K}$-run r *from a* $\mathcal{K}$*-state* $D \in \Sigma$ *is an infinite sequence of* $\mathcal{K}$*-states* $r = D^0 \to D^1 \to \cdots$ *such that* $D^0 = D$ *and* $\tau(D^i, D^{i+1})$*, for* $i \geq 0$*. For every run* r *and* $i \geq 0$*, we define* $r(i)$ *as* D^i.

Note that, unlike much of the literature on the subject, we do not require the runs to be bounded.

Also note that we consider only infinite, countable interpretation domains. To make the notation lighter, we assume without loss of generality that the interpretation domain is always the set of naturals $\mathbb{N}$.

2.2 Model checking artifact systems

In the following we use the logic $FO^V_{\mathcal{D},C}$ to define the transitions of artifact-centric systems. The logic $FO^V_{\mathcal{D},C}$ is the fragment of first-order logic in which formulae are built by using predicates from $\mathcal{D} \oplus \mathcal{D}'$, constants from C, and variables from V.

Definition 7 (Logic $FO^V_{\mathcal{D},C}$)**.** *Given a database schema* $\mathcal{D} = \{P_1/a_1, \ldots, P_n/a_n\}$*, the language of the logic* $FO^V_{\mathcal{D},C}$ *is defined by the following BNF:*

$$\Phi ::= t = t' \mid P_i(\vec{t}) \mid P'_i(\vec{t}) \mid \neg\Phi \mid \Phi \vee \Phi \mid \forall x \Phi,$$

where $t, \bar{t} \in C \cup V$ *and* $\vec{t}$ *is an* a_i*-tuple of elements from* $C \cup V$.

We use parentheses as standard when required.

Let $* = \{x, y, z, x_1, \ldots\}$ be a countable set of variables and $\mathbf{2} = \{x, y\}$. We consider the following logics: $FO^*_{\mathcal{D},\mathbb{N}}$, $FO^*_{\mathcal{D},\emptyset}$, and $FO^{\mathbf{2}}_{\mathcal{D},\emptyset}$. We denote formulae in these logics with Greek upper case letters Φ, Ψ, while specifications against which systems are checked will be denoted by Greek lower case letters φ, ψ.

We use the standard abbreviations $\exists$, $\top$, $\bot$, $\wedge$, $\Rightarrow$, and $\neq$. Free and bound variables are defined as standard. A *sentence* is a formula with no free variables.

A *U-assignment* is a total function $\sigma : V \mapsto U$. For technical convenience, we assume that every U-assignment σ is extended to C and is the identity on it, i.e., $\forall c \in C, \sigma(c) = c$. Given an assignment σ, we denote by σ^x_u the U-assignment s.t. $\sigma^x_u(x) = u$ and $\sigma^x_u(\overline{x}) = \sigma(\overline{x})$, for every $\overline{x} \in V$ s.t. $\overline{x} \neq x$.

We adjust the standard active-domain semantics used in the artifact systems literature (Belardinelli, Lomuscio, and Patrizi 2011; 2012a) to the setting here defined.

Definition 8 (Active-Domain Semantics for FO-formulae)**.** *Given a database schema* $\mathcal{D}$*, two* $\mathcal{D}$*-interpretations* D, D' *over* $\mathbb{N}$*, an* $\mathbb{N}$*-assignment* σ*, and a FO-formula* $\Phi \in FO^*_{\mathcal{D},C}$ *over* $\mathcal{D}$*, we inductively define whether* $D \oplus D'$ *satisfies* Φ *under* σ*, written* $(D \oplus D', \sigma) \models \Phi$*, as follows:*

$$
\begin{array}{ll}
(D \oplus D', \sigma) \models t = \bar{t} & \textit{iff } \sigma(t) = \sigma(\bar{t}) \\
(D \oplus D', \sigma) \models P_i(\vec{t}) & \textit{iff } \langle \sigma(t_1), \ldots, \sigma(t_\ell) \rangle \in D(P_i) \\
(D \oplus D', \sigma) \models P'_i(\vec{t}) & \textit{iff } \langle \sigma(t_1), \ldots, \sigma(t_\ell) \rangle \in D'(P_i) \\
(D \oplus D', \sigma) \models \neg\Phi & \textit{iff } (D \oplus D', \sigma) \not\models \Phi \\
(D \oplus D', \sigma) \models \Phi \vee \Psi & \textit{iff } (D \oplus D', \sigma) \models \Phi \\
& \quad \textit{or } (D \oplus D', \sigma) \models \Psi \\
(D \oplus D', \sigma) \models \forall x \Phi & \textit{iff for every } u \in \textit{adom}(D) \\
& \quad (D \oplus D', \sigma^x_u) \models \Phi
\end{array}
$$

A formula Φ *is* true *in* $D \oplus D'$*, written* $D \oplus D' \models \Phi$*, iff* $(D \oplus D', \sigma) \models \Phi$ *for all* $\mathbb{N}$*-assignments* σ.

Notice that two constants map the same element iff they are the same constant.

We focus on the problem of verifying an artifact system against a temporal specification of interest. We adjust the notion of sentence-atomic FO-CTL (Belardinelli, Lomuscio, and Patrizi 2011) to the case of logic $FO^*_{\mathcal{D},\mathbb{N}}$.

Definition 9 (FO$\underline{^{sa}}$CTL formulae)**.** *Given an artifact system* $\mathcal{S} = \langle \mathcal{D}, \mathbb{N}, D_0, \mathbb{T} \rangle$*, the language of* sentence-atomic *FO-CTL (denoted by FO$\underline{^{sa}}$CTL) formulae over* $\mathcal{S}$ *is inductively defined as follows:*

$$\varphi ::= \Phi_P \mid \neg\varphi \mid \varphi \vee \varphi \mid AX\varphi \mid A\varphi\mathcal{U}\varphi \mid E\varphi\mathcal{U}\varphi$$

where Φ_P *is an* $FO^*_{\mathcal{D},\mathbb{N}}$ *sentence where no primed terms nor primed predicates appear.*

Note from above that while we use the $FO^*_{\mathcal{D},\mathbb{N}}$ logic as the building block for artifacts' descriptions we do not use primed variables in the specifications.

The notions of free and bound variables can be extended in the obvious way to FO$\underline{^{sa}}$CTL. We use the standard abbreviations $EX\varphi \equiv \neg AX\neg\varphi$, $AF\varphi \equiv A\top\mathcal{U}\varphi$, $AG\varphi \equiv \neg E\top\mathcal{U}\neg\varphi$, $EF\varphi \equiv E\top\mathcal{U}\varphi$, and $EG\varphi \equiv \neg A\top\mathcal{U}\neg\varphi$.

For a system S, the semantics of FO$\underline{^{sa}}$CTL formulae is provided in terms of its model $\mathcal{K}_{\mathcal{S}}$.

Definition 10 (Satisfaction for FO$\underline{^{sa}}$CTL)**.** *Consider a system* $\mathcal{S}$ *and its model* $\mathcal{K}_{\mathcal{S}}$*, a formula* φ *of FO$\underline{^{sa}}$CTL and a* $\mathcal{K}_{\mathcal{S}}$*-state* $D \in \Sigma$*, the satisfaction relation* $\models$ *is inductively defined as follows:*

$$
\begin{array}{ll}
(\mathcal{K}_{\mathcal{S}}, D) \models \Phi_P & \textit{iff } D \models \Phi_P \\
(\mathcal{K}_{\mathcal{S}}, D) \models \neg\varphi & \textit{iff } (\mathcal{K}_{\mathcal{S}}, D) \not\models \varphi \\
(\mathcal{K}_{\mathcal{S}}, D) \models \varphi \vee \psi & \textit{iff } (\mathcal{K}_{\mathcal{S}}, D) \models \varphi \textit{ or } (\mathcal{K}_{\mathcal{S}}, D) \models \psi \\
(\mathcal{K}_{\mathcal{S}}, D) \models AX\varphi & \textit{iff for all } \mathcal{K}_{\mathcal{S}}\textit{-runs } r \textit{ s.t. } r(0) = D, \\
& \quad (\mathcal{K}_{\mathcal{S}}, r(1)) \models \varphi \\
(\mathcal{K}_{\mathcal{S}}, D) \models A\varphi\mathcal{U}\psi & \textit{iff for all } \mathcal{K}_{\mathcal{S}}\textit{-runs } r \textit{ s.t. } r(0) = D, \\
& \quad \exists k \geq 0 \textit{ s.t. } (\mathcal{K}_{\mathcal{S}}, r(k)) \models \psi \textit{ and} \\
& \quad \forall j \textit{ s.t. } 0 \leq j < k, (\mathcal{K}_{\mathcal{S}}, r(j)) \models \varphi \\
(\mathcal{K}_{\mathcal{S}}, D) \models E\varphi\mathcal{U}\psi & \textit{iff for some } \mathcal{K}_{\mathcal{S}}\textit{-run } r, r(0) = D, \\
& \quad \exists k \geq 0 \textit{ s.t. } (\mathcal{K}_{\mathcal{S}}, r(k)) \models \psi\textit{, and} \\
& \quad \forall j \textit{ s.t. } 0 \leq j < k, (\mathcal{K}_{\mathcal{S}}, r(j)) \models \varphi
\end{array}
$$

A formula φ is true *in $\mathcal{K}_\mathcal{S}$, written $\mathcal{K}_\mathcal{S} \models \varphi$, iff $(\mathcal{K}_\mathcal{S}, D_0) \models \varphi$. We say that $\mathcal{S}$ satisfies φ, written $\mathcal{S} \models \varphi$, iff $\mathcal{K}_\mathcal{S} \models \varphi$.*

Observe that satisfaction of $FO^*_{\mathcal{D},\mathbb{N}}$ is defined on joins $D \oplus D'$ as it deals with primed and unprimed predicates. Here we write $D \models \Phi_P$ because we only need to interpret only the unprimed predicates.

In the rest of this paper we focus on the following problem.

Definition 11 (Model checking problem)**.** *Given a logic $\mathcal{L}_\mathcal{D}$, an $\mathcal{L}_\mathcal{D}$-artifact system $\mathcal{S}$ and a FO$\underline{^{sa}}$ CTL specification φ, the* model checking problem *involves establishing whether $\mathcal{K}_\mathcal{S} \models \varphi$.*

Observe that the input can be given in finite terms, as the description of an $\mathcal{L}_\mathcal{D}$-artifact system is finite; so the problem is well-defined. This problem can be shown to be undecidable in the general case.

Theorem 12. *Model checking an arbitrary $FO^*_{\mathcal{D},\mathbb{N}}$-artifact system against a FO$\underline{^{sa}}$ CTL specification is undecidable.*

Proof idea. The proof in (Belardinelli, Lomuscio, and Patrizi 2011) can be adapted to this case. □

In the following we will explore the decidability for restrictions of the problem above.

3 Model Checking $CARL_{\mathcal{D},\mathbb{N}}$–Artifact Systems

One possibility to obtain decidability is to consider a weak form of quantification when writing artifacts' descriptions. For example we could consider the two variable fragment of first-order logic (Grädel, Kolaitis, and Vardi 1997) or the guarded fragment (Grädel 1999).

In this paper we follow a different approach and address the question of whether one can choose a reasonable logic $\mathcal{L}_\mathcal{D}$ in the artifacts' descriptions so that model checking $\mathcal{L}_\mathcal{D}$-artifact systems is decidable.

First, we identify some essential properties of the artifact systems. Then, we define the logic capable of expressing only Boolean combinations of such properties. Finally, we show that in most cases model checking artifact systems defined using this logic is undecidable.

In the proofs that follow, we often use the FO$\underline{^{sa}}$ CTL specification $EG\top$. This is intended to show that the negative results we present cannot be overcome by adopting a weaker specification language.

3.1 The logic $CARL_{\mathcal{D},\mathbb{N}}$

The logic $CARL_{\mathcal{D},\mathbb{N}}$ describes properties of unary relations only. $CARL_{\mathcal{D},\mathbb{N}}$ expresses Boolean combinations of the following properties of an unary relation P:

- P contains a constant c;
- P is empty;
- One element is added to P, i.e., $P' = P \cup \{x\}$ for some $x \notin P$;
- One element is removed from P, i.e., $P = P' \cup \{x\}$ for some $x \notin P'$;
- P does not change, i.e., $P = P'$.

These properties are very basic and natural, but are sufficient to express interesting properties for artifact systems. However, as we see below, even the properties above already induce undecidability of the model checking problem.

Definition 13 (Logic $CARL_{\mathcal{D},C}$)**.** *Given a database schema $\mathcal{D} = \{P_1/1, \dots, P_n/1\}$, the language of the logic $CARL_{\mathcal{D},C}$ is inductively defined by the following BNF:*

$$\begin{aligned}\Phi ::= \quad & \neg\Phi \mid \Phi \vee \Phi \mid P_i(c) \mid P'_i(c) \mid \textsc{Empty}(P_i) \mid \\ & \textsc{Equal}(P_i) \mid \textsc{Insert}(P_i) \mid \textsc{Delete}(P_i)\end{aligned}$$

where c is an element of the set of constants C.

Recall that $P_i(c)$ refers to the relation P_i *before* the transition, and $P'_i(c)$ refers to the relation *after* the transition. For convenience the formal semantics of $CARL_{\mathcal{D},C}$ is given by translating it into $FO^2_{\mathcal{D},\mathbb{N}}$.

Definition 14 (Semantics of $CARL_{\mathcal{D},\mathbb{N}}$)**.** *We define a translation of $\mathfrak{T}$ to $FO^2_{\mathcal{D},\mathbb{N}}$ as follows.*

$$\begin{aligned}
\mathfrak{T}(\neg\Phi) &= \neg\mathfrak{T}(\Phi)\\
\mathfrak{T}(\Phi_1 \vee \Phi_2) &= \mathfrak{T}(\Phi_1) \vee \mathfrak{T}(\Phi_2)\\
\mathfrak{T}(P_i(c)) &= P_i(c)\\
\mathfrak{T}(P'_i(c)) &= P'_i(c)\\
\mathfrak{T}(\textsc{Empty}(P_i)) &= \forall x.\neg P_i(x)\\
\mathfrak{T}(\textsc{Equal}(P_i)) &= \forall x.P'_i(x) \Leftrightarrow P_i(x)\\
\mathfrak{T}(\textsc{Insert}(P_i)) &= \exists x.\neg P_i(x) \wedge P'_i(x)\\
&\quad \wedge \forall y.y \neq x \Rightarrow (P_i(y) \Leftrightarrow P'_i(y))\\
\mathfrak{T}(\textsc{Delete}(P_i)) &= \exists x.P_i(x) \wedge \neg P'_i(x)\\
&\quad \wedge \forall y.y \neq x \Rightarrow (P_i(y) \Leftrightarrow P'_i(y))
\end{aligned}$$

Then, the semantics of $CARL_{\mathcal{D},\mathbb{N}}$ is defined using the semantics of $FO^*_{\mathcal{D},\mathbb{N}}$ by considering $(D \oplus D', \sigma) \models \Phi$ iff $(D \oplus D', \sigma) \models \mathfrak{T}(\Phi)$.

3.2 Undecidability of model checking $CARL_{\mathcal{D},\mathbb{N}}$-artifacts systems

A key result of this section is the following theorem.

Theorem 15. *For any database schema $\mathcal{D}$ that contains two unary relation symbols, model checking $CARL_{\mathcal{D},\mathbb{N}}$-artifact systems against FO$\underline{^{sa}}$ CTL is undecidable.*

The proof uses two-counter finite automata (Minsky machines) defined as follows.

Definition 16. *A* two-counter automaton *is a tuple $\mathcal{A} = \langle Q, q_0, q_f, R\rangle$, where $Q = \{q_0, q_1, \dots, q_f\}$, q_0 is an initial state, q_f is a final state, and $R : Q \times \{\top,\bot\} \times \{\top,\bot\} \times Q \times \{-1, 0, 1\} \times \{-1, 0, 1\}$ is the transition relation.*

A *configuration* for a two-counter automata is a triple (q_i, n, m) where $q_i \in Q$ and $n, m \in \mathbb{N}$. Let $Empty(x)$ be equal $\top$ if $x = 0$ and $\bot$ otherwise. We define a relation $\rightarrow^R$ such that for two configuration $\beta_1 = (q_i, n, m)$, $\beta_2 = (q_j, n', m')$ we have $\beta_1 \rightarrow^R \beta_2$ iff we have $(q_i, Empty(n), Empty(m), q_j, n' - n, m' - m) \in R$. A *run* is a possibly infinite sequence of configurations such that for all consecutive configurations β_1, β_2 we have $\beta_1 \rightarrow^R \beta_2$.

A run is *accepting* if it contains a configuration containing q_f.

The following lemma follows from (Minsky 1967).

Lemma 17. *The problem whether there exists an infinite run of a given two-counter automaton is undecidable.*

We can now we prove Theorem 15.

Proof. For a given two-counter automaton $\mathcal{A} = \langle Q, q_0, q_f, R\rangle$, we show how to define a $CARL_{\mathcal{D},\mathbb{N}}$-artifact systems $S^{\mathcal{A}}$ and an FO$\overset{sa}{-}$CTL formula φ such that $S^{\mathcal{A}} \models \varphi$ iff $\mathcal{A}$ has an infinite run. Firstly, we construct a system with tree relations, and then we show how one of the relations can be removed.

Three relations. Let $S^{\mathcal{A}} = \langle \mathcal{D}^{\mathcal{A}}, \mathbb{N}, D_0^{\mathcal{A}}, \mathbb{T}^{\mathcal{A}}\rangle$, where $\mathcal{D}^{\mathcal{A}} = \{P_1/1, P_2/1, P_3/1\}$, $D_0^{\mathcal{A}}$ be such that P_1 and P_2 are empty and P_3 is a singleton relation containing 0. Our aim is to define the set $\mathbb{T}^{\mathcal{A}}$ so that each database interpretation $D = \{P_1^D, P_2^D, P_3^D\}$ in the model $\mathcal{K}_{S^{\mathcal{A}}}$ of $S^{\mathcal{A}}$ satisfies $|P_3^D \cap \{0,\dots,f\}| = 1$ and represents a configuration $(q_i, |P_1^D|, |P_2^D|)$ where $q_i \in P_3^D$. In other words, the size of P_1^D/P_2^D represents the value of the first/second counter and P_3^D contains the state.

Let $empty_{\top}(R) = \textsc{Empty}(R)$ and $empty_{\bot}(R) = \neg\textsc{Empty}(R)$. Let $modify_{-1}(R) = \textsc{Delete}(R)$, $modify_0(R) = \textsc{Equal}(R)$, and $modify_{+1}(R) = \textsc{Insert}(R)$.

For each tuple $t = (q_i, \phi_1, \phi_2, q_j, \varDelta_1, \varDelta_2) \in R$, we define an artifact transition α_t as:

$$P_3(i) \wedge empty_{\phi_1}(P_1) \wedge empty_{\phi_2}(P_2) \wedge P_3'(j)$$
$$\wedge \bigwedge_{k\neq j, k\leq f} \neg P_3(k) \wedge modify_{\varDelta_1}(P_1) \wedge modify_{\varDelta_2}(P_2)$$

Finally, let $\mathbb{T}^{\mathcal{A}}$ be the set of all α_t. It can be observed that paths in $\mathcal{K}_{S^{\mathcal{A}}}$ represent the computations of $\mathcal{A}$. Therefore, there is an infinite path in $\mathcal{K}_{S^{\mathcal{A}}}$ if and only if there is an infinite run in of $\mathcal{A}$. The existence of an infinite path corresponds to the truth of the formula $EG\top$.

Two relations. We show that we do not need a relation to store the states. The *configuration description* of a configuration (q, n_1, n_2) of $\mathcal{A}$ is a triple (q, v_1, v_2) for $v_i \in \{\top, \bot, ?\}$ s.t. if $v_i = \top$ then $n_i = 0$, and if $v_i = \bot$ then $n_i > 0$. The *transition description* is a triple $(q, \varDelta_1, \varDelta_2)$ for $\varDelta_i \in \{-1, 0, 1\}$ that is intended to represent the artifact transition "change the value of the i-th counter by $\varDelta_i$ and change the state to q".

Let $\mathcal{Z}_c = \{0, 1, \dots, 9|Q| - 1\}$ and $\mathcal{Z}_a = \{9|Q|, \dots, 18|Q| - 1\}$. We choose an arbitrary bijection function $encode_c : Q \times \{\bot, \top, ?\} \times \{\bot, \top, ?\} \to \mathcal{Z}_c$ to encode the configuration descriptions as numbers and $encode_a : Q \times \{-1, 0, +1\} \times \{-1, 0, +1\} \to \mathcal{Z}_a$ to encode transition descriptions. Below we identify descriptions with their encodings.

To simplify the proof, we describe the set $\mathbb{T}$ informally.

For $X \subseteq \mathcal{Z}_a \cup \mathcal{Z}_c$, let $exactly(X, R)$ be a formula stating that $R \cap (\mathcal{Z}_a \cup \mathcal{Z}_c) = X$. For $R \in \{P_1, P_1', P_2, P_2'\}$, such a formula can be easily expressed in $CARL_{\mathcal{D},\mathbb{N}}$.

A configuration $\beta = (q_i, n, m)$ will be encoded as a database interpretation $D^\beta = (P_1^\beta, P_2^\beta)$ such that $exactly(\{(q, ?, ?)\}, P_1)$, $exactly(\emptyset, P_2)$, $|P_1^\beta| = n + 1$ and $|P_2^\beta| = m$. So, the initial interpretation of $\mathcal{S}^{\mathcal{A}}$ will be s.t. $P_1 = \{(q_0, ?, ?)\}$ and $P_2 = \emptyset$.

The update of a configuration is performed in several steps by using the rules described below. We will denote transitions informally; e.g., "add n to P_1" stands for $\neg P_1(n) \wedge P_1'(n) \wedge \textsc{Insert}(P_1)$, etc. The rules below should be defined for all appropriate $q, q' \in Q$, $e_1, e_2 \in \{\bot, \top\}$ and $\varDelta_1, \varDelta_2 \in \{-1, 0, 1\}$.

We use the following example to illustrate the encoding. Let $\nu = (q_1, 0, 1)$, $\nu' = (q_2, 1, 1)$, be such $\nu \to^R \nu'$ and assume that $|Q| = 5$, i.e. $\mathcal{Z}$ contains only numbers less than 90. Assume that ν is encoded as $P_1 = \{(q_1, ?, ?)\}, P_2 = \{95\}$.

1. If $P_1((q, ?, ?))$, then if $\textsc{Empty}(P_2)$, add $(q, ?, \top)$ to P_2, otherwise add $(q, ?, \bot)$ to P_2. In both cases, remove $(q, ?, ?)$ from P_1.
   In the example, P_1 and P_2 change as follows: $P_1 = \{\}, P_2 = \{95, (q_1, ?, \top)\}$.
2. If $\textsc{Empty}(P_1)$ and $P_2((q, ?, e_2))$, add $(q, \top, e_2)$ to P_1, otherwise add $(q, \bot, e_2)$ to P_1. In both cases, remove $(q, ?, \bot)$ from P_2.
   In the example, P_1 and P_2 change as follows: $P_1 = \{(q, \top, \bot)\}, P_2 = \{95\}$.
3. If $P_1((q, e_1, e_2))$ and $(q, e_1, e_1, q', \varDelta_1, \varDelta_2) \in R$, then add $(q', \varDelta_1, \varDelta_2)$ to P_1.
   In the example, P_1 and P_2 change as follows: $P_1 = \{(q, \top, \bot), (q', +1, 0)\}, P_2 = \{95\}$.
4. If $P_1((q', \varDelta_1, \varDelta_2))$ and $P_1((q, e_1, e_2))$, remove (q, e_1, e_2) from P_1 and modify P_2 according to $\varDelta_2$, i.e., if $\varDelta_2 = -1$ then remove one element and so forth.
   In the example, P_1 and P_2 change as follows: $P_1 = \{(q', +1, 0)\}, P_2 = \{95\}$.
5. If $P_1((q', \varDelta_1, \varDelta_2))$ and P_2 does not contain (q, e_1', e_2') for any e_1', e_2', and $\neg P_2((q', \varDelta_1, \varDelta_2))$, add $(q', \varDelta_1, \varDelta_2)$ to P_2.
   In the example, P_1 and P_2 change as follows: $P_1 = \{(q', +1, 0)\}, P_2 = \{95, (q', +1, 0)\}$.
6. If P_1 and P_2 contain $(q', \varDelta_1, \varDelta_2)$, remove $(q', \varDelta_1, \varDelta_2)$ from P_1 and add $(q', ?, ?)$ to P_2. *In the example, P_1 and P_2 change as follows: $P_1 = \{\}, P_2 = \{95, (q', +1, 0), (q', ?, ?)\}$.*
7. If $P_2((q', \varDelta_1, \varDelta_2))$ and $\neg P_1((q', \varDelta_1, \varDelta_2))$, then modify P_1 according to $\varDelta_1$ and remove $(q', \varDelta_1, \varDelta_2)$ from P_2.
   In the example, P_1 and P_2 change as follows: $P_1 = \{1543\}, P_2 = \{95, (q', ?, ?)\}$.
8. If $P_2((q', ?, ?))$, remove $(q', ?, ?)$ from P_2 and add it to P_1.
   In the example, P_1 and P_2 change as follows: $P_1 = \{1543, (q', ?, ?)\}, P_2 = \{95\}$.

Clearly, $P_1 = \{1543, (q', ?, ?)\}, P_2 = \{95\}$ describes the configuration $(q', 1, 1)$.

Note that the rules above are formalised by means of conjunctions. For example, and instance of the rule 1 is formalised as $P_1((q,?,?)) \wedge \textsc{Empty}(P_2) \wedge \neg P_2((q,?,\top)) \wedge P'_2((q,?,\top)) \wedge \neg P'_1((q,?,?)) \wedge \textsc{Insert}(P_2) \wedge \textsc{Delete}(P_1)$. In this case, the conjunct $\textsc{Delete}(P_1)$ assures that $(q,?,?)$ is the only element deleted from P_1, while $\textsc{Insert}(P_2)$ and $\neg P_2((q,?,\top))$ guarantee that $(q,?,\top)$ is the only element added to P_2.

Paths in $\mathcal{K}_{\mathcal{S}^{\mathcal{A}}}$ are such that each 8-th state represents the configuration of $\mathcal{A}$, and the subpath containing every 8-th state of paths of $\mathcal{K}_{\mathcal{S}^{\mathcal{A}}}$ represents computations. So there exists an infinite path in $\mathcal{K}_{\mathcal{S}^{\mathcal{A}}}$ if and only if there is an infinite run of $\mathcal{A}$. □

3.3 Undecidability of model checking $CARL_{\mathcal{D},\emptyset}$-artifacts systems

In the proof of Theorem 15 we use arbitrary many constants. However, this is not the source of undecidability.

Theorem 18. *Model checking $CARL_{\mathcal{D},\emptyset}$-artifact systems against FO$\underline{^{sa}}$CTL is undecidable.*

Proof. For a given two-counter automaton $\mathcal{A}$, we show how to define $CARL_{\mathcal{D},\emptyset}$-artifact systems $S_{\mathcal{A}}$ and an FO$\underline{^{sa}}$CTL formula φ such that $S \models \varphi$ iff $\mathcal{A}$ has an infinite run. The proof is similar to the proof of Theorem 15 in case of three relations, except that now we use additional relations to represent the state of $\mathcal{A}$.

More precisely, we define $S^{\mathcal{A}}$ as $\langle \mathcal{D}^{\mathcal{A}}, \mathbb{N}, D_0^{\mathcal{A}}, \mathbb{T}^{\mathcal{A}} \rangle$, where $\mathcal{D} = \{P_1/1, \ldots, P_{|Q|+2}/1\}$, D_0 is such that all P_i are empty except for P_3 which is a singleton relation containing 0.

For each tuple $t = (q_i, \phi_1, \phi_2, q_j, \Delta_1, \Delta_2) \in R$, we define an artifact transition α_t as:
$\neg\textsc{Empty}(P_{i+2}) \wedge empty_{\phi_1}(P_1) \wedge empty_{\phi_2}(P_2) \wedge \neg\textsc{Empty}(P'_{j+2}) \wedge \bigwedge_{k \neq j} \textsc{Empty}(P'_{k+2}) \wedge modify_{\Delta_1}(P_1) \wedge modify_{\Delta_2}(P_2)$

It can be shown that $\mathcal{K}_{\mathcal{S}^{\mathcal{A}}}$ represent the computations of $\mathcal{A}$. □

3.4 Further results

Since there is a translation from $CARL_{\mathcal{D},\mathbb{N}}$ to $FO^2_{\mathcal{D},\mathbb{N}}$, we obtain the following.

Corollary 19. *Model checking $FO^2_{\mathcal{D},\mathbb{N}}$-artifact systems against FO$\underline{^{sa}}$CTL whose database schema contains two unary symbols is undecidable.*

Corollary 20. *Model checking $FO^2_{\mathcal{D},\emptyset}$-artifact systems against FO$\underline{^{sa}}$CTL is undecidable.*

It can also be shown that including a single binary relation leads to undecidability. This result can be extended to any relations of arity greater that two.

Theorem 21. *Model checking $FO^2_{\mathcal{D},\mathbb{N}}$-artifact systems whose database contains a single binary relation against FO$\underline{^{sa}}$CTL is undecidable.*

Proof. We use the constants $1, 2, 3$ to simulate three relations.

As before we define $S^{\mathcal{A}}$ as $\langle \mathcal{D}^{\mathcal{A}}, \mathbb{N}, D_0^{\mathcal{A}}, \mathbb{T}^{\mathcal{A}} \rangle$. This time, we take $\mathcal{D} = \{P_1/2\}$ and D_0 is such that $P_1 = \{(3,0)\}$. We modify the set of artifact transitions from the proof in case of three unary relations in the following manner:

- Replace $P_i(t)$ by $P_1(i,t)$ and $P'_i(t)$ by $P'_1(i,t)$.
- Replace $\textsc{Empty}(P_i)$ by $\forall x \neg P_1(i,x)$.
- Replace $\textsc{Equal}(P_i)$ by $\forall x. P'_1(i,x) \Leftrightarrow P_1(i,x)$.
- Replace $\textsc{Insert}(P_i)$ by $\exists x. \neg P_1(i,x) \wedge P'_1(i,x) \wedge \forall y. y \neq x \Rightarrow (P_1(i,x) \Leftrightarrow P'_1(i,x))$.
- Replace $\textsc{Delete}(P_i)$ by $\exists x. \neg P'_1(i,x) \wedge P_1(i,x) \wedge \forall y. y \neq x \Rightarrow (P_1(i,x) \Leftrightarrow P'_1(i,x))$.

It can be checked that paths in $\mathcal{K}_{\mathcal{S}^{\mathcal{A}}}$ represent the computations of $\mathcal{A}$. So an infinite path exists iff there exists an infinite computation of $\mathcal{A}$. □

4 Model Checking $FO^*_{\mathcal{D},\mathbb{N}}$–Artifact Systems with a Single Unary Relation

While the results of the previous section are negative, in the following we identify a fragment for which the model checking problem is decidable.

Theorem 22. *Model checking $FO^*_{\mathcal{D},\mathbb{N}}$-artifact systems whose database consists of a single unary relation symbol against FO$\underline{^{sa}}$CTL is decidable.*

The above identifies a restriction on the database schema, but allows the most general logic here considered to define the transitions of the systems. It is instructive to see that restricting our attention to unary relations still enables us to define interesting classes of systems.

Example 23. *An infinite state counter machine can help organising queues. The task of the machine is to print numbered tickets, display the ticket number on a screen, and remove the number from memory. The machine has three states: ON where it prints tickets and displays numbers, $CLOSING$ where it displays numbers but will not print tickets, and OFF where it remains idle.*

The machine can be described as an artifact system with a database consisting of a single unary relation. We can use the constants 0, 1 and 2 to represent the states ON, $CLOSING$ and OFF, and all other natural numbers to represent tickets. An example of a property that may be expressed in FO$\underline{^{sa}}$CTL is whether it is always possible to remove all numbers from the machine's memory. It follows from Theorem 22 that checking this specification is decidable.

The rest of this section is organised as follows. Firstly, we investigate the expressive power of the logic $FO^*_{\mathcal{D}\oplus\mathcal{D}',\mathbb{N}}$. In the following subsection we define the logic $\mathfrak{Card}$ and we show that any set of artifact transitions defined in $FO^*_{\mathcal{D}\oplus\mathcal{D}',\mathbb{N}}$ can be replaced by an equivalent set of formulae from the logic $\mathfrak{Card}$. Then, in Subsection 4.2 we show how to transform an artifact system $\mathcal{S}$ into an infinite Kripke structure $\mathcal{K}'$ and an FO$\underline{^{sa}}$CTL property φ into a CTL property φ' such that $\mathcal{K}_{\mathcal{S}} \models \varphi$ iff $\mathcal{K}' \models \varphi'$. In Subsection 4.3 we define a pushdown system $\mathcal{P}$ and a CTL* formula φ'' such that $\mathcal{K}' \models \varphi'$ iff $\mathcal{P} \models \varphi''$. We conclude in Subsection 4.4 by giving the algorithm solving the decidability of Theorem 22.

In the following we assume $\mathcal{D} = \{P/1\}$.

4.1 The logic $\mathfrak{Card}$

A *grounded literal* over $\mathcal{D}$ is one of the following formulae: $P(c)$, $\neg P(c)$, $P'(c)$, $\neg P'(c)$, where c stands for any constant.

Definition 24. *A formula is called a* $\mathfrak{Card}$-transition *if it is of the form $\Upsilon \wedge \bigwedge_{i\in I} \exists_{\gamma_i b_i} x \Phi_i$ where Υ is a conjunction of grounded literals, $\gamma_i \in \{\geq, \leq\}$, b_i are natural numbers (including 0), and Φ_i are a boolean combinations of $P(x)$ and $P'(x)$.*

The formulae of the logic $\mathfrak{Card}$ *are disjunctions of* $\mathfrak{Card}$-transitions.

Intuitively, $\exists_{\leq b} x\Phi(x)$ represents the fact that there are at most b distinct elements satisfying Φ; conversely $\exists_{\geq b} x\Phi(x)$ signifies that there are at least b distinct elements satisfying Φ. The formal semantics of the logic $\mathfrak{Card}$ is defined by the translation into $FO^*_{\mathcal{D},\mathbb{N}}$ that simply replaces each $\exists_{\geq b}\Phi(x)$ with $\exists x_1,\ldots,x_b. \bigwedge_{i\neq j} x_i \neq x_j \wedge \bigwedge_i \Phi(x_i)$ and $\exists_{\leq b}\Phi(x)$ with $\forall x_1,\ldots,x_{b+1}. \bigwedge_{i\neq j} x_i \neq x_j \Rightarrow \bigvee_i \neg\Phi(x_i)$.

As we prove below, the logics $\mathfrak{Card}$ and $FO^*_{\mathcal{D},\mathbb{N}}$ have the same expressive power.

Example 25. *Consider the formula $\forall y(y \neq c_1 \Rightarrow (P(y) \Leftrightarrow P'(y)))$. This is equivalent to the disjunction of the following formulae:*

- $P(c_1) \wedge P'(c_1) \wedge \exists_{\leq 0} x(P(x) \Leftrightarrow \neg P'(x))$
- $\neg P(c_1) \wedge \neg P'(c_1) \wedge \exists_{\leq 0} x(P(x) \Leftrightarrow \neg P'(x))$
- $P(c_1) \wedge \neg P'(c_1) \wedge \exists_{=1} x(P(x) \wedge \neg P'(x))$ $\wedge \exists_{\leq 0} x(\neg P(x) \wedge P'(x))$
- $\neg P(c_1) \wedge P'(c_1) \wedge \exists_{\leq 0} x(P(x) \wedge \neg P'(x))$ $\wedge \exists_{=1} x(\neg P(x) \wedge P'(x))$

where $\exists_{=1}\Phi$ stands for $\exists_{\leq 1}\Phi \wedge \exists_{\geq 1}\Phi$.

Lemma 26. *Let Ψ be a $FO^*_{\mathcal{D}\oplus\mathcal{D}',\mathbb{N}}$ sentence. Then, there is a $\mathfrak{Card}$ formula Ψ' equivalent to Ψ such that the size of Ψ' is exponential in the size of Ψ.*

Proof. The logic $FO^*_{\mathcal{D}\oplus\mathcal{D}',\mathbb{N}}$ is a fragment of first-order logic, so we can use standard tools. The result follows by applying Ehrenfeucht-Fraisse games to the case of unary relations. Indeed, for any property Ψ of $FO^*_{\mathcal{D}\oplus\mathcal{D}',\mathbb{N}}$ there is a number $n \leq |\Psi|$ such that if duplicator wins an n rounds game on the two structures, then both structures satisfy Ψ or none of them does.

Let C be the set of constants from Ψ. Given two structures $\mathcal{A}, \mathcal{B}$ we write $\mathcal{A} \sim_C \mathcal{B}$ if for all constants $c \in C$ we have $P^{\mathcal{A}}(c)$ iff $P^{\mathcal{B}}(c)$ and $P'^{\mathcal{A}}(c)$ iff $P'^{\mathcal{B}}(c)$.

Let $cnt(\Phi(x))$ be the number of elements not from C satisfying $\Phi(x)$ and $cnt_n(\Phi(X))$ be equal to $min(n+1, cnt(\Phi(x)))$. For each model $\mathcal{A}$, we define $cnt_n(\mathcal{A})$ as a tuple (r,s,t,u) where

$$\begin{aligned} r &= cnt_n(P_{\mathcal{A}}(x) \wedge P'_{\mathcal{A}}(x)) \\ s &= cnt_n(P_{\mathcal{A}}(x) \wedge \neg P'_{\mathcal{A}}(x)) \\ t &= cnt_n(\neg P_{\mathcal{A}}(x) \wedge P'_{\mathcal{A}}(x)) \\ u &= cnt_n(\neg P_{\mathcal{A}}(x) \wedge \neg P'_{\mathcal{A}}(x)) \end{aligned}$$

That is, if $cnt_n(\mathcal{A}) = (1, n+1, 3, 4)$, then $\mathcal{A}$ contains 1 world not from C satisfying $P_{\mathcal{A}}(x) \wedge P'_{\mathcal{A}}(x)$, more than n worlds not from C satisfying $P_{\mathcal{A}}(x) \wedge \neg P'_{\mathcal{A}}(x)$ and so on.

Let $\mathcal{A}$ and $\mathcal{B}$ be two structures such that $\mathcal{A} \sim_C \mathcal{B}$ and $cnt_n(\mathcal{A}) = cnt_n(\mathcal{B})$. On these structures, duplicator has a winning strategy and therefore Ψ cannot distinguish them.

We construct a $\mathfrak{Card}$-transition $\Psi_{A,A',r,s,t,u}$ such that $\mathcal{A}$ satisfies $\Psi_{A,A',r,s,t,u}$ iff $cnt_n(\mathcal{A}) = (r,s,t,u)$ and $\mathcal{A} \sim_C \mathcal{B}$. Let $r' = r + |A \cap (C \setminus A')|$, $s' = s + |(C \setminus A) \cap A'|$, $t' = t + |A \cap A'|$ and $u' = u + |(C \setminus A) \cap (C \setminus A')|$. Let $\Psi_{A,A',r,s,t,u}$ be the conjunction of the following formulae:

1. $\bigwedge_{c\in A} P(c) \wedge \bigwedge_{c \in C\setminus A} \neg P(c)$
2. $\bigwedge_{c\in A'} P'(c) \wedge \bigwedge_{c \in C\setminus A'} \neg P'(c)$
3. $\exists_{\geq r'} x(P(x) \wedge P'(x))$
4. only if $r < n+1$: $\exists_{\leq r'} x(P(x) \wedge P'(x))$
5. $\exists_{\geq s'} x(P(x) \wedge \neg P'(x))$
6. only if $s < n+1$: $\exists_{\leq s'} x(P(x) \wedge \neg P'(x))$
7. $\exists_{\geq t'} x(\neg P(x) \wedge P'(x))$
8. only if $t < n+1$: $\exists_{\leq t'} x(\neg P(x) \wedge P'(x))$
9. $\exists_{\geq u'} x(\neg P(x) \wedge \neg P'(x))$
10. only if $u < n+1$: $\exists_{\leq u'} x(\neg P(x) \wedge \neg P'(x))$

The formula $\Psi_{A,A',r,s,t,u}$ is indeed a $\mathfrak{Card}$-transition.

Let $\mathcal{X}_\Psi$ be the set of all the formulae of the form $\Psi_{A,A',r,s,t,u}$ such that there is a structure that satisfies $\Psi \wedge \Psi_{A,A',r,s,t,u}$, and let Ψ' be the disjunction of all the formulae in $\mathcal{X}_\Psi$. We show that any structure $\mathcal{A}$ satisfies Ψ if and only if it satisfies Ψ'.

Assume that $\mathcal{A}$ satisfies Ψ. Let $A = \{c \in C \mid P^{\mathcal{A}}(c)\}$, $A' = \{c \in C \mid P'^{\mathcal{A}}(c)\}$ and $(r,s,t,u) = cnt_n(\mathcal{A})$. Obviously, $\mathcal{A}$ satisfies $\Psi_{A,A',r,s,t,u} \wedge \Psi$, therefore $\Psi_{A,A',r,s,t,u}$ is in $\mathcal{X}_\Psi$ and $\mathcal{A}$ satisfies Ψ'.

Assume that $\mathcal{A}$ satisfies Ψ'. It follows that $\mathcal{A}$ satisfies $\Psi_{A,A',r,s,t,u}$ for some A, A', r, s, t, u such that there is a structure $\mathcal{B}$ satisfying $\Psi_{A,A',r,s,t,u} \wedge \Psi$. So $\mathcal{A} \sim_C \mathcal{B}$ and $cnt_n(\mathcal{A}) = cnt_n(\mathcal{B})$, and therefore, by the Ehrenfeucht-Fraisse argument presented above, $\mathcal{A}$ satisfies Ψ.

Therefore we have constructed a $\mathfrak{Card}$ formula Ψ' that is equivalent to Ψ. A quick check shows that the number of elements in $\mathcal{X}_\Psi$ is bounded by $2^{2|C|} \cdot n^4$, which is exponential in size of Ψ. Note that if we consider formulae with a fixed number of constants, then the resulting formula is polynomial in size of Φ. $\square$

The above proof is constructive and can be mechanised. As remarked, the construction gives a $\mathfrak{Card}$ formula which is exponential in the size of the original $FO^*_{\mathcal{D}\oplus\mathcal{D}',\mathbb{N}}$ formula.

Note that for any set of artifact transitions $\mathbb{T}$ and formulae Ψ, Ψ', the sets $\mathbb{T} \cup \{\Psi, \Psi'\}$ and $\mathbb{T} \cup \{\Psi \vee \Psi'\}$ are equivalent; i.e., they prescribe the same transitions. Therefore, in the rest of this section we assume without loss of generality that all the artifact transitions are $\mathfrak{Card}$-transitions.

4.2 Infinite Kripke structures

Let $\mathcal{S} = \langle \mathcal{D}, \mathbb{N}, D_0, \mathbb{T}\rangle$ be a $\mathfrak{Card}$-artifact system and let $\mathcal{K}_{\mathcal{S}} = \langle \Sigma, D_0, \tau\rangle$ be the model of $\mathcal{S}$. Let φ be an FO$\underline{^{sa}}$CTL formula and $\mathcal{C}_a$ be the set of all constants appearing in $\mathbb{T}$ and φ. We define a Kripke structure $\mathcal{K}'$ and a CTL specification φ' such that $\mathcal{K}_{\mathcal{S}} \models \varphi$ iff $\mathcal{K}' \models \varphi'$.

Recall that the states of $\mathcal{K}_{\mathcal{S}}$, the model of $\mathcal{S}$, are in fact instances of P. The aim of this section is to define a Kripke structure $\mathcal{K}$ whose states represent the instances of P in the following way: an instance $D = \{P\}$ is represented by a pair $abs(D) = (P \cap \mathcal{C}_a, |P|)$, where $P \cap \mathcal{C}_a$ is the set of *relevant* constants that are in P and $|P|$ is the number of elements in P.

Let us consider $\mathcal{K}' = \langle \Sigma', s'_0, \tau', \pi' \rangle$ where:

- $\Sigma' = \{abs(s) \mid s \in \Sigma\}$ is the set of abstractions of all states in Σ,
- $s'_0 = abs(D_0)$,
- $\pi' : \Sigma' \to \mathcal{C}_a \cup \{e_0, e_1, \dots\}$ is a labelling such that $\pi((A, n)) = A \cup \{e_0, e_1, \dots, e_n\}$,
- $\tau' = \{(abs(D), abs(D')) \mid \tau(D, D')\}$.

We define a translation $\mathfrak{T}_{CTL}$ from FO$\underline{^{sa}}$CTL formulae into CTL formulae. Let Ψ be a $FO^*_{\mathcal{D},\mathbb{N}}$ sentence and Ψ' be an equivalent sentence that is a disjunction of $\mathfrak{C}$ard-transitions. We translate Ψ' into a CTL formula Ψ'' by replacing every $P(c_i)$ with c_i, $\exists_{\geq k} x P(x)$ with e_k, $\exists_{\leq k} x P(x)$ with $\neg e_{k+1}$, $\exists_{\geq k} x \neg P(x)$ with $\top$ (recall that P is always finite), and $\exists_{\leq k} x \neg P(x)$ with $\bot$. Finally, we replace Ψ with Ψ''.

We define $\mathfrak{T}_{CTL}(\psi)$ as the result of applying the above translation to all the $FO^*_{\mathcal{D},\mathbb{N}}$ sentences in ψ, and we define $\varphi' = \mathfrak{T}_{CTL}(\varphi)$.

The following lemma follows from the fact that the logic $\mathfrak{C}$ard cannot distinguish between elements other than constants. So when checking for satisfaction there is no need to keep the precise information about the content of P.

Lemma 27. *For an artifact system S, its model $\mathcal{K}_{\mathcal{S}}$, an FO$\underline{^{sa}}$CTL specification φ, let K' and φ' be the infinite Kripke structure and the CTL specification defined as above. Then, $\mathcal{K}_{\mathcal{S}} \models \varphi$ iff $\mathcal{K}' \models \varphi'$.*

Proof (sketch). We show by induction that for every subformula ψ of φ and every $D \in \Sigma$, we have $\mathcal{K}_{\mathcal{S}}, D \models \psi$ iff $\mathcal{K}', abs(D) \models \mathfrak{T}_{CTL}(\psi)$.

Consider the case when $\psi = AX\psi_1$ and let D be any state of Σ. Firstly, assume that $\mathcal{K}', abs(D) \models \mathfrak{T}_{CTL}(\psi)$. Let $D_1, \dots, D_n$ be all successors of D (i.e. $\tau(D, D_i)$ for all i). By the definition of τ', for each i we have $\tau'(abs(D), abs(D_i))$. Since $abs(D)$ satisfies $\mathfrak{T}_{CTL}(AX\psi_1)$, $abs(D_i)$ satisfies ψ_1 and, by inductive assumption, D_i satisfies ψ_1. Therefore, D satisfies ψ.

Assume that $\mathcal{K}_{\mathcal{S}}, D \models \psi$. Let (A', n') be a successor of $abs(D) = (A, n)$. This means that there are two states E, E' such that $abs(E) = (A, n)$, $abs(E') = (A', n')$, and $\tau(E, E')$. Let $n_c = |P^E \cap P^{E'} \setminus C_a|$ be the number of common elements in the relation P of E and E' except for constants from C_a. Let D' be such that $P^{D'}$ is a sum of A', any n_c elements of D and $n' - n_c - |A'|$ elements that are not in P^D or C_a. Then, for any $\mathfrak{C}$ard-transition Φ defined by using the constants from C_a we have $E, E' \models \Phi$ iff $D, D' \models \Phi$.

Therefore, we have $\tau(D, D')$; so D' satisfies ψ_1 and, by the induction hypothesis, $abs(D') = (A', n')$ satisfies $\mathfrak{T}_{CTL}(\psi_1)$. Since (A', n') was an arbitrary successor of $abs(D)$, we conclude that $\mathcal{K}', abs(D) \models \mathfrak{T}_{CTL}(\psi)$.

The remaining cases are similar and omitted. □

Finite representation. For convenience above we assumed that the labelling π' has an infinite codomain. However, φ' uses only finitely many propositional variables. In the rest of this section we assume that the codomain of π' is $\mathcal{C}_a \cup \{e_0, \dots, e_l\}$, where l is the greatest index such that e_l appears in φ'.

While the relation τ' is infinite, we can give a finite description for it. To do this, let b be the smallest natural number greater than all the numbers that appear in $\mathfrak{C}$ard-transitions of $\mathcal{S}$. Let $bound_b(n) = max(-b, min(n, b))$ be a function that returns $-b$ if $n < -b$, b is $n > b$ and n otherwise.

Let T be a set of tuples such that $(A, n_b, A', n'_b, \Delta) \in T$ if and only if for some n, n' such that $n_b = bound_b(n)$, $n'_b = bound_b(n')$, $\Delta = bound_b(n' - n)$ and $\tau'((A, n,), (A', n'))$.

This representation is sufficient since τ' is defined by $\mathfrak{C}$ard-transitions that do not distinguish sets larger than b. In other words, given a set T, we can construct τ' as follows: $\tau'((A, n,), (A', n'))$ if and only if $(A, bound_b(n_b), A', bound_b(n'_b), bound_b(n' - n)) \in T$.

Clearly, for any $(A, n_b, A', n'_b, \Delta) \in T$ we have $A, A' \subseteq C_a$, $n, n' \in \{0, \dots, b\}$ and $\Delta \in \{-b, \dots, b\}$, so T is finite.

4.3 Pushdown systems

Definition 28. *A* pushdown system *is a tuple $\mathcal{P} = (\Sigma^{\mathcal{P}}, q_0^{\mathcal{P}}, \Gamma^{\mathcal{P}}, \bot^{\mathcal{P}}, \tau^{\mathcal{P}}, \pi^{\mathcal{P}})$, where $\Sigma^{\mathcal{P}}$ is a set of states containing $q_0^{\mathcal{P}}$, $\Gamma^{\mathcal{P}}$ is a set of stack symbols containing the stack bottom symbol $\bot^{\mathcal{P}}$, $\Delta^{\mathcal{P}} \subseteq (\Sigma^{\mathcal{P}} \times \Gamma^{\mathcal{P}}) \times (\Sigma^{\mathcal{P}} \times (\Gamma^{\mathcal{P}} \setminus \{\bot^{\mathcal{P}}\})^*)$ is a finite set of transition rules, and $\pi^{\mathcal{P}} : \Sigma^{\mathcal{P}} \times \Gamma^{\mathcal{P}} \to Var$ is a labelling function for some fixed set of propositional variables Var.*

A configuration *of a pushdown system is a pair (s, α), where $s \in \Sigma^{\mathcal{P}}$ and $\alpha \in \Gamma^{\mathcal{P}*}$ starts from $\bot^{\mathcal{P}}$ and does not contain $\bot^{\mathcal{P}}$ at any other position. We define the transition relation $\to_{\mathcal{P}}$ as follows: $(s, \alpha\gamma) \to_{\mathcal{P}} (s', \alpha\alpha')$, where $\gamma \in \Gamma^{\mathcal{P}}$ and $\alpha, \alpha' \in \Gamma^{\mathcal{P}*}$, iff $(s, \gamma, s', \alpha') \in \tau^{\mathcal{P}}$.*

Runs are defined as usual (Hopcroft and Ullman 1979). It is known that model checking pushdown systems against CTL* is decidable (Bozzelli 2007).

In what follows, given a Kripke model we define a pushdown system whose configurations represent states (A, n) of the Kripke structure by using the system states to encode A and the stack to store n. In doing so we will ensure that the content of the stack is always a prefix of the word $012 \dots (b-1)b^\omega$. This will allow us to check easily the exact value of n if $n < b$. Since we can remove only one element from the stack at a time, we will use a number of auxiliary configurations to simulate a change of state of Kripke structure.

Formally, let $B = \{-b, \dots, b\}$, $B_+ = \{0, \dots, b\}$ and $P_C = \{A \mid A \subseteq C_a\}$. Below we define a pushdown system $\mathcal{P} = (P_C \cup P_C \times B \times B, \emptyset, B_+, 0, \tau'', \pi'')$ and a CTL* formula φ'' s.t. $\mathcal{P} \models \varphi''$ iff $\mathcal{K}' \models \varphi'$.

We represent a configuration of $\mathcal{P}$ of the form (q, α) as $(q, |\alpha| - 1)$. Since $(q, 0)$ is represented by $(q, 0)$, no ambiguity arises.

A configuration of the form (A, n) represents the state (A, n) of $\mathcal{K}'$. A configurations of the form $((A, n, k), m)$ for

$|n| < b$ is an auxiliary configuration that represents an intermediate state such that $min(b, n+m) = k$ and that the state precedes the state $(A, n+m)$ that will appear in $n+1$ steps. Similarly, a configuration of the form $((A, b, k), m)$ represents the situation where the value stored in the stack can be increased by any number greater than or equal to b. Finally, a configuration of the form $((A, -b, k), m)$ encodes the situation where the value stored in the stack can be decreased by any number greater than or equal to b.

Let $add\_one(m) = m.(min(m+1, b))$ where "." stands for concatenation. Formally, τ'' contains the following rules.

- $((A, b, k), m, (A, b, k), add\_one(m))$ for all possible A, k, m.
- $((A, n, k), m, (A, n-1, k), add\_one(m))$ for all $n > 0$ and all possible A, k, m.
- $((A, n, k), m, (A, n+1, k), \emptyset)$ for all $n < 0$ and all possible A, k, m.
- $((A, -b, k), m, (A, -b, k), \emptyset)$ for all possible A, k, m.
- $(A, n_b, (A', n, n'_b), n_b)$ for all $(A, n_b, A', n'_b, n) \in T$.
- $((A, 0, n'_b), n'_b, A, n'_b)$ and all possible A, n'_b.

Example 29. *Assume that* $\tau'((\emptyset, 4), (\{7\}, 3))$ *and* $b = 2$. *The corresponding computation of* $\mathcal{P}$ *is* $(\emptyset, 01222) \rightarrow_{\mathcal{P}} ((\{7\}, -1, 2), 01222) \rightarrow_{\mathcal{P}} ((\{7\}, 0, 2), 0122) \rightarrow_{\mathcal{P}} (\{7\}, 0122)$.

For $\tau'((\{7\}, 3), (\emptyset, 6))$, *the corresponding computation is:* $(\{7\}, 0122) \rightarrow_{\mathcal{P}} ((\emptyset, 2, 2), 0122) \rightarrow_{\mathcal{P}} ((\emptyset, 2, 2), 01222) \rightarrow_{\mathcal{P}} ((\emptyset, 1, 2), 012222) \rightarrow_{\mathcal{P}} ((\emptyset, 0, 2), 0122222) \rightarrow_{\mathcal{P}} (\emptyset, 0122222)$.

Finally, we define π'' as $\pi''((A, n, k), m) = \emptyset$ and $\pi''(A, m) = A \cup \{e_0, \ldots, e_m\}$.

Note that not all runs of $\mathcal{P}$ correspond to runs of $\mathcal{K}'$. For example, for $b = 2$ the run
$(\emptyset, 0) \rightarrow_{\mathcal{P}} ((\emptyset, 2, 2), 0) \rightarrow_{\mathcal{P}} ((\emptyset, 2, 2), 01) \rightarrow_{\mathcal{P}} ((\emptyset, 2, 2), 012) \rightarrow_{\mathcal{P}} \ldots$ continuously increases the stack and will never reach a state of the form (A, m). To avoid such runs, we add a fairness constraint by considering only the paths satisfying e_0 infinitely often.

To transform φ' to a CTL* formula φ'', we define a function $\mathfrak{T}^*$ by induction as follows.

$$\begin{array}{lcl}
\mathfrak{T}^*(p) & = & p, \text{ where } p \text{ is a propositional variable} \\
\mathfrak{T}^*(\neg\psi) & = & \neg\mathfrak{T}^*(\psi) \\
\mathfrak{T}^*(\psi_1 \vee \psi_2) & = & \mathfrak{T}^*(\psi_1) \vee \mathfrak{T}^*(\psi_2) \\
\mathfrak{T}^*(AX\psi) & = & A(GFe_0 \Rightarrow \neg e_0 U(e_0 \wedge \mathfrak{T}^*(\psi))) \\
\mathfrak{T}^*(A\psi_1 \mathcal{U} \psi_2) & = & A(GFe_0 \Rightarrow (e_0 \Rightarrow \mathfrak{T}^*(\psi_1))U \\
 & & (e_0 \wedge \mathfrak{T}^*(\psi_2))) \\
\mathfrak{T}^*(E\psi_1 \mathcal{U} \psi_2) & = & E(GFe_0 \wedge (e_0 \Rightarrow \mathfrak{T}^*(\psi_1))U \\
 & & (e_0 \wedge \mathfrak{T}^*(\psi_2)))
\end{array}$$

Consider $\varphi'' = \mathfrak{T}^*(\varphi')$. We have the following.

Lemma 30. *For an infinite Kripke structure K' and a CTL specification φ', let P and φ'' be the pushdown system and the CTL* specification defined as above. Then, $\mathcal{K}' \models \varphi'$ iff $\mathcal{P} \models \varphi''$.*

The proof of the above lemma follows from the construction of $\mathcal{P}$ and φ''.

4.4 Model checking

In view of the translation defined above we can now give the algorithm that establishes Theorem 22.

Algorithm `Input`: $FO^*_{\mathcal{D},\mathbb{N}}$-artifact system $\mathcal{S}$ built on a unary relation, FO$\underline{^{sa}}$CTL formula φ.
`Output`: TRUE or FALSE.

1. Transform the set of artifact transitions to the set of $\mathfrak{Card}$-transitions as described in Subsection 4.1.
2. Generate a finite representation of the Kripke structure $\mathcal{K}'$ and a CTL formula φ' as described in Subsection 4.2.
3. Generate a pushdown system $\mathcal{P}$ and a CTL* formula φ'' as defined in Subsection 4.3.
4. Return the result of model checking $\mathcal{P}$ against φ''.

The Algorithm above gives the proof of Theorem 22. Notice that in the last step we employ the algorithm for model checking the whole of CTL*; so it can also be applied against sentence atomic FO-CTL*.

An upper bound for the model checking problem of Theorem 22 can be established by observing that model checking pushdown systems against CTL* is 2EXPTIME-complete (Bozzelli 2007) and it is applied to a pushdown system exponential in the size of the input.

Corollary 31. *The problem of model checking artifact systems whose database consists of a single unary relation against sentence-atomic FO-CTL* is in* 3EXPTIME.

5 Conclusions

Artifact-centric systems have been put been forward as a new paradigm in data-aware services and they are increasingly being used as a user-oriented architecture upon which services can be deployed. The issue of verification of artifact-centric systems remains of paramount importance both at theoretical and at practical level.

A number of classes of artifact-centric systems have been proposed for which the verification problem is decidable. As discussed in the related work part of the introduction, a key requirement made of all these system is boundedness of the systems' executions. While it has been shown this can be imposed on the system at program level (Belardinelli, Lomuscio, and Patrizi 2011), there remain classes of scenarios whereby no bound can in principle be set either on the states or the runs of the system.

To explore the implications of this we have here addressed the verification of unbounded artifact-centric systems against sentence-atomic FO-CTL specifications. Our results show that not only the problem is undecidable in general, but that it remains so even when admitting only the simple artifact-centric descriptions built on the logics $CARL_{\mathcal{D},\mathbb{N}}$ and $CARL_{\mathcal{D},\emptyset}$.

While this points to a bleak picture for the verification of unbounded systems in general terms, we were able to identify a class of unbounded systems for which verification against sentence-atomic FO-CTL is decidable. This class is in a way "maximal" with respect to the relations in the database schema.

References

Alonso, G.; Casati, F.; Kuno, H. A.; and Machiraju, V. 2004. *Web Services - Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Springer.

Belardinelli, F., and Lomuscio, A. 2013. Decidability of model checking non-uniform artifact-centric quantified interpreted systems. In *Proceedings of the 23rd International Joint Conference on Artificail Intelligence (IJCAI'13)*, 725–731. AAAI Press.

Belardinelli, F.; Lomuscio, A.; and Patrizi, F. 2011. Verification of Deployed Artifact Systems via Data Abstraction. In *Proceedings of the 9th International Conference on Service-Oriented Computing (ICSOC'11)*, volume 7200 of *Lecture Notes in Computer Science*, 142–156. Springer.

Belardinelli, F.; Lomuscio, A.; and Patrizi, F. 2012a. An abstraction technique for the verification of artifact-centric systems. In *Proceedings of the 13th International Conference Principles of Knowledge Representation and Reasoning (KR12)*, 319–328. AAAI Press.

Belardinelli, F.; Lomuscio, A.; and Patrizi, F. 2012b. Verification of gsm-based artifact-centric systems through finite abstraction. In *Proceedings of the 10th International Conference on Service Oriented Computing (ISCOC'12)*, volume 7636 of *Lecture Notes in Computer Science*, 17–31. Springer.

Bozzelli, L. 2007. Complexity results on branching-time pushdown model checking. *Theoretical computer science* 379(1):286–297.

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 1999. *Model Checking*. Cambridge, Massachusetts: The MIT Press.

Cohn, D., and Hull, R. 2009. Business Artifacts: A Data-Centric Approach to Modeling Business Operations and Processes. *IEEE Data Engineering Bulletin* 32(3):3–9.

Deutsch, A.; Hull, R.; Patrizi, F.; and Vianu, V. 2009. Automatic verification of data-centric business processes. In *Proceedings of the 12th International Conference on Database Theory (ICDT'09)*, volume 361 of *ACM International Conference Proceeding Series*, 252–267. ACM.

Gonzalez, P.; Griesmayer, A.; and Lomuscio, A. 2012. Verifying GSM-based business artifacts. In *Proceedings of the 19th International Conference on Web Services (ICWS'12)*, 25–32. IEEE Press.

Grädel, E.; Kolaitis, P. G.; and Vardi, M. Y. 1997. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic* 3(1):53–69.

Grädel, E. 1999. On the restraining power of guards. *Journal of Symbolic Logic* 64(4):1719–1742.

Hariri, B. B.; Calvanese, D.; Giacomo, G. D.; Masellis, R. D.; and Felli, P. 2011. Foundations of relational artifacts verification. In *Proceedings of the 9th Int. Conference on Business Process Management (BPM'11)*, volume 6896 of *Lecture Notes in Computer Science*, 379–395. Springer.

Hariri, B. B.; Calvanese, D.; Montali, M.; Giacomo, G. D.; and Deutsch, A. 2013. Verification of relational data-centric dynamic systems with external services. In *Proceedings of the 32nd Symposium on Principles of Database Systems (PODS'13)*, 163–174. ACM.

Heath III, F. T.; Boaz, D.; Gupta, M.; Vaculín, R.; Sun, Y.; Hull, R.; and Limonad, L. 2013. Barcelona: A design and runtime environment for declarative artifact-centric BPM. In *Proceedings of the 11th International Conference on Service-Oriented Computing (ICSOC13)*, volume 8274 of *Lecture Notes in Computer Science*, 705–709. Springer.

Hopcroft, J. E., and Ullman, J. D. 1979. *Introduction to Automata Theory, Languages, and Computation*. Adison-Wesley Publishing Company.

Hull, R.; Damaggio, E.; De Masellis, R.; Fournier, F.; Gupta, M.; Heath, III, F. T.; Hobson, S.; Linehan, M.; Maradugu, S.; Nigam, A.; Sukaviriya, P. N.; and Vaculin, R. 2011. Business Artifacts with Guard-Stage-Milestone Lifecycles: Managing Artifact Interactions with Conditions and Events. In *Proceedings of the 5th ACM International Conference on Distributed Event-Based Systems (DEBS'11)*, 51–62. ACM.

Hull, R. 2008. Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges. In *Proceedings of the 2008 Confederated International Conferences CoopIS, DOA, GADA, IS, and ODBASE*, volume 5332 of *Lecture Notes in Computer Science*, 1152–1163. Springer.

Minsky, M. 1967. *Computation: finite and infinite machines*. Prentice-Hall.