Dynamic Logic for Data-aware Systems: Decidability Results

Francesco Belardinelli
Laboratoire IBISC, UEVE
IRIT Toulouse
belardinelli@ibisc.fr

Andreas Herzig
IRIT Toulouse
Université de Toulouse
CNRS
herzig@irit.fr

Abstract

We introduce a first-order extension of dynamic logic (FO-DL), suitable to represent and reason about the behaviour of Data-aware Systems (DaS), which are systems whose data content is explicitly exhibited in their description. We illustrate the expressivity of the formal framework by modelling English auctions as DaS and by specifying relevant properties in FO-DL. Most importantly, we develop an abstraction-based verification procedure, thus proving that the model checking problem for DaS against FO-DL is decidable, provided some mild assumptions on the interpretation domain.

1 Introduction

In recent years data-aware and data-driven systems acquired pre-eminence as a formal framework to represent and reason about systems in which data play a key role in the system's execution [Bhattacharya *et al.*, 2007; Deutsch *et al.*, 2007; 2009]. Initially, data-aware systems (DaS) appeared as a novel paradigm for the design of business processes in service-oriented computing [Singh and Huhns, 2005]. The originality of this approach consists in "combin[ing] data and processes in a holistic manner as the basic building block[s]" of the system's description [Cohn and Hull, 2009]. Normally a DaS is comprised of a *data model* accounting for the relational structure of data, as well as the *business processes* manipulating data. Both the data model and business processes are equally important components of the system specification. Due to that, DaS differ profoundly from most of the tradition on service architectures and composition, in which data are typically abstracted away to reduce the complexity of the system description, thus making the verification task amenable to standard model checking techniques [Singh and Huhns, 2005].

Recently, the data-driven paradigm has found applications beyond service-oriented computing. Notably, a wealth of contributions have put forward data-driven approaches to knowledge representation and reasoning (KR&R) as well as to the modelling of multi-agent system (MAS). More specifically, [Bagheri *et al.*, 2011; 2013] introduced knowledge and action bases (KAB) to describe distributed systems comprising a relational database, over which agent-services operate. KAB were then extended to include a number of different features relevant for modelling MAS, including commitments [Montali *et al.*, 2014], plans [Calvanese *et al.*, 2016], and norms [Baldoni *et al.*, 2016]. More directly related to the present contribution, [Belardinelli *et al.*, 2012; 2014] tackled the verification problem for artifact-centric multi-agent systems—a particular class of data-aware systems—and proved it to be decidable under specific assumptions. Here we extend the state of the art by considering an expressive dynamic modal language. Indeed, in the line including [Deutsch *et al.*, 2009; Bagheri *et al.*, 2011; 2013; Belardinelli *et al.*, 2012; 2014], the specification language for data-aware systems is typically a first-order extension of either branching-time temporal logic CTL or of the μ-calculus [Clarke *et al.*, 1999]. In such languages one can express properties holding in some or all executions of the system. However, the structure of such executions is not exhibited transparently in the specification language, and actions do not appear explicitly in the temporal operators of CTL nor in those of the μ-calculus. To overcome these issues, here we consider a dynamic modal language [Pratt, 1976; Harel *et al.*, 2000] in which actions are explicitly named in the modal operators. This syntactic choice allows us to express complex behaviours in a precise and concise way. However, the application of dynamic logic to the specification of data-aware systems calls for novel verification techniques w.r.t. the state of the art.

Contribution. We introduce a novel interpretation of first-order dynamic logic that suits the formal description of relevant aspects of DaS. The language has assignments $v := t$ and $v :=?$ as atomic actions, which are intuitively interpreted as assigning to variable v the value t, or, respectively, any value. More complex actions are built by using the operators of Propositional Dynamic Logic (PDL). Most importantly, by using finite abstractions we prove that the model checking problem for a significant class of data-aware systems is decidable. We illustrate the importance of the technical result through an application to the verification of English auctions.

Related work. Dynamic logic is one of the success stories of formal methods applied to computer science. However, its first-order extensions are comparatively less studied than its standard propositional version. The language we consider appears in [Harel *et al.*, 2000] with some key differences. First, we do not include function symbols and the only relation symbols are the identity '$=$' and *less or equal than* '$\leq$'. More importantly, we interpret this language on structures with a finite *active* domain. We will show that this key difference, motivated by the literature on databases, allows us to obtain a decidable verification problem.

As regards DaS, besides the contributions discussed above, a further line of research works within the situation calculus [De Giacomo *et al.*, 2012; de Giacomo *et al.*, 2014; De Giacomo *et al.*, 2016]. By assuming that the underlying theories are *bounded*, the model checking problem is proved decidable. Here we do not need to assume boundedness as a separate hypothesis. More closely related to the present contribution, [**?**] shows that GOLOG programs in the situation calculus are naturally bounded, and therefore their verification is decidable [**?**]. Differently from these works (including [Bagheri *et al.*, 2011; 2013; Belardinelli *et al.*, 2012; 2014]) here we consider a rich first-order dynamic modal language where actions are explicit.

2 Data-aware Systems and Dynamic Logic

In this section we introduce formally DaS as well as all notions to be used in the rest of the paper. Hereafter we assume:

1. a finite, non-empty set V of individual variables $v_1, \ldots, v_n$;
2. a finite, possibly empty set C of individual constants $c_1, \ldots, c_m$;
3. a possibly infinite set $U \supseteq C$ of individuals $u_1, u_2, \ldots$ for the interpretation of variables, together with a total order $\leq$ defined on U.

Definition 1 (DaS) *A* Data-aware System *is a tuple $\mathcal{M} = \langle V, C, U\rangle$, where V, C, and U are defined as above.*

Intuitively, given a program written in some programming language (e.g., Fortran, Java, C), a DaS $\mathcal{M}$ is the collection of all variables and constants appearing in the program, together with the range of values for the variables. Notice that, while most programming languages are typed, here we consider an untyped formalism. We do so to keep the presentation more easily accessible to the general audience. The present formalism can be extended quite seamlessly to typed languages, even though the exercise is time- and space-consuming. Hereafter we distinguish between general data-aware systems and DaS in the technical sense of Def. 1.

Given a DaS $\mathcal{M}$, a *state* $s : V \to U$ is an interpretation of the variables in V. States can be thought of as a representation of the registers of the program associated with $\mathcal{M}$ at a given time. Given a state s, its *active domain* $adom(s) = \{u \in U \mid u = s(v) \text{ for some } v \in V\} \cup C \subseteq U$ is the co-domain of s (the set of all values for the variables in V at s), together with constant values. This (standard) notion of active domain is taken from the literature on database systems [Abiteboul *et al.*, 1995]. Observe that $adom(s)$ is finite as V and C are.

Finally, since the domain U of interpretation is possibly infinite, we have an infinite number of states in general, and therefore DaS are infinite-state systems.

Syntax. In order to write formulas that describe the dynamics of DaS, we consider an infinite set P of parameters $x_1, x_2, \ldots$. Then, a *term* $t \in T$ is any element in $V \cup C \cup P$, that is, t can be either a variable, or a constant, or a parameter. We introduce *assignments* $v := t$ and $v :=?$ as atomic actions, as well as *atomic formulas* $t = t'$ and $t \leq t'$, for $v \in V$ and $t, t' \in T$.

Definition 2 (FO-DL) *Formulas ϕ and actions α in First-order Dynamic Logic are defined by mutual recursion as follows:*

$$\begin{array}{lcl} \phi & ::= & t = t \mid t \leq t \mid \neg\phi \mid \phi \to \phi \mid [\alpha]\phi \mid \forall x \phi \\ \alpha & ::= & v := t \mid v :=? \mid \alpha ; \alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \phi? \end{array}$$

where $v \in V$ is a variable, $t \in T$ is a term, and $x \in P$ is a parameter.

Definition 2 is inspired by the language for first-order dynamic logic appearing in [Harel *et al.*, 2000]. Atomic actions are assignments $v := t$, that assign the value of term t to variable v, and "wildcard" assignments $v :=?$, that assign to v some arbitrary value from domain U. Complex actions are obtained by composition ;, non-deterministic choice $\cup$, iteration $*$, and test ?.

Differently from [Harel *et al.*, 2000], we here restrict the first-order component: FO-DL contains no function symbol and only two relation symbols, the identity '$=$' and *less or equal than* '$\leq$'. Indeed, first-order dynamic logic with two function symbols (and identity) has the same expressive power as elementary arithmetic [Harel *et al.*, 2000, Proposition 12.1], and therefore it is undecidable. Notwithstanding these restrictions, we will see that our FO-DL is expressive enough for many cases of interest; notably, we will see in in Section 2.1 that we can express auctions.

As a note on syntax, observe that parameters $x \in P$ are used to write formulas and are infinite in number, while variables $v \in V$ are really constituents of the Data-aware System and are finitely many. The two must be kept distinct. Hereafter, propositional connectives are introduced as standard and formula $\langle\alpha\rangle\phi$ is shorthand for $[\alpha]\phi$.

Given a term $t \in T$, we introduce the function $P(t)$ that returns set $\{t\}$ if $t = x \in P$ is a parameter, and the empty set otherwise. Then, given a formula ϕ and an action α in FO-DL, we define sets $\mathit{fr}(\phi)$ and $\mathit{fr}(\alpha)$ of free parameters, and $P(\phi)$ and $P(\alpha)$ of all parameters as follows:

$$\begin{array}{lllll} \mathit{fr}(t \star t') & = P(t \star t') & = & P(t) \cup P(t') & \\ \mathit{fr}(\neg\psi) & = \mathit{fr}(\psi) & \text{and} & P(\neg\psi) & = P(\psi) \\ \mathit{fr}(\psi \to \psi') & = \mathit{fr}(\psi) \cup \mathit{fr}(\psi') & \text{and} & P(\psi \to \psi') & = P(\psi) \cup P(\psi') \\ \mathit{fr}([\alpha]\psi) & = \mathit{fr}(\alpha) \cup \mathit{fr}(\psi) & \text{and} & P([\alpha]\psi) & = P(\alpha) \cup P(\psi) \\ \mathit{fr}(\forall x \psi) & = \mathit{fr}(\psi) \setminus \{x\} & \text{and} & P(\forall x \psi) & = P(\psi) \\ \mathit{fr}(v := t) & = P(v := t) & = & P(t) & \\ \mathit{fr}(v :=?) & = P(v :=?) & = & \emptyset & \\ \mathit{fr}(\alpha \sharp \alpha') & = \mathit{fr}(\alpha) \cup \mathit{fr}(\alpha') & \text{and} & P(\alpha \sharp \alpha') & = P(\alpha) \cup P(\alpha') \\ \mathit{fr}(\alpha^*) & = \mathit{fr}(\alpha) & \text{and} & P(\alpha^*) & = P(\alpha) \\ \mathit{fr}(\psi?) & = \mathit{fr}(\psi) & \text{and} & P(\psi?) & = P(\psi) \end{array}$$

for $\star \in \{=, \leq\}$ and $\sharp \in \{;, \cup\}$.

Observe that the definitions of *fr* and P differ only as to the clause for quantified formulas. Also, wildcard assignments $v :=?$ mention parameters implicitly. This remark motivates the following definition of $\#P(\phi) \in \mathbb{N}$:

$$\begin{array}{lcl} \#P(t \star t') & = & |P(t) \cup P(t')| = |P(t \star t')| \\ \#P(\neg\psi) & = & \#P(\psi) \\ \#P(\psi \to \psi') & = & \#P(\psi) + \#P(\psi') - |P(\psi) \cap P(\psi')| \end{array}$$

$$
\begin{array}{lcl}
\#P([\alpha]\psi) & = & \#P(\alpha) + \#P(\psi) - |P(\psi) \cap P(\psi')| \\
\#P(\forall x \psi) & = & \#P(\psi) \\
\#P(v := t) & = & |P(t)| = |P(v := t)| \\
\#P(v :=?) & = & 1 \\
\#P(\alpha \sharp \alpha') & = & \#P(\alpha) + \#P(\alpha') - |P(\alpha) \cap P(\alpha')| \\
\#P(\alpha^*) & = & \#P(\alpha) \\
\#P(\psi?) & = & \#P(\psi)
\end{array}
$$

Because of the base case $\#P(v :=?) = 1$ we have that $|P(\phi)| \leq \#P(\phi)$ and $|P(\alpha)| \leq \#P(\alpha)$. In particular, differently from P, $\#P$ keeps track of the occurrences of wildcard assignments as well. We use this fact in the construction of finite abstractions in Section 3. Notice that symbol P has been overloaded as it is used to denote functions on terms, formulas, and program expressions.

We now illustrate the expressive power of FO-DL by means of some toy examples.

Example 1 We consider a simple program that takes as input two integers, assign these as values to variables a and b, and then swaps their values by using a temporary variable tmp. Such a program can be specified in pseudo-code as follows:

```
int a, b;
int tmp;

    tmp := a;
    a   := b;
    b   := tmp;
```

Even for such a simple program we might want to check that *the new values for variables a and b are equal to the old values for b and a respectively*, and this is indeed the case for every possible value of a and b. To do so, consider the following formula in FO-DL:

$$
\begin{array}{c}
[a :=?; b :=?] \forall x, y((a = x \wedge b = y) \rightarrow \\
{[tmp := a; a := b; b := tmp]}\, (a = y \wedge b = x))
\end{array} \qquad (1)
$$

Intuitively, formula (1) corresponds to the specification above. In particular, parameters x and y are used to compare the values of variables a and b before and after the swap, that is, parameters are meant to store the value of variables throughout the computation.

As a further example, we consider an action α in FO-DL whose intended meaning is to compute the maximum among variables in $V = \{a_0, \ldots, a_n\}$:

$$
\begin{array}{rcl}
\alpha & = & max := a_0; \\
 & & \texttt{if } max < a_1 \texttt{ then } max := a_1; \\
 & & \vdots \\
 & & \texttt{if } max < a_n \texttt{ then } max := a_n;
\end{array}
$$

where $<$ is defined in the standard way (viz. $u < u'$ iff $u \leq u'$ and $u \neq u'$) and `if` ϕ `then` α is defined as usual in dynamic logic as $(\phi?; \alpha) \cup \neg\phi?$.

Then, we might want to check that action α actually returns the maximum. To this end, consider formula (2) in FO-DL that intuitively specifies that α is correct, as it actually computes the maximum, no matter what values for $a_0, \ldots, a_n$ are provided as input:

$$[a_0 :=?; \ldots; a_n :=?] \forall x [\alpha](x \leq max) \qquad (2)$$

As a result, FO-DL can be used as a specification language to express correct termination of programs involving variables for individuals, among other properties. ■

Semantics. To assign a meaning to the formulas and actions in FO-DL, we introduce *interpretations* $\sigma : P \rightarrow U$ of parameters into the domain U. In a state s, interpretations can be extended to any term $t \in T$ as follows: for $t = v \in V$, $(s \circ \sigma)(t) = s(v)$; for $t = c \in C$, $(s \circ \sigma)(t) = c$; while for $t = x \in P$, $(s \circ \sigma)(t) = \sigma(x)$. Hence, variables are interpreted according to s, parameters according to σ, while we adopt a Herbrandian interpretation of constants: they are interpreted as themselves. Given $u \in U$, σ^x_u is the interpretation that assigns u to $x \in P$ and coincides with σ on all other parameters.

Given a state s, we define the meaning of formula ϕ according to interpretation σ.

Definition 3 (Satisfaction) *The satisfaction relation $\models$ for DaS $\mathcal{M}$, state s, interpretation σ, and formula ϕ is given as:*

$$
\begin{array}{ll}
(\mathcal{M}, s, \sigma) \models t = t' & \textit{iff } (s \circ \sigma)(t) = (s \circ \sigma)(t') \\
(\mathcal{M}, s, \sigma) \models t \leq t' & \textit{iff } (s \circ \sigma)(t) \leq (s \circ \sigma)(t') \\
(\mathcal{M}, s, \sigma) \models \neg\phi & \textit{iff } (\mathcal{M}, s, \sigma) \not\models \phi \\
(\mathcal{M}, s, \sigma) \models \phi \rightarrow \phi' & \textit{iff } (\mathcal{M}, s, \sigma) \not\models \phi \textit{ or } (\mathcal{M}, s, \sigma) \models \phi' \\
(\mathcal{M}, s, \sigma) \models [\alpha]\phi & \textit{iff for all } s', R_\alpha(s, s') \textit{ implies } (\mathcal{M}, s', \sigma) \models \phi \\
(\mathcal{M}, s, \sigma) \models \forall x \phi & \textit{iff for all } u \in adom(s), (\mathcal{M}, s, \sigma^x_u) \models \phi
\end{array}
$$

where the relation R_α is defined by mutual recursion as

$$
\begin{array}{lll}
R_{v:=t}(s, s') & \textit{iff} & s'(v) = (s \circ \sigma)(t) \\
 & & \textit{and for every } v' \neq v, s'(v') = s(v') \\
R_{v:=?}(s, s') & \textit{iff} & s'(v) \in U \\
 & & \textit{and for every } v' \neq v, s'(v') = s(v') \\
R_{\alpha;\beta}(s, s') & \textit{iff} & \textit{for some } s'', R_\alpha(s, s'') \textit{ and } R_\beta(s'', s') \\
R_{\alpha \cup \beta}(s, s') & \textit{iff} & \textit{either } R_\alpha(s, s') \textit{ or } R_\beta(s, s') \\
R_{\alpha^*}(s, s') & \textit{iff} & R^*_\alpha(s, s'), \textit{ where } R^*_\alpha \textit{ is the reflexive and} \\
 & & \textit{transitive closure of } R_\alpha \\
R_{\phi?}(s, s') & \textit{iff} & (\mathcal{M}, s, \sigma) \models \phi \textit{ and } s' = s
\end{array}
$$

We remark that the interpretation of actions is as standard in first-order dynamic logic. In particular, atomic assignments $v := t$ and $v :=?$ update variable v with the value of term t, or, respectively, of any value in U.

As regards formulas, differently from [Harel *et al.*, 2000], quantification is restricted to the active domain $adom(s)$ in a given state s. This is a typical assumption in the theory of databases [Abiteboul *et al.*, 1995], which we will show to be sufficient for our expressivity purposes, as it is the case in formulas (1) and (2).

We say that formula ϕ is *true* at state s of DaS $\mathcal{M}$, or $(\mathcal{M}, s) \models \phi$, iff for every interpretation σ, $(\mathcal{M}, s, \sigma) \models \phi$; while ϕ is *true* in $\mathcal{M}$, or $\mathcal{M} \models \phi$, iff for all $s \in \mathcal{M}$, $(\mathcal{M}, s) \models \phi$. As an example, the formula

$$[a :=?][b :=?](a < b \rightarrow \langle c :=? \rangle (a < c \wedge c < b))$$

is true for every dense linear order $<$, in particular for the structure $\langle \mathbb{Q}, \leq \rangle$ of the rational numbers seen as a DaS $\mathcal{M} = \langle \{a, b, c\}, \emptyset, \mathbb{Q} \rangle$. On the other hand, the formula

$$[a :=?][b :=?](a < b \rightarrow \exists x(a < x \wedge x < b))$$

is not true in general on dense orders, as parameter x ranges over the finite set $adom(s) = \{s(a), s(b), s(c)\}$ of active elements. As a consequence, wildcard assignments and first-order quantification express different notions in our semantics for FO-DL.

To investigate their relationship further, notice that if formula ϕ does not exhaust the finite set V of variables, then standard universal quantification, ranging on U rather than $adom(s)$, is captured by formula $[v := ?]\forall x(x = v \to \phi)$, with the proviso that variable v does not occur in ϕ. However, since set V is finite, the translation scheme above is not applicable in general. In particular, first-order quantification is always restricted to the active domain.

In Example 1 we informally discussed the verification of programs against specifications (1) and (2). More precisely, we now state the model checking problem for this setting.

Definition 4 (Model Checking Problem) *Given a DaS $\mathcal{M}$ and formula ϕ in FO-DL, determine whether $\mathcal{M} \models \phi$.*

Since $\mathcal{M}$ is an infinite-state system in general, Def. 4 assumes some finitary representation of $\mathcal{M}$, in particular of the interpretation domain U. This can be achieved in standard ways [Bagheri *et al.*, 2013; Belardinelli *et al.*, 2014].

Then, verifying the correctness of the swap and max programs in Example 1 when the variables are assigned real values is tantamount to model checking formulas (1) and (2) on all states interpreting the variables on the reals $\mathbb{R}$. Notice that in this case $\mathcal{M}$ has infinitely many states: it is therefore not obvious whether the model checking problem is decidable.

2.1 Example: Auctions

In this section we model a toy example of an English (ascending bid) auction as a DaS. We assume familiarity with English auctions, in which a number of bidders bid for one or more items put on offer by an auctioneer. Bidders are assumed to be rational, in particular they have a true value up to which they are ready to bid. The auctioneer keeps track of the bids and, at the end of the bidding phase, she assigns the item to the bidder with the highest offer. We refer to [Easley and Kleinberg, 2010] for a detailed presentation of English auctions.

To represent the scenario above as a DaS with n bidders, we assume that the set V contains variables bid_i and t_value_i for every bidder $i \leq n$, respectively representing the current bid and the true value of bidder i; moreover, we suppose that V contains a variable $high$ for the highest offer so far. All these variables range over the rationals $\mathbb{Q}$. Also, we consider a variable $time\_out$ to terminate the bidding phase, taking values 0 and 1. We then model an English auction for n bidders as a DaS $\mathcal{M} = \langle V, \{0, 1\}, \mathbb{Q}\rangle$.

Then, we can specify the bidding cycle by means of the following action α:

```
$time\_out := 0$;
while ($time\_out = 0$) do {
        $\bigcup_{i \leq n}$ ($bid_i := ?$;
                if ($bid_i \leq t\_value_i$) $\wedge$ ($high < bid_i$)
                then $high := bid_i$)
        $\cup$ $time\_out := 1$ }
```

Action α consists in an initial assignment of false (0) to $time\_out$, and then a `while` ψ `do` β loop (expressible in dynamic logic as $(\psi?; \beta)^*; \neg\psi?$), whose exit condition is that the bidding process has timed out (by means of the non-deterministic assignment $time\_out := 1$). Within the loop, a bidder is chosen non-deterministically to pass a bid via a wildcard assignment $bid_i := ?$. If the bid is comprised between bidder i's true value and the highest bid so far, then it becomes the new $high$.

By means of action α we can specify in FO-DL a number of properties of auctions. For instance, we might want to say that *throughout the auction, for every bidder, the values of bids are comprised between the highest offer so far and her true value*:

$$\bigwedge_{i \leq n} [\alpha]((bid_i \leq t\_value_i) \wedge (high < bid_i)) \tag{3}$$

Moreover, in FO-DL we can also compare values for the same variable at different points in the bidding process. As an example, we can express that *true values do not change during the auction*:

$$\bigwedge_{i \leq n} \forall x((x = t\_value_i) \to [\alpha](x = t\_value_i)) \tag{4}$$

In (4) parameter x allows us to compare the value of t_value_i before and after the execution of α.

Finally, in FO-DL we can specify that *the value of the highest bid does not decrease throughout the auction*:

$$\forall x((x = high) \to [\alpha](x \leq high)) \tag{5}$$

Formulas (3)-(5) are simple specifications that nonetheless allow us to make comparisons between the values of variables at different points of the system's execution, which are typically not expressible in propositional languages. However, to verify properties such as (3)-(5), we cannot rely on standard model checking techniques because we are working with data-aware systems, for which the model checking problem is known to be undecidable in general [Deutsch *et al.*, 2009]. In the next section we will develop a verification method adapted to the present framework and language.

3 Decidability via Finite Abstractions

In this section we introduce an abstraction-based technique to prove that the model checking problem in Def. 4 is indeed decidable. Then, we apply this result to the particular scenario of English auctions of Section 2.1. In the rest of the section we assume a (possibly finite) abstract domain $U^A \supseteq C$.

Definition 5 (Abstract State) *Given a (possibly finite) abstract domain $U^A \supseteq C$, an* abstract state *is a function $s : V \to U^A$ together with a total order $\leq_s$ defined on U^A that extends the restriction $\leq |_C$ of $\leq$ on C.*

Notice that, differently from the standard states in Section 2, the total order $\leq_s$ depends on the particular abstract state s. In particular, for different s and s', $\leq_s$ may be different from $\leq_{s'}$. Nonetheless, each $\leq_s$ is an extension of restriction $\leq |_C$, that is, the order $\leq$ on C is preserved. Hence, for every $c, c' \in C$, $c \leq_s c'$ iff $c \leq_{s'} c'$. Total extensions (for a finite U^A) can be found constructively, using topological sorting algorithms [Cormen *et al.*, 1990]. Hereafter, we do not discuss this matter further, and assume that suitable total extensions are always at hand.

Further, given a DaS $\mathcal{M} = \langle V, C, U \rangle$ and $U^A \supseteq C$, let the *abstraction* of $\mathcal{M}$ be the DaS $\mathcal{M}^A = \langle V, C, U^A \rangle$. Next, we introduce a notion of *isomorphism* between states and their abstractions.

Definition 6 (Isomorphism) *A state $s \in \mathcal{M}$ and an abstract state $s' \in \mathcal{M}^A$ are* isomorphic, *or $s \simeq s'$, iff for some bijection $\iota : adom(s) \to adom(s') \cup C$,*

(i) *$s' = \iota \circ s$;*

(ii) *ι is the identity on C;*

(iii) *ι preserves $\leq$, that is, for every $u, u' \in adom(s)$, $u \leq u'$ iff $\iota(u) \leq_{s'} \iota(u')$.*

Any bijection ι as above is a witness *for $s \simeq s'$, or $s \overset{\iota}{\simeq} s'$ for short.*

Intuitively, isomorphic states share the same relational structure. Observe that, as regards order $\leq_{s'}$, witness ι preserves the interpretation of individuals in the active domain $s'(V)$ only (as well as the constant in C). This feature of isomorphisms is key to obtain finite abstractions.

Now we show that isomorphic states satisfy the same first-order formulas ϕ as defined by the following BNF:

$$\phi \quad ::= \quad t = t' \mid t \leq t' \mid \neg\phi \mid \phi \to \phi \mid \forall x \phi$$

However, ϕ might contain free parameters interpreted outside the active domain. This remark motivates the following definition.

Definition 7 (Equivalent Interpretations) *Given a state $s \in \mathcal{M}$, an isomorphic abstract state $s' \in \mathcal{M}^A$, and a formula ϕ, the interpretations $\sigma : P \to U$ and $\sigma' : P \to U^A$ are* equivalent for ϕ w.r.t. s and s' *iff for some bijection $\theta : adom(s) \cup \sigma(fr(\phi)) \to adom(s') \cup \sigma'(fr(\phi))$,*

(i) *the restriction $\theta|_{adom(s)}$ is a witness for $s \simeq s'$;*

(ii) *$\sigma' = \theta \circ \sigma$;*

(iii) *for every $u, u' \in adom(s) \cup \sigma(fr(\phi))$, $u \leq u'$ iff $\theta(u) \leq_{s'} \theta(u')$.*

Again, as regards $\leq_{s'}$, a witness θ preserves only the order of individuals in $s'(V) \cup C \cup \sigma'(fr(\phi))$, which is a finite set. As customary in first-order logic, we can prove that equivalent interpretations on isomorphic states preserve first-order formulas [Abiteboul *et al.*, 1995]. We report this result for our particular setting.

Lemma 1 *Given a state $s \in \mathcal{M}$, an isomorphic abstract state $s' \in \mathcal{M}^A$, and an FO-formula ϕ, if interpretations σ and σ' are equivalent for ϕ w.r.t. s and s', then $(\mathcal{M}, s, \sigma) \models \phi$ iff $(\mathcal{M}^A, s', \sigma') \models \phi$.*

We now prove that Lemma 1 can be lifted to the full language FO-DL. To do so, we need one more assumption. Given total order $\leq$, the corresponding strict linear order $<$ is *dense with no endpoints* iff *(i)* for every $u, u' \in U$, $u < u'$ implies that for some $u'' \in U$, $u < u''$ and $u'' < u'$; and *(ii)* for every $u \in U'$, $u < u'$ and $u'' < u$ for some $u', u'' \in U$. It is well-known that every non-empty countable dense linear order without endpoints is order-isomorphic to $\mathbb{Q}$ [Roitman, 1990, Theorem 27]. Hence, hereafter we assume $U = \mathbb{Q}$ as interpretation domain.

We can now prove the main preservation result of this paper. Hereafter we write $R_\alpha(s) = \{s' \mid R_\alpha(s, s')\}$.

Theorem 2 *Let $\mathcal{M} = \langle V, C, \mathbb{Q} \rangle$ be a DaS with abstraction $\mathcal{M}^A = \langle V, C, U^A \rangle$. Then, consider a state $s \in \mathcal{M}$, an isomorphic abstract state $s' \in \mathcal{M}^A$, and an FO-DL formula ϕ. If $|U^A| \geq |V| + |C| + \#P(\phi)$, then for all interpretations σ and σ' equivalent for ϕ w.r.t. s and s',*

1. *if ϕ is of the form $[\alpha]\psi$, then*
   - *(a) for every state $w \in R_\alpha(s)$, there exists an abstract state $w' \in R'_\alpha(s')$, isomorphic to w, such that σ and σ' are equivalent for ψ w.r.t. w and w';*
   - *(b) for every abstract $w' \in R'_\alpha(s')$, there exists a state $w \in R_\alpha(s)$, isomorphic to w', such that σ and σ' are equivalent for ψ w.r.t. w and w'.*
2. *Further, $(\mathcal{M}, s, \sigma) \models \phi$ iff $(\mathcal{M}^A, s', \sigma') \models \phi$.*

Sketch of Proof. We prove both item 1 and 2 by mutual induction on the structure of ϕ. We start by proving item 1, specifically by induction on the structure of α. As for the base case, let $\alpha \equiv v := t$ and consider interpretations σ and σ' equivalent for ϕ w.r.t. s and s'. If $R_\alpha(s, w)$ then $w(v) = (s \circ \sigma)(t)$, while the interpretation of the other variables remains the same. Then consider abstract state w' such that $w'(v) = (s' \circ \sigma')(t)$ and w' coincides with s' on all other variables. Clearly w and w' are isomorphic, as s and s' are and σ and σ' are equivalent. In particular, w' can be chosen so that the total order $\leq_{w'}$ preserves order $\leq$ on $w'(V) \cup C \cup \sigma(fr(\psi))$: if t is either a variable or a constant, then the result follows by condition *(iii)* in Def. 6; if t is a parameter, then it follows by *(iii)* in Def. 7. Finally, $R'_\alpha(s', w')$ and by construction σ and σ' are equivalent for ψ w.r.t. w and w'.

Let $\alpha \equiv v :=?$ and again consider interpretations σ and σ' equivalent for ϕ w.r.t. s and s'. Let θ be a witness for $s \simeq s'$. If $R_\alpha(s, w)$ then $w(v) \in U$. Since $|U^A| \geq |V| + |C| + \#P(\phi)$, we can find a 'fresh' element $u' \in U^A \setminus (s'(V \setminus \{v\}) \cup C \cup \sigma'(fr(\phi)))$ and set $w'(v) = u'$, while w' coincides with s' on all other variables. Then define θ' as θ but $\theta'(w(v)) = u'$. Clearly, θ' is a witness for $w \simeq w'$ and $\sigma' = \theta' \circ \sigma$. Moreover, the total order $\leq_{w'}$ can be completed so that item *(iii)* in Def. 7 is satisfied. Thus, $R'_\alpha(s', w')$ and σ and σ' are equivalent for ψ w.r.t. w and w'.

The inductive steps for $\alpha \equiv \beta'; \beta''$ and $\alpha \equiv \beta' \cup \beta''$ are immediate. The case for $\alpha \equiv \beta^*$ can be proved by induction on the length of the path connecting states s and w such that $R^*_\beta(s, w)$, by using the induction hypothesis.

Finally, let $\alpha \equiv \chi?$. Then, $R_\alpha(s, w)$ implies that $(\mathcal{M}, s, \sigma) \models \chi$ and $w = s$. Since σ and σ' are equivalent for ϕ w.r.t. s and s', and $|U^A| \geq |V| + |C| + \#P(\phi) \geq |V| + |C| + \#P(\chi)$, we have in particular that σ and σ' are equivalent for χ (w.r.t. s and s'). By item 2 and the induction hypothesis we obtain that $(\mathcal{M}, s, \sigma) \models \chi$ iff $(\mathcal{M}^A, s', \sigma') \models \chi$, that is, $R'_\alpha(s', w')$ for $w' = s'$.

The proof of item 1.(b) is similar, we only notice that in the base case for $\alpha \equiv v :=?$ we use the fact that the linear order $<$ on $\mathbb{Q}$ is dense and without endpoints, to find $u \in \mathbb{Q}$ that simulates $w'(v) \in U^A$. We now move to prove item 2.

The base case for first-order formulas follows by Lemma 1; the inductive cases for propositional connectives are immediate. In particular, notice that, since $|U^A| \geq |V| + |C| + \#P(\neg\psi) = |V| + |C| + \#P(\psi)$ and $|U^A| \geq |V| + |C| + \#P(\psi_1 \wedge \psi_2) \geq |V| + |C| + \#P(\phi_i)$, for $i = 1, 2$, the induction hypothesis holds.

For $\phi \equiv \forall x\psi$, $(\mathcal{M}, s, \sigma) \models \psi$ iff for all $u \in adom(s)$, $(\mathcal{M}, s, \sigma^x_u) \models \phi$. If θ is a witness to the fact that σ and σ' are equivalent for ϕ w.r.t. s and s', then interpretations σ^x_u and $\sigma'^x_{\theta(u)}$ are equivalent for ψ (also w.r.t. s and s'). Moreover, $|U^A| \geq |V| + |C| + \#P(\phi) = |V| + |C| + \#P(\psi)$. Hence, the induction hypothesis holds and it follows that $(\mathcal{M}^A, s', \sigma'^x_{\theta(u)}) \models \phi$. As u is arbitrary and θ is a bijection between $adom(s)$ and $adom(s')$, we obtain $(\mathcal{M}^A, s', \sigma') \models \phi$.

Suppose that $\phi \equiv [\alpha]\psi$. As regards the $\Leftarrow$ direction, $(\mathcal{M}, s, \sigma) \not\models \phi$ iff for some $w \in R_\alpha(s)$, $(\mathcal{M}, w, \sigma) \not\models \psi$. By item $1.a$ there exists $w' \in R'_\alpha(s')$ such that w' is isomorphic to w, and σ and σ' are equivalent for ψ w.r.t. w and w'. Since $|U^A| \geq |V| + |C| + \#P(\phi) \geq |V| + |C| + \#P(\psi)$, by induction hypothesis $(\mathcal{M}^A, w', \sigma') \not\models \psi$, that is, $(\mathcal{M}^A, s', \sigma') \not\models \phi$. The $\Rightarrow$ direction is proved similarly, by using item $1.b$. □

By Theorem 2 a state s and its abstraction s' satisfy the same formulas in FO-DL, whenever the abstract domain U^A contains enough elements to mimick transitions from s. Most importantly, U^A can be assumed to be finite. However, the linear order $<$ on $\mathbb{Q}$ has to be dense, with no endpoints.

Furthermore, we say that DaS $\mathcal{M}$ and abstraction $\mathcal{M}^A$ are *bisimilar* iff *(i)* for every $s \in \mathcal{M}$, there is an isomorphic $s' \in \mathcal{M}^A$, and *(ii)* for every $s' \in \mathcal{M}^A$, there is an isomorphic $s \in \mathcal{M}$. As a consequence of Theorem 2 we obtain the following result.

Corollary 3 *Let $\mathcal{M} = \langle V, C, \mathbb{Q}\rangle$ be a DaS with bisimilar abstraction $\mathcal{M}^A = \langle V, C, U^A\rangle$. For every FO-DL formula ϕ, if $|U^A| \geq |V| + |C| + \#P(\phi)$ then $\mathcal{M} \models \phi$ iff $\mathcal{M}^A \models \phi$.*

In particular, whenever U^A is finite, the number of states in $\mathcal{M}^A$ is finite as well. Therefore, by Corollary 3 we can verify an FO-DL formula ϕ on the infinite-state DaS $\mathcal{M}$ by model checking the finite abstraction $\mathcal{M}^A$ (e.g. by translating to propositional DL).

Discussion. We briefly discuss the application of our results to the auction DaS $\mathcal{M}$ for n bidders in Section 2.1. Observe that for $\mathcal{M} = \langle V, \{1, 2\}, \mathbb{Q}\rangle$, we have $V = \bigcup_{i \leq n}\{bid_i, t\_value_i\} \cup \{high, time\_out\}$, and therefore $|V| = 2(n+1)$, $|C| = 2$. Further, for formula (3), we have $\#P(3) = n^2$. Hence, in order to verify (3) it is sufficient to consider an abstract domain U^A of finite cardinality $|U^A| \geq |V| + |C| + \#P(3) = n(n+2) + 4$. Then, formula (3) can be verified on such abstraction $\mathcal{M}^A$, and finally the result can be transfered to $\mathcal{M}$ in virtue of Corollary 3.

Furthermore, Corollary 3 gives us some useful insight on a problem related to model checking. Given a class $\mathcal{C}$ of DaS and an FO-DL formula ϕ, the *satisfiability problem* consists in determining whether ϕ is satisfied in $\mathcal{C}$, that is, true in some DaS belonging to $\mathcal{C}$. We remarked earlier that every non-empty countable dense linear order without endpoints is order-isomorphic to $\mathbb{Q}$. As a consequence, the satisfiability problem for the class of non-empty countable dense linear orders without endpoints can be reduced to the model checking problem for the rationals $\mathbb{Q}$. In particular, we obtain the following result.

Corollary 4 *The satisfiability problem of FO-DL formulas for the class of DaS whose interpretation domain U is a non-empty countable dense linear order without endpoints is decidable.*

In future work we plan to extend these results to other classes of DaS and to determine the exact complexity of the model checking and satisfiability problems.

4 Conclusions

In this paper we introduced a formal framework for Data-aware Systems inspired by (first-order) dynamic logic. The data content of a DaS is modelled by a finite set of variables (and constants) that can range over a possibly infinite interpretation domain. As a result, DaS are infinite-state systems in general (Section 2). We showed that DaS are rich enough to express elaborate multi-agent scenarios, including auction-based mechanisms (Section 2.1), and then we considered the verification problem w.r.t. the first-order extension FO-DL of dynamic logic. The main theoretical result of the paper consists in showing that, under specific conditions, the model checking problem for DaS w.r.t. FO-DL is decidable. The decidability result is obtained by defining finite, bisimilar abstractions that preserve the interpretation of FO-DL (Section 3). Hence, to verify an FO-DL formula ϕ in a concrete, infinite-state DaS $\mathcal{M}$, we build the finite abstraction $\mathcal{M}^A$, then model check ϕ on $\mathcal{M}^A$, and finally transfer the verification result to $\mathcal{M}$ by virtue of Corollary 3.

We envisage a number of extensions for the present contribution. Propositional Dynamic Logic is one of the success stories of formal methods applied to computer science, with a steady stream of fundamental contributions [Harel, 1984; Harel *et al.*, 2000]. On the other hand, its first-order extensions are much less investigated. In particular, the well-known undecidability results in first-order logic have spawned a wealth of contributions in mathematical logic and theoretical computer science [Börger *et al.*, 1997]. For first-order dynamic logic the situation is even more complex, as many well-known results that hold for first-order logic, fail for its dynamic counterpart, e.g., Löwenheim-Skolem theorem, completeness, compactness. However, in first-order dynamic logic there has not been a comparable push aimed at singling out decidable and tractable fragments. In future work we plan to individuate further classes of DaS and fragments of FO-DL that have a decidable model checking and satisfiability problems, and study their exact complexity. We deem results along this line valuable to improve the understanding of (first-order) dynamic logics, data-aware systems, and auction-based mechanisms.

Acknowledgments

F. Belardinelli acknowledges the support of the ANR JCJC Project SVeDaS (ANR-16-CE40-0021).

References

[Abiteboul *et al.*, 1995] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Bagheri *et al.*, 2011] B. Bagheri, D. Calvanese, G. De Giacomo, R. De Masellis, and P. Felli. Foundations of relational artifacts verification. In *Proceedings of the 9th Int. Conference on Business Process Management (BPM11)*, volume 6896 of *Lecture Notes in Computer Science*, pages 379–395. Springer, 2011.

[Bagheri *et al.*, 2013] B. Bagheri, D. Calvanese, M. Montali, G. Giacomo, and A. Deutsch. Verification of relational data-centric dynamic systems with external services. In *Proceedings of the 32nd Symposium on Principles of Database Systems (PODS13).*, pages 163–174. ACM, 2013.

[Baldoni *et al.*, 2016] M. Baldoni, C. Baroglio, D. Calvanese, R. Micalizio, and M. Montali. Towards data- and norm-aware multiagent systems. In M. Baldoni, J. Müller, I. Nunes, and R. Zalila-Wenkstern, editors, *Engineering Multi-Agent Systems - 4th International Workshop, EMAS 2016, Singapore, Singapore, May 9-10, 2016, Revised, Selected, and Invited Papers*, volume 10093 of *Lecture Notes in Computer Science*, pages 22–38. Springer, 2016.

[Belardinelli *et al.*, 2012] Francesco Belardinelli, Alessio Lomuscio, and Fabio Patrizi. An Abstraction Technique for the Verification of Artifact-Centric Systems. In *Proc. of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*, pages 319 – 328, 2012.

[Belardinelli *et al.*, 2014] F. Belardinelli, A. Lomuscio, and F. Patrizi. Verification of agent-based artifact systems. *Journal of Artificial Intelligence Research*, 51:333–376, 2014.

[Bhattacharya *et al.*, 2007] Kamal Bhattacharya, Cagdas E. Gerede, Rick Hull, Rong Liu, and Jianwen Su. Towards Formal Analysis of Artifact-Centric Business Process Models. In *Proc. of BPM*, 2007.

[Börger *et al.*, 1997] Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

[Calvanese *et al.*, 2016] D. Calvanese, M. Montali, F. Patrizi, and M. Stawowy. Plan synthesis for knowledge and action bases. In S. Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1022–1029. IJCAI/AAAI Press, 2016.

[Clarke *et al.*, 1999] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

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

[Cormen *et al.*, 1990] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 1990.

[De Giacomo *et al.*, 2012] Giuseppe De Giacomo, Yves Lespérance, and Fabio Patrizi. Bounded Situation Calculus Action Theories and Decidable Verification. In *Proc. of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*, pages 467–477, 2012.

[de Giacomo *et al.*, 2014] G. de Giacomo, Y. Lesperance, F. Patrizi, and S. Vassos. Progression and verification of situation calculus agents with bounded beliefs. In *Proceedings of the International conference on Autonomous Agents and Multi-Agent Systems (AAMAS14)*, pages 141–148. IFAAMAS, 2014.

[De Giacomo *et al.*, 2016] Giuseppe De Giacomo, Yves Lespérance, and Fabio Patrizi. Bounded situation calculus action theories. *Artif. Intell.*, 237:172–203, 2016.

[Deutsch *et al.*, 2007] Alin Deutsch, Liying Sui, and Victor Vianu. Specification and Verification of Data-Driven Web Applications. *J. Comput. Syst. Sci.*, 73(3):442–474, 2007.

[Deutsch *et al.*, 2009] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic Verification of Data-Centric Business Processes. In *Proc. of ICDT*, 2009.

[Easley and Kleinberg, 2010] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, USA, 2010.

[Harel *et al.*, 2000] David Harel, Jerzy Tiuryn, and Dexter Kozen. *Dynamic Logic*. MIT Press, Cambridge, MA, USA, 2000.

[Harel, 1984] D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, volume II: Extensions of Classical Logic*, chapter 10, pages 497–604. Reidel, 1984.

[Montali *et al.*, 2014] M. Montali, D. Calvanese, and G. De Giacomo. Verification of data-aware commitment-based multiagent system. In Ana L. C. Bazzan, Michael N. Huhns, Alessio Lomuscio, and Paul Scerri, editors, *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pages 157–164. IFAAMAS/ACM, 2014.

[Pratt, 1976] V. R. Pratt. Semantical considerations on floyd-hoare logic. Technical report, Cambridge, MA, USA, 1976.

[Roitman, 1990] J. Roitman. *Introduction to Modern Set Theory*. A Wiley-interscience publication. Wiley, 1990.

[Singh and Huhns, 2005] Munindar P. Singh and Michael N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. Wiley, 2005.