Foundations of Data-Aware Process Analysis: A Database Theory Perspective

Diego Calvanese, Marco Montali
Free University of Bozen/Bolzano
lastname@inf.unibz.it

Giuseppe De Giacomo
Sapienza Università di Roma
degiacomo@dis.uniroma1.it

ABSTRACT

In this work we survey the research on foundations of data-aware (business) processes that has been carried out in the database theory community. We show that this community has indeed developed over the years a multi-faceted culture of merging data and processes. We argue that it is this community that should lay the foundations to solve, at least from the point of view of formal analysis, the dichotomy between data and processes still persisting in business process management.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification

General Terms

Verification

Keywords

Business artifacts, data-centric processes, first-order temporal logics.

1. INTRODUCTION

When it comes to manage the assets of an organization, data and processes should be considered as two sides of the same coin [105]. A 2009 survey by Forrester[1] [90], whose outcome is also reported in [108], has addressed the important question of which of the two aspects should be given priority from the point of view of IT management. Unsurprisingly, the role played by an individual within IT strongly affects the perception of the relative importance of processes and data within an organization: Professionals concerned with the management of business processes downplay the importance of data, and view it as subsidiary to the processes that manage them; as a consequence, they do not pay attention to the quality of data and on how the business processes can ensure that data assets can be maintained clean. On the contrary, data management experts consider data as the driver of the processes in an organization, and assume that guaranteeing data quality is sufficient to ensure proper consideration of business relevant data so as to impact process improvement efforts.

An immediate consequence of this dichotomy, is that there is very limited collaboration and cost sharing between the teams on the one hand running the (master) data management (MDM) initiatives and on the other hand managing the business process. This is also confirmed by Forrester's survey, where for 83% of the respondents there was no interaction, and only in 8% of the cases the master data management and business process modeling efforts were fully coordinated. A further consequence is that there is little attention also on the side of tool vendors to address in their products the requirements coming from a combined treatment of processes and data. On the one hand, data management tool vendors consider processes only insofar as they affect the direct management of the data within the tools, but they do not pay attention to the processes that actually make use of the data. On the other hand, business process modeling suites do not allow for the connection of data to the processes. Service oriented architectures (SOA), which make it possible to divide the functionality of large systems into component services, are advocated as a solution to the data-process dichotomy. However, while favoring component reuse, they do not address the need of connecting the data to the organizational processes so as to facilitate their improvement, and in fact data continues to be "hidden" inside systems [100]. In addition to SOA, [100] identifies two key areas in which an explicit representation of data in process models is crucial. The first is the *modeling of the core assets* of an organization, due to the fact that the data stored in different IT systems is crucial for the execution of the business processes that create the value of the organization itself. Hence, the business processes depend on such data, and in order to keep the organization operational, the former need access to the latter. This dependency should be accounted for explicitly. The second is *business process controlling*, due to the fact that both the key performance indicators, and the business goals of the organization on which they depend, are defined in terms of data. To evaluate and control these indicators, the activities contributing to the goals need to be identified, and this is done by considering the appropriate data objects on which these activities operate. In order to support this task, process models need to shift the emphasis from control flow to the data [120, 85].

It follows that there is a strong need to incorporate data modeling features in (business) process modeling languages, and to enrich business process analysis tools to deal with data [100]. This demands for suitable modeling languages, methodologies and systems supporting the integrated management of processes and data, and, possibly above all, it calls for a more foundational approach,

[1] http://www.forrester.com/

PODS'13, June 22–27, 2013, New York, New York, USA.

to provide a clear semantics for (data-aware) process models, and to consequently enable their analysis.

Analysis is attracting a lot of interest based on the momentum that verification of software and hardware systems has had in the last 15 years, also recognized by the 2007 A.M. Turing Award, given to Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis, for their role in developing Model-Checking into a highly effective verification technology that is widely adopted in the hardware and software industries. Model checking, see [122] for an introduction oriented to database theoreticians, is based on the idea of formulating dynamic properties of interest in some temporal logic like LTL, CTL, μ-calculus [79, 117] (whose temporal component is intrinsically non-first order, since based implicitly or explicitly on forms of fixpoints) and check such formulas over the transition system (explicitly or implicitly represented) mathematically capturing the dynamics of the system of interest. A key element for current model checking techniques is that states can be modelled propositionally, giving rise to a finite-state transition system.

When data are relevant, states need to be model relationally rather than propositionally [122]. That is, we associate to the state of the process the state of the data, possibly seen as a relational database. These transition systems are typically *infinite-state* since there is no bound on the number of tuples that can be added to database relations as the computation goes on. The presence of data also calls for query languages for process analysis that combine two dimensions: a *temporal dimension* to query the process execution flow, and a *first-order dimension* to query the data present in the relational structures maintained by the states of the system, and to relate objects across different states. In other words, first-order variants of temporal logics are required [122, 126, 69].

The resulting verification problem is much harder than in the pure finite-state control-flow setting, and deeply challenges the possibility of employing off-the-shelf, conventional finite-state model checkers. In particular, a data-aware process that combines a finite control-flow with the manipulation of a full-fledged database, can easily encode the behavior of a Turing machine, causing the model checking problem to become immediately undecidable even for simple propositional CTL/LTL properties.

The fundamental question is then: how can we mediate between the expressiveness of the temporal property language, and the identification of classes of data-aware processes, for which analysis becomes decidable, but at the same time still applicable to notable, real-world data-aware processes? Research in verification has tacked verification of infinite-state systems (e.g., see [50] for a survey). However, in much of this work the emphasis is on studying recursive control rather than data, which is either ignored or propositionally abstracted. If data are included they are of a very specific form, like recursive procedures with integer parameters [48], rewriting systems with data [47], or Petri nets with data associated to tokens [95].

Processes and data has been a continuously present stream of research in database theory. Over the years, a lot of work has been done on database evolution and transaction (see Section 3.1), on temporal query languages and data management (see Section 3.2), on active databases (see Section 3.3), on workflow systems (see Section 3.4), and on temporal integrity constraints (see Section 3.5). Most of this work looks at dynamics of a database system, however starting from the work on relational transducers (see Section 4.1), business processes, i.e., processes at a higher level of abstraction, have started attracting attention. Then a first call-to-arms was issued in early 2000 by Rick Hull in [86] concerning the need of modeling and analyzing business processes in the context of e- or web-services. The interest in web-services gave rise to a beautiful stream of work on verifying database-centric dynamic services (see Section 4.2). A second call-to-arms was issued again by Rick Hull in the late 2000, about the modeling and the formal analysis of "artifact-centric business processes" [85]. It is this call that has generated the latest work on data-aware process analysis that has been flourishing in the last years (see Sections 4.3 and 4.4).

In this work we present an overview of the research on foundations of data-aware (business) processes that has been carried out in the database theory community in the last three decades. We show that this community has indeed developed over the years a multifaceted culture of combining static and dynamic aspects of data management, which has recently culminated in a series of significant lines of research addressing the foundations of data-aware process analysis. We argue that it is this community that should pursue further the investigation of the fundamental issues underlying the dichotomy between data and processes, which still persists in business process management and calls for a unifying, well-founded framework.

This survey complements four key companion surveys in the area: the ones by Rick Hull [86, 85], which single out problems and challenges on data-aware service oriented and artifact-centric computing; the one by Moshe Vardi [122], which presents the body of work on model checking, including challenges arising due to the presence of data; the one by Victor Vianu [126], which surveys the line of work developed starting from mid 2000 on verification of data-centric dynamic services; and the one by Tova Milo [69], which surveys how fundamental data management techniques can be applied to the challenging problem of managing control flows characterizing business processes.

2. THE BPM PERSPECTIVE

Process analysis is a central research theme in business process management (BPM). In a recent survey [118], Wil van der Aalst pointed out that *process model analysis* has been the second most influential topic in a decade of BPM conferences (following *process modeling languages*). However, in BPM process analysis has been mainly tackled, so far, by following a *divide et impera* approach. This has led to the development of sophisticated, effective techniques dealing with the process, control-flow dimension but abstracting away from the data. In particular, a plethora of verification techniques has been developed to verify whether the control-flow of a process meets specific, pre-defined properties (such as absence of deadlocks, boundedness, and soundness), or domain-dependent properties. Virtually all these techniques rely on the fact that the dynamics induced by a process control-flow can be captured by means of a (possibly infinite-state) propositional *labeled transition system*, whose labels represent the process tasks/activities, and where concurrency is represented by interleaving, as typically done in formal verification [27]. Usually, such a transition system is not explicitly represented, but it is instead implicitly "folded" into a Petri net, which provides a compact representation of the process control-flow thanks to its native capability of accommodating concurrency.

Beside verification of pre-defined properties, also the verification of arbitrary, domain-dependent properties has been tackled in BPM, relying on standard temporal logics such as CTL and LTL. In particular, such temporal logics are exploited to specify properties about the dynamics of the (Petri net representing the) system by either focusing on place configurations (i.e., the amount of tokens present in a given place), or on task execution [77]. In spite of the undecidability results for verification of general Petri nets [76, 77], decidability holds for safe/bounded nets, whose reachability graph consists of a finite-state labeled transition system. This, in turn,

makes it possible to rely on conventional finite-state model checking techniques, lifting the focus from decidability to complexity issues [127, 109, 110, 23].

When it comes to formal specification and analysis of data-aware processes, no satisfactory solution has been provided so far within BPM. The main approach that captures data-aware extensions of Petri nets is the one of colored Petri nets. However, in the instantiation used in BPM [119] they are not suited to represent a full-fledged database. They introduce data as variables associated to tokens, and manipulate them by means of a sort of *procedural attachment*, i.e., by attaching procedures/functions to the transitions of the net. These procedures/functions can be implemented as an arbitrary program (typically written in a functional language), and hence are completely unconstrained. This lifts the system towards executability[2], but sacrifices its analyzability and verifiability.

High-level, business process modeling languages such as the OMG standard BPMN[3], and the OASIS standard BPEL[4] service orchestration language suffer from similar limitations when the data dimension is taken into account. All these languages largely leave the connection between the process dimension and the data dimension underspecified. For example, they do not conceptually capture the behavior of atomic tasks, consequently abstracting away from how they progress data. To obtain a fully-specified model, one therefore needs again to attach an arbitrary program to every BPMN atomic task or service invoked by the BPEL process [102].

3. DYNAMICS IN DATABASE THEORY

We overview here how the database theory community has been contributing to the analysis of data-aware processes. We do so by first looking at some key lines of research that have considered the interaction of both static and dynamic aspects of data management. Specifically, we consider below the following lines of research:

1. database evolution and transactions;
2. temporal data management;
3. active databases;
4. workflow formalisms and systems;
5. temporal integrity constraints.

For each of these areas we overview the main research objectives and achievements. Our aim here is not to be comprehensive, but rather to highlight the mainstream directions relevant to the topic of this paper that have characterized the research in databases.

3.1 Database Evolution and Transactions

The problem of evolution of data in a database by means of atomic operations and their combination inside transactions has been considered from early on as a key issue to investigate in databases. Apart from the fundamental problems of concurrency control and serializability (see, e.g., [93, 112, 98, 82] for early results), updates and transactions have been considered also in view of their interaction with (static) database constraints. Equivalence and optimization of relational transactions, consisting of linear sequences of insertions, deletions, and updates, using simple selection conditions based on individual attribute values for each tuple, is investigated in [13, 17]. A formal model (called *dynamic relational model*) for evolution over time of a database, seen as a sequence of instances is presented in [123, 124]. The effects on evolution of dynamic constraints (specifically, dynamic functional dependencies), which relate one database instance to the next in the sequence are studied. Specifically, the problem of inferring static constraints from knowledge about the evolution history of the database, as expressed by the dynamic constraints, is investigated. The impact of dynamic constraints on the update of a specific form of views, in which each tuple represents an object with its properties, is considered in [125].

The connection between transactions and static constraints is further investigated in [14], which presents a model where valid database states are described using a set of admissible (parameterized) transactions, as opposed to constraints. Such *transactional database schemas* are in general incomparable with schemas described via constraints, though they are able to simulate natural types of constraints, such as those generated by the early semantic database models (ER [53], IFO [7]). Equivalence of transactional schemas is shown to be undecidable in general, but decidable cases are singled out. The work studies also preservation of constraints by transactions, showing decidability, e.g., for inclusion dependencies, but undecidability for arbitrary FOL constraints. Further decidable restrictions, investigated in [15], are obtained by limiting on the one hand the kind of allowed operations to insertions and deletions (but no updates), and on the other hand the properties to be checked to specific ones (in line with what typically done in software verification). Interestingly, when the maximum length of a transaction is bound to a fixed value, decidability can be shown for stronger properties.

In the transaction language TL introduced in [16], which features inserts, deletes, and a (non-deterministic) "while" construct, transactions may use a fixed number of temporary relations, and may be "unsafe", i.e., introduce *new* values, not in the original database. Safety is also relaxed to "weak safety", where new values are allowed only in temporary relations. Further, a notion of "update completeness", which is more natural than query completeness [52], is proposed, and it is shown that TL is update complete. [18] builds on this work, by defining a variant of TL in which "while" has a deterministic semantics, and new values may not appear in the result, but only in intermediate relations. It is shown that such language is complete for deterministic updates. The variant where new values are disallowed alltogether, is shown complete for fixpoint queries, hence strictly less expressive that PSPACE updates. Instead, on an ordered domains, one obtains exactly PSPACE. The work presents also declarative update languages, which are extensions of Datalog with negation in the body, disjunction in the head, and unsafe head variables. Derivation of a fact corresponds to insertion, whereas deletions are not foreseen. Such language is equivalent to TL without deletion, and is complete for non-deterministic updates where input and output relations are disjoint.

3.2 Temporal Data Management

A temporal database provides mechanisms to store data as it evolves, and to query its historical states using suitable extensions of standard query languages like SQL. Research on temporal database originates from the observation that temporal data management can be very difficult if one uses conventional database systems [88]. The work on temporal databases and query languages goes back to [111], which provides a Description of syntax and semantics of temporal extension of Quel (a calculus-based query language for the Ingres system) that makes use of Allen's interval relations [22].

A temporal database model, in which each tuple is timestamped with a union of time intervals, is defined [80]. The notion of "weak relation" as the equivalence class of all timestamped relations for which the snapshots at each timepoint are equal, is introduced, and an algebra over such weak relations is defined and studied. Datalog

[2] http://cpntools.org/
[3] http://www.bpmn.org/
[4] https://www.oasis-open.org/committees/wsbpel

extended with unary function symbols (i.e., successor), is studied in [57], and a mechanism is proposed to finitely represent infinite query answers via rules that may be returned together with explicit tuples. A framework for reasoning about infinite temporal information, based on generalized tuples with additional temporal attributes and constraints, is presented in [89]. Temporal attributes are defined by infinitely repeating points (of the form $z(n) = c + kn$) and constraints are conjunctions of linear equalities and inequalities on temporal attributes. Contrast this to constraint databases, where constraints are used to describe multiple databases, as opposed to a single database with infinite temporal information. The paper relates predicates definable by generalized relations with those definable in Presburger arithmetic. It studies the complexity of relational algebra on generalized relations, which return finite representations of possibly infinite answers. Whereas positive existential queries are in PTIME (in data complexity), arbitrary queries (with negation) are NP-hard and in 2EXPTIME.

The semantics and expressive power of Templog [1], which extends horn logic programs with temporal operators (next, always, eventually) is studied in [32, 33], showing that the declarative semantics and the operational semantics, based on a suitably defined temporal extension of SLD-resolution, coincide. The resolution-based calculus is shown to be sound and complete, but restrictions that would ensure decidability of satisfiability of Templog specifications are not considered. For the propositional variant of Templog expressiveness is investigated, and shown to be equivalent to that of *fixpoint linear time logic* (i.e., Büchi automata on infinite strings) [121] restricted so as to allow only least fixpoints applied to positive formulas, which in turn corresponds to finite-word regular languages. The temporal database formalisms proposed in [57, 32, 89] are compared in [34] with respect to their power in expressing queries (query expressiveness) and constraints on the data (data expressiveness). Also a query language, operating on temporal databases [89], with the ability to express predicates over multiple temporal attributes it defined, and sufficient conditions for finite evaluation of queries, even over infinite periodic data, are given.

Several works survey the area of temporal databases and query languages. In particular, [55] surveys temporal query languages, distinguishing between abstract and concrete languages and studying formal semantics, expressiveness, and query processing. In [88], instead, the problem of temporal data management is addressed more in general, also considering design and implementation aspects for temporal database systems.

We conclude by observing that the work on temporal query languages includes also work on logics that are first-order variants of propositional temporal logics used in verification, such as first-order LTL [54] and others that we will consider later. The formulas in such logics are *monadic* in the temporal component, in the sense that they can be seen as a combination through temporal operators of open formulas that contain only one free temporal variable. However, temporal query languages go beyond these logics, allowing also for combining formulas that are not monadic in the temporal component, so as to express sophisticated relationship among data and associated timestamps [6].

3.3 Active Databases

At the end of the 80's there was a great interest in active databases. These are processes described in terms of sets of event-condition-action rules operating on a database: such rules are triggered (under the control of a condition) by an initial external event, and successively by the internal update events caused by the execution of the rule actions. Much of the research in this area has been devoted to study the execution of such languages, by defining and implementing interpreters for them [91]. A general formal framework for active databases is introduced in [103, 104], and is used as the basis for comparing several active database prototypes with respect to their expressive power and complexity.

Substantial work has also been devoted to the analysis, especially termination and confluence of rule based trigger systems [21]. The termination problem can be formulated as follows: given a set of rules, check whether, for every initial database (and every possible triggering event), every sequence of rule activations (which are in general non-deterministic) eventually terminates. Notice that, although rules are in general assumed to *not* bring in new data into the system (but just to manipulate the objects present in the initial database), the difficulty of the problem lies in the fact that termination has to be determined for every possible initial database.

The undecidability of termination in general was immediately observed. However, special cases and restrictions that guarantee decidability have been singled out. For example, [29] shows decidability of termination of *propositional* trigger systems in which the triggered rules are managed via a stack or a set. Such decidability results turn out, however, to be rather fragile; e.g., the system studied in [29] becomes undecidable when a stack replaces the queue in the management of the triggered rules. When going beyond the propositional case and considering relations updated by the rule actions, one approach, followed by [28], is to perform the analysis by an (unfaithful, i.e., sound) abstraction of data, so as to guarantee soundness (but not necessarily completeness) of the termination check. Instead, [30] aims at sound and complete techniques, and investigates the borders of decidability for the termination problem. It is shown that in the following two cases, which impose rather severe conditions on the specification of the systems, decidability holds: *(i)* updates are restricted to have a single atom in rule bodies and to be safe (i.e., all head variables appear in the body); *(ii)* updates may be arbitrary, but only unary relations may be non-empty in the initial database. In both cases, even apparently minor relaxations of these restrictions lead to undecidability.

3.4 Workflow Formalisms and Systems

A further topic integrating static and dynamic aspects of data that has been addressed by the database community is that of systems to manage workflows. A workflow can be considered as a collection of activities designed in such a way that a group of (human or artificial) agents can carry out in a coordinated way a specific complex process. Workflow management systems provide a framework for capturing the interaction among the activities in a workflow [116].

Research in databases has contributed to this area by studying formalisms and systems that would support transactional aspects of workflows [49, 46]. Specifically, [49] considers a setting that deals both with task dependencies in a workflow, and dependencies between operations on the data by multiple interacting transactions. However, it does not consider the actual data and the changes performed on it, but only the order in which operations are executed.

Another interesting perspective on workflows is the use of typical database functionalities (persistence, transactions, complex querying, provenance, etc.) to support the activities related to managing workflows and their execution [107, 35, 36, 31, 64]. The recent survey [69] contains an in-depth treatment of this aspect.

The importance of data not only in the context of a single workflow, but to drive the integration between multiple, inter-organizational workflows, has been considered since the late nineties in the Vortex workflow management system [87]. In Vortex, data implicitly introduce additional dynamic (data-flow) constraints among activities belonging to the different interacting workflows. Verification of Vortex workflows has been studied by

considering the control-flow component, but by considering the contribution of the data component only in terms of the induced data-flow constraints, without explicitly capturing the complex interplay between the two components [78].

The dichotomy between an expressive workflow modeling language able to account for data, and the language used for verification, which abstracts away data, is present also in other approaches. For example, [65] adopts transaction logic with task pre-/post-conditions to model full-fledged workflows operating over relational databases, but forbids the presence of such conditions when it comes to reasoning and verification, thus effectively loosing the link between the workflow behavior and the database.

3.5 Temporal Integrity Constraints

Temporal integrity constraints are integrity constraints that collectively constrain multiples states of a database over time. Even though they are not meant to explicitly represent a business process, they nevertheless declaratively specify which evolutions of the system and of the corresponding data are considered legal.

Early on, temporal constraints were recognized important for transactions. [96] relates dynamic integrity constraints expressed in temporal logic to transaction specifications defined by FOL pre and postconditions. The aim is to monitor the integrity constraints through transactions, and it is shown how to align the transaction specification so that it generates only state sequences that satisfy the dynamic constraints. Later, [39] investigates the properties of weakest preconditions for various transaction and specification languages, concentrating on specification languages that are relevant to integrity constraints, such as FOL.

Most of the work on checking temporal constraints in temporal databases focused on first-order variants of LTL [54], which is in general highly undecidable [83]. However, a decidability condition that was singled out early on concerns safety formulas with no quantification except for implicit universal quantification allowed outside temporal operators [58, 59]. First-order LTL that uses only temporal connectives referring to the past has been proposed as a convenient language to express integrity constraints, since such formulas can be efficiently checked, by accumulating the historical information that is necessary for the check in auxiliary relations of the current state [56].

4. BUSINESS PROCESS ANALYSIS IN DATABASE THEORY

We next turn to research that directly tackle data-aware process analysis, focusing on higher level processes than transactions, such as business processes. We recall that the presence of data on the one hand makes the system dynamics infinite-state in general and, on the other hand, requires to go beyond propositional temporal logics. Indeed, in order to properly query the state of the system by extracting data, one needs first-order quantification within and across the states of the system.

Various approaches have been proposed, that differ in:

- The structure of the process component, as well as its interaction with the data component, and with the external environment.
- The kind of analysis problem that is considered; most works focus on verification of arbitrary temporal properties expressed in the adopted formalism; other approaches fix a set of specific problems they aim to solve.
- The considered temporal formalism, and consequently the kind of properties that can be expressed; such properties are typically formulated in a variant of first-order temporal logic.
- The form of quantification that is allowed in the first-order temporal formalism, where the cases that have been considered are quantification over the initial state only, and quantification across states. The latter case takes into account also those objects that have been introduced during the evolution of the system, and in general requires suitable restrictions on the scope of the quantification so as to guarantee decidability.

4.1 Relational Transducers

One of the most significant approaches proposed by the database community to model high-level (business) processes is that of relational transducers, originally proposed to support forms of e-commerce [19, 20]. Relational transducers explicitly account for a dynamic component, reminiscent of active databases and transactional workflows, on top of full-fledged databases.

More specifically, a relational transducer is a tuple (S, σ, ω), where: *(i)* S is the relational transducer schema, constituted by pairwise disjoint relational schemas for input, state, output, and fixed (external) database, and where the log is a further relational schema used to maintain the semantically meaningful portion of an input-output exchange; *(ii)* σ is a state-update transition function mapping instances of input, state and fixed database to instances of the next state; *(iii)* ω is an output-update transition function mapping instances of input, state and fixed database to instances of the next output. The semantics of a relational transducer is based on linear time. In particular, it captures the evolution of state and output sequences, in response to a sequence of inputs representing the interaction with the external world, as stored in the log.

The problems subject to analysis range from log validity (i.e., checking whether a log sequence can be generated with some input sequence), goal reachability, containment (i.e., testing whether every valid log of one transducer is also valid for another) and compatibility (i.e., checking whether two transducers have a common log) to the verification of specific first-order temporal properties with a past-time operator. These problems are in general undecidable. However, decidability and, in particular, a NEXPTIME upper bound for the verification problem, have been obtained in [19, 20] by requiring transducers to be *semi-positive cumulative state* (Spocus). In a Spocus transducer, the state accumulates all inputs received, and the outputs are defined by a non-recursive, semi-positive set of datalog rules.

In [113, 114], a generalization of Spocus transducers, called ASM transducers, has been studied. ASM transducers do not necessarily accumulate input, and their rule application is in general guarded by arbitrary first-order formulas. In this setting, the aforementioned problems are reconsidered, and verification is addressed for a variant of first-order LTL in which temporal operators cannot appear in the scope of first-order quantifiers, except for outermost universal quantifiers. Even though verification is undecidable for general ASM transducers, two main restrictions that guarantee decidability are identified: *(i)* ASM transducers for which the fixed (external) database is explicitly known, and the set of values allowed in the input is restricted to those appearing in that database; *(ii)* ASM transducers which bound a-priori the maximum amount of input that can be received in one computation step. Complexity of verification for such restricted versions range between PSPACE-complete to EXPSPACE-complete, depending on whether the maximum arity of the employed relations is bounded a-priori or not.

4.2 Data-Driven Web Systems

Web systems provide distributed access to information stored on the web, typically powered by databases and manipulated by complex web applications/services that interact with third-party ser-

vices and external users. In [72], the case of a single web service that interacts with an external user is considered, studying the trade-off between the expressiveness of the web service specification language and the feasibility of verification. Verification is tackled by relying on a formalization of such kind of system in terms of a model that extends ASM transducers. In fact, a web service is modeled by means of *(i)* a database that remains fixed during the execution, *(ii)* a set of state relations that evolve in response to user inputs, *(iii)* a set of web page schemas that query the current database and state to generate user input choice, and *(iv)* state transitions that are triggered by the input chosen by the user. The firing of a transition triggers actions to be taken for progressing the state, then leading the interaction to the next web page.

To achieve decidability of verification, [72] imposes restrictions over the web service similar to [113, 114], limiting the use of quantification in state, action, and target rule formulas to *input-bounded quantification*, and limiting formulas of input rules to be existential. The expressive power of ASM transducers is extended with the possibility of referring to the input at the previous step in the run. Verification is then tackled for the input-bounded version of the linear-time logic considered in [113, 114], extended with the possibility of referring to the previous input. In particular, decidability of verification for this logic over an input-bounded web service is decidable in EXPSPACE (in fact, PSPACE-complete when the arity of the service schema is bounded a-priori). This result is obtained by a reduction to finite satisfiability of existential first-order logic augmented with a transitive closure operator. Furthermore, it is shown that even small relaxations of the imposed restrictions lead to undecidability. Beside linear-time properties, also verification of branching-time properties is tackled, by considering variants of the logics CTL and CTL*, where first-order quantification obeys to the same restrictions as in the LTL case. In particular, it is shown that for these logics it is necessary to further restrict the web service model to guarantee decidability. Two main restrictions are studied: *(i) propositional, input bounded web services*, which forbid the use of previous input relations and pose the strong assumption that all states and actions are propositional (but inputs are still parametrized in the specification); and *(ii) input-driven search web services*, which restrict the usage of previous input relations but are still able to capture common applications involving a user-driven search. In the case of propositional web services, verification for the CTL and CTL* variants is proved to be respectively in coNEXPTIME and EXPSPACE (PSPACE by fixing the database schema). Complexity of verification respectively becomes in EXPTIME and 2EXPTIME in the case of input-driven search web services.

Notably, even though the established complexity upper bounds provide no indication about the practical feasibility of verification, an effective implementation has been carried out in the WAVE system [71], focusing on the linear-time case. In particular, the experimentation carried out in [71] has demonstrated that, by leveraging on a fruitful coupling of novel verification and database optimization techniques, complete verification is practically feasible for a reasonably broad class of applications. A system building on WAVE and dealing with aspects ranging from specification of Web applications to explanation of verification results is presented in [74].

In [75, 73], the single-service setting tackled in [72, 71] has been extended to the case of composition of web-services (also called peers), which interact by asynchronous message exchange. A service reacts to incoming messages and user input by updating its internal state through a function of the current contents of the database, state, user input, and received messages, possibly replying with outgoing messages. Decidability of verification is studied for two property specification formalisms: the variant of first-order LTL studied in prior work, and conversation protocols (that leverage on an industrial standard). In particular, various semantics for message-based communication are exploited (singleton versus set messages, lossy versus perfect communication channels, bounded versus unbounded received message queues), which provide various extensions to the notion of input-boundedness in the case of multiple interacting services. Under appropriate communication semantics, decidability of verification is established in PSPACE, showing at the same time that even slight relaxations of the imposed restrictions immediately lead to undecidability.

Related to this research line is also the work on the Colombo framework for service composition [40]. There automated composition synthesis is studied in presence of data. In particular, decidability under certain conditions that ensure reduction of relational states to propositional states is established. The restriction that along a run only finitely many new object are introduced is crucial for the technique proposed therein. Notice that this input-bounded condition guarantees that service runs are "bounded", in the sense discussed in Section 4.4.

4.3 Artifact-Centric Systems

The artifact-centric approach to business process modeling, which began at IBM Research in the late 1990's and was first presented in [101], proposes business artifacts (or simply *artifacts*) to model key business-relevant entities. Artifacts are equipped with an information model, representing the data maintained by the artifact, and they evolve over time following a so-called lifecycle. Processes organize atomic tasks or available services that are of interest into a possibly complex workflow.

The artifact-centric approach provides a simple and robust structure for business process development, which has been advocated as superior to the traditional activity-centric approach, especially when dealing with complex and large process models. While the traditional workflow approach does not lend itself to componentization in a natural way [94], the artifact-centric approach is claimed to enhance efficiency, especially when dealing with business process transformations to expand and/or streamline the process [43, 41, 97]. Fundamental notions from the artifact-centric approach have also been deployed in commercial products underlying IBM's commercial service offerings [115].

The surveys [85, 60] overviewed the research results on the artifact-centric approach to business process specification, management, deployment and analysis, tracing the roadmap of research directions and challenges. As far as verification is concerned, this triggered several lines of research aiming at decidable techniques for verification over processes and data, to be reassessed and extended towards the artifact-centric setting.

Seminal works on the analysis of artifact-centric systems is presented in [81, 42]. In [81], systems constituted by multiple interconnected artifacts are studied. The artifact information model contains the current state, and a tuple of attributes. Each attribute, in turn, may refer either to a primitive value or to some other artifact instance. Artifact lifecycles have a procedural flavor, based on finite state machines whose transitions either create a new artifact, or modify/eliminate an existing one. In this setting, first-order CTL with quantification across states is considered, showing decidability of verification for such formulas in the case of bounded domains, and in the case of unbounded domains, under the assumption that quantification only ranges over artifacts (and not values), and the number of artifacts is bounded. [42] tackles artifact systems that are similar, in spirit, to the ones of [81], but where lifecycles follow a more declarative style, based on business rules that activate

services. Services are in turn described in terms of preconditions and non-deterministic effects related to the creation, manipulation and elimination of artifacts. Manipulation of attributes focuses only on whether these attributes are defined or undefined (so that values are abstracted away). A set of pre-defined reasoning tasks (successful path completion, existence of dead-end paths, attribute redundancy) is tackled, showing that all are undecidable in the general case, but become decidable if no new artifacts can be created, or by imposing various restrictions, such as monotonicity of services (i.e., each attribute is written at most once).

In [70], the artifact model proposed in [42] is extended so as to include a static read-only database, and to handle a relational state in addition to attributes, whose values are not abstracted away and can be compared by service and property specifications according to a dense linear order. Runs can receive unbounded external input from the infinite domain of values. As verification formalism, a variant of first-order LTL is considered, where statements about individual artifact instances in the run may share variables that are outermost universally quantified. Decidability of verification is obtained by restricting such logic and the system specification to be *guarded*. The guarded restriction introduces a form of bounded quantification in the properties and formulas driving the system's evolution, which resembles input-boundedness [113, 114, 72]. In particular, read-only and read-write database relations are accessed differently, querying the latter only by checking whether they contain a given tuple of constants. It is shown that this restriction is tight, and that integrity constraints cannot be added to the framework, since even a single functional dependency leads to undecidability of verification. Decidability comes with a PSPACE upper bound for fixed-arity schemas, and EXPSPACE otherwise. [61, 62, 63] extend this approach by forbidding read-write relations, but this allows the extension of the decidability result to integrity constraints expressed as embedded dependencies with terminating chase, and to any decidable arithmetic.

Another line of research building on the artifact-centric paradigm is [9, 3, 10, 11], which study the specification and verification of artifact-centric systems that rely on an active XML-based information model. *Active XML* [2] (*AXML* for short) extends XML by allowing parts of the document to be specified in an intensional way, by means of embedded calls to internal functions or external services. In the artifact-centric setting, AXML documents support the design of complex workflows, providing at the same time a description of the underlying data and of the sub-tasks (formally, internal functions) to be orchestrated by the workflow. In particular, the boundaries of decidability for the verification of systems based on multiple, interacting AXML documents are delineated. Temporal properties of runs are specified in a tree pattern-based temporal logic, called Tree-LTL, which exploits tree-like patterns to query the states of the system, and combines them through linear-time temporal operators to predicate about the evolution of a system run. Similarly to the logics considered in Section 4.2, in Tree-LTL variables are existentially quantified within a state, or universally quantified by means of an outermost quantifier. The systems considered for verification rely on *guarded AXML* (*GAXML*) documents, which control the initiation and completion of sub-tasks by means of boolean combinations of tree-patterns. Decidability of verification is achieved by disallowing recursion in GAXML systems, which leads to bound the total number of sub-tasks invoked along a run. In this setting, the complexity of verification is shown to be co2-EXPTIME-complete.

In [4, 5], the problem of comparing different data-aware workflow specification frameworks based on AXML is tackled. It is argued that comparing workflow specification formalisms is intrinsically difficult because of the diversity of data models and control-flow mechanisms, and the lack of a standard yardstick for expressiveness. For example, AXML workflows could employ automata, pre-and-post conditions, or declarative temporal logic formulas to express the dynamics of the system. A unifying approach based on views is then proposed, where views are exploited to isolate the relevant aspects to be taken into account when comparing different specification frameworks. Notably, the approach is used to show that the different control mechanisms for Active XML workflows are largely equivalent, an indication of the robustness of the model.

4.4 Data-Centric Dynamic Systems

A recent line of research has been aiming at developing a framework for the combination of data and processes that is expressive and robust with respect to the system model, general in the verification formalism, and at the same time guarantees decidability of verification:

- Expressiveness guarantees the ability to capture a wide range of concrete systems (such as web applications and artifact-centric systems, which call for full create-read-update-delete, CRUD, operations over the database), and favours the adoption of the framework in real-world scenarios.
- Robustness makes the framework apt to adjustments (e.g., allowing for inclusion of different kinds of integrity constraints), which improves usability in the modeling phase.
- Generality of the verification formalism provides the ability to capture both linear-time and branching-time properties, at the same time supporting first-order quantification across states.

These apparently incompatible requirements can be accomplished together by imposing (automatically checkable) structural conditions on the process dynamics and on how they accounts for the evolution of the data over time. We refer to this kind of framework as *Data-Centric Dynamic Systems* (*DCDS*).

From a technical point of view, research on DCDSs has drawn inspiration from the works dealing with termination of rule-based systems, and in particular from acyclicity conditions in data exchange [92]. The key idea connecting data exchange with DCDSs is that the semantics of tasks progressing the data can be related to the firing of a (set of) tuple-generating dependencies (TGDs). Also, notice that all the works discussed below assume that verification is done with respect to a system with a given initial database.

The first works literally exploiting the connection between tasks and TGSs were [51, 66]. There the transition relation itself is described in terms of TGDs that map the current state, represented as a relational database instance, to the next one. Null values are used to model the incorporation of new, unknown data into the system. The process evolution is essentially a form of chase. Under suitable weak acyclicity conditions this chase terminates, guaranteeing in turn that the system is finite-state. Decidability is then shown for a first-order μ-calculus without first-order quantification across states. This approach was extended by [24], where TGDs are replaced by actions allowing negation in the preconditions. In this revised framework, values imported from the external environment are represented by uninterpreted function terms, which play the same role as nulls in the work by [51, 66]. Since both [24] and [51, 66] rely on a purely relational setting, this choice leads to an ad-hoc interpretation of equality, where each null value/function term is considered only equal to itself.

Differently from these works, [37] considered a first-order variant of CTL with no quantification across states as verification formalism. The framework supports the incorporation of new values from the external environment as parameters of the actions; the

corresponding execution semantics considers all the possible actual values, thus leading to an infinite-state transition systems. As for decidability of verification, it is shown that, under the assumption that each state of the system (constituted by the union of artifacts' relational instances) has a bounded active domain, it is possible to construct a *faithful* (i.e., sound and complete with respect to the verification logic) abstract transition system which, differently from the original one, has a finite number of states. [38] looks at quantification across in this setting. It relies on the semantic property of *genericity* [8] (called there "uniformity"), which guarantees that the transition system representing the execution of the process under study is not able to distinguish among states that have the same constants and the same patterns of data. Under the assumptions of genericity and state boundedness, decidability of verification is achieved for a richer logic, namely CTL with quantification across states, interpreted under the active domain semantics.

In [26], a comprehensive study of relational DCDSs is provided, where new information from the external world can be incorporated into the system through calls to deterministic and non-deterministic services. Verification for variants of first-order μ-calculus is investigated, where first-order quantification affects the objects across the possibly infinite states of the system. With such an expressive formalism, attention must be paid so as not to accumulate along a run an unbounded amount of information about which the formula to be verified may predicate. The accumulation of information may occur in the variables that are used as arguments of quantification, hence one needs to suitably control how these variables are quantified upon in the formula. In particular, two variants of first-order μ-calculus are singled out. The first one, called *history-preserving μ-calculus*, preserves knowledge of objects encountered along a run, by relying on an active domain semantics. In this case, decidability (in EXPTIME in the initial database) is obtained under the assumption that the data introduced along a run are bounded, though they may not be bounded in the overall system (*run-boundedness* condition). The second one, called *persistence-preserving μ-calculus*, preserves knowledge of an object only if the object is continuously present across successive states. In this case, decidability (again in EXPTIME in the initial database) is obtained even when infinitely many values are introduced along a run, as long as there is an overall bound on the number of objects accumulated in the same state (*state-boundedness* condition). Technically, decidability is shown in both cases by relying on finite-state abstractions that are faithful (i.e., sound and complete) according to suitable notions of bisimulations, tailored towards the two verification languages. The abstractions exploit an implicit form of genericity, which is enforced by the model underlying DCDSs and by the services they interact with. Even though run-boundedness and state-boundedness are semantic conditions that are undecidable to check, sufficient syntactic conditions have been singled out, respectively relying on the notion of *weak acyclicity* in data exchange [92], and on the novel notion of *generate-recall acyclicity*.

Considering the combination of the results in [38] and [26], we can observe that, for state-bounded systems, in the presence of first-order quantification over the active domain of the current state, without limiting the scope of quantification to the objects that persist across states, verification is decidable in case of CTL (i.e., with alternation-fee fixpoints), but becomes undecidable for LTL, and hence for the μ-calculus in general. The proof of undecidability in [26] is based on a reduction to LTL with freeze quantifiers [68].

5. CONCLUSIONS AND OUTLOOK

In this work, we surveyed the research on foundations of data-aware (business) processes that has been carried out in the database theory community. This community has developed rich techniques to deal with data and processes and among the various areas of computer science it is probably the one in the best position to lay the foundations of data-aware process analysis.

Several challenges are ahead of us. In particular, the work done in the last years on verification of data-aware processes shows that the analysis techniques proposed are exponential in those data that "change". So circumscribing what can be changed by a process appears to be a key issue to make verification practical. This is particularly relevant in the context of processes acting on web data like those that are the focus of [12, 84].

Notably, recent significant works on the analysis of data-aware processes, such as those in [38, 26], rely on a mix of contributions, formalisms, and techniques that have their roots in several areas of computer science, in particular database theory, formal methods, logic, process management, and knowledge representation. In particular, with respect to the latter, it is worth noting that a field where data and processes have always been considered together is that of *reasoning about actions* in Artificial Intelligence. Since the introduction of Situation Calculus in [99, 106], a first-order setting was considered for describing the states that actions could modify. An additional difficulty is the presence of incomplete information, typical of knowledge bases. Verification of processes is obviously of interest, but most decidability results are based on bounding the domain, thus reducing to a propositional case. The techniques discussed here do impact that literature as well and reflect into conditions for decidability of verification (see, e.g., [67, 25]).

We also notice that the majority of contemporary techniques for data-aware process analysis rely on the construction of faithful, finite-state abstractions starting from the infinite-state transition system representing the original model. In principle, this enables the exploitation of conventional model checkers to actually address the analysis problem, which is a promising approach. At the same time, we stress the importance of further pursuing the cross-fertilization with other research such as that of *data words* [44, 45] that have developed techniques to deal with systems whose state space is in general infinite due to the presence of data.

Acknowledgements

We would like to thank Babak Bagheri Hariri, Riccardo De Masellis, Alin Deutsch, Paolo Felli, Rick Hull, Maurizio Lenzerini, Alessio Lomuscio, Fabio Patrizi, Jianwen Su, and Moshe Vardi for stimulating ideas, inspiring discussions, and novel insights on the topics of this paper. This research has been partially supported by the EU under the ICT Collaborative Project ACSI (Artifact-Centric Service Interoperation), grant agreement n. FP7-257593. Diego Calvanese acknowledges the kind support of the Wolfgang Pauli Institute Vienna and of the Technical University Vienna for his sabbatical stay.

6. REFERENCES

[1] M. Abadi and Z. Manna. Temporal logic programming. *J. of Symbolic Computation*, 8(3):277–295, 1989.

[2] S. Abiteboul, O. Benjelloun, and T. Milo. Positive active XML. In *Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2004)*, pages 35–45, 2004.

[3] S. Abiteboul, P. Bourhis, A. Galland, and B. Marinoiu. The AXML artifact model. In *Proc. of the 16th Int. Symp. on Temporal Representation and Reasoning (TIME 2009)*, pages 11–17, 2009.

[4] S. Abiteboul, P. Bourhis, and V. Vianu. Comparing workflow specification languages: a matter of views. In

Proc. of the 14th Int. Conf. on Database Theory (ICDT 2011), pages 78–89, 2011.

[5] S. Abiteboul, P. Bourhis, and V. Vianu. Comparing workflow specification languages: A matter of views. *ACM Trans. on Database Systems*, 37(2):10:1–10:59, 2012.

[6] S. Abiteboul, L. Herr, and J. Van den Bussche. Temporal versus first-order logic to query temporal databases. In *Proc. of the 15th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'96)*, pages 49–57, 1996.

[7] S. Abiteboul and R. Hull. IFO: A formal semantic database model. *ACM Trans. on Database Systems*, 12(4):297–314, 1987.

[8] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.

[9] S. Abiteboul, L. Segoufin, and V. Vianu. Static analysis of Active XML systems. In *Proc. of the 27th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2008)*, pages 221–230, 2008.

[10] S. Abiteboul, L. Segoufin, and V. Vianu. Modeling and verifying Active XML artifacts. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 32(3):10–15, 2009.

[11] S. Abiteboul, L. Segoufin, and V. Vianu. Static analysis of Active XML systems. *ACM Trans. on Database Systems*, 34(4):23:1–23:44, 2009.

[12] S. Abiteboul, P. Senellart, and V. Vianu. The ERC webdam on foundations of web data management. In *Proc. of the 21st Int. World Wide Web Conf. (WWW 2012)*, pages 211–214, 2012.

[13] S. Abiteboul and V. Vianu. Transactions in relational databases (preliminary report). In *Proc. of the 10th Int. Conf. on Very Large Data Bases (VLDB'84)*, pages 46–56, 1984.

[14] S. Abiteboul and V. Vianu. Transactions and integrity constraints. In *Proc. of the 4th ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS'85)*, pages 193–204, 1985.

[15] S. Abiteboul and V. Vianu. Deciding properties of transactional schemas. In *Proc. of the 5th ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS'86)*, pages 235–239, 1986.

[16] S. Abiteboul and V. Vianu. A transcation language complete for database update and specification. In *Proc. of the 6th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'87)*, pages 260–268, 1987.

[17] S. Abiteboul and V. Vianu. Equivalence and optimization of relational transactions. *J. of the ACM*, 35(1):70–120, 1988.

[18] S. Abiteboul and V. Vianu. Procedural and declarative database update languages. In *Proc. of the 7th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'88)*, pages 240–250, 1988.

[19] S. Abiteboul, V. Vianu, B. Fordham, and Y. Yesha. Relational transducers for electronic commerce. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 179–187, 1998.

[20] S. Abiteboul, V. Vianu, B. Fordham, and Y. Yesha. Relational transducers for electronic commerce. *J. of Computer and System Sciences*, 61(2):236–269, 2000.

[21] A. Aiken, J. Widom, and J. M. Hellerstein. Behavior of database production rules: Termination, confluence, and observable determinism. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 59–68, 1992.

[22] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[23] A. Awad, G. Decker, and M. Weske. Efficient compliance checking using BPMN-Q and temporal logic. In *Proc. of the 6th Int. Conference on Business Process Management (BPM 2008)*, volume 5240 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2008.

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

[25] B. Bagheri Hariri, D. Calvanese, M. Montali, G. De Giacomo, R. De Masellis, and P. Felli. Description logic Knowledge and Action Bases. *J. of Artificial Intelligence Research*, 46, 2013.

[26] B. Bagheri Hariri, D. Calvanese, M. Montali, G. De Giacomo, and A. Deutsch. Verification of relational data-centric dynamic systems with external services. In *Proc. of the 32nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2013)*, 2013.

[27] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[28] J. Bailey, L. Crnogorac, K. Ramamohanarao, and H. Søndergaard. Abstract interpretation of active rules and its use in termination analysis. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 188–202. Springer, 1997.

[29] J. Bailey, G. Dong, and K. Ramamohanarao. Structural issues in active rule systems. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 203–214. Springer, 1997.

[30] J. Bailey, G. Dong, and K. Ramamohanarao. Decidability and undecidability results for the termination problem of active database rules. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 264–273, 1998.

[31] Z. Bao, S. B. Davidson, and T. Milo. Labeling workflow views with fine-grained dependencies. *Proc. of the VLDB Endowment*, 5(11):1208–1219, 2012.

[32] M. Baudinet. Temporal logic programming is complete and expressive. In *Proc. of the 16th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL'89)*, pages 267–280, 1989.

[33] M. Baudinet. On the expressiveness of temporal logic programming. *Information and Computation*, 117(2):157–180, 1995.

[34] M. Baudinet, M. Niézette, and P. Wolper. On the representation of infinite temporal data and queries. In *Proc. of the 10th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'91)*, pages 280–290, 1991.

[35] C. Beeri, A. Eyal, S. Kamenkovich, and T. Milo. Querying business processes. In *Proc. of the 32nd Int. Conf. on Very Large Data Bases (VLDB 2006)*, pages 343–354, 2006.

[36] C. Beeri, A. Eyal, S. Kamenkovich, and T. Milo. Querying business processes with BP-QL. *Information Systems*, 33(6):477–507, 2008.

[37] F. Belardinelli, A. Lomuscio, and F. Patrizi. Verification of deployed artifact systems via data abstraction. In *Proc. of the 9th Int. Joint Conf. on Service Oriented Computing (ICSOC 2011)*, volume 7084 of *Lecture Notes in Computer Science*, pages 142–156. Springer, 2011.

[38] F. Belardinelli, A. Lomuscio, and F. Patrizi. An abstraction technique for the verification of artifact-centric systems. In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012)*, 2012.

[39] M. Benedikt, T. Griffin, and L. Libkin. Verifiable properties of database transactions. In *Proc. of the 15th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'96)*, pages 117–127, 1996.

[40] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, and M. Mecella. Automatic composition of transition-based semantic web services with messaging. In *Proc. of the 31st Int. Conf. on Very Large Data Bases (VLDB 2005)*, pages 613–624, 2005.

[41] K. Bhattacharya, N. S. Caswell, S. Kumaran, A. Nigam, and F. Y. Wu. Artifact-centered operational modeling: Lessons from customer engagements. *IBM Systems Journal*, 46(4):703–721, 2007.

[42] K. Bhattacharya, C. E. Gerede, R. Hull, R. Liu, and J. Su. Towards formal analysis of artifact-centric business process models. In *Proc. of the 5th Int. Conference on Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*, pages 288–234. Springer, 2007.

[43] K. Bhattacharya, R. Guttman, K. Lyman, F. F. Heath, S. Kumaran, P. Nandi, F. Y. Wu, P. Athma, C. Freiberg, L. Johannsen, and A. Staudt. A model-driven approach to industrializing discovery processes in pharmaceutical research. *IBM Systems Journal*, 44(1):145–162, 2005.

[44] M. Bojanczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Trans. on Computational Logic*, 12(4):27, 2011.

[45] M. Bojanczyk and T. Place. Toward model theory with data values. In *Proc. of the 39th Int. Coll. on Automata, Languages and Programming (ICALP 2012)*, pages 116–127, 2012.

[46] A. J. Bonner. Workflow, transactions, and datalog. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'99)*, pages 294–305, 1999.

[47] A. Bouajjani, P. Habermehl, Y. Jurski, and M. Sighireanu. Rewriting systems with data. In *Proc. of the 16th Int. Symp. on Fundamentals of Computation Theory (FCT 2007)*, 2007.

[48] A. Bouajjani, P. Habermehl, and R. Mayr. Automatic verification of recursive procedures with one integer parameter. *Theoretical Computer Science*, 295, 2003.

[49] Y. Breitbart, A. Deacon, H.-J. Schek, A. P. Sheth, and G. Weikum. Merging application-centric and data-centric approaches to support transaction-oriented multi-system workflows. *SIGMOD Record*, 22(3):23–30, 1993.

[50] O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification of infinite structures. In *Handbook of Process Algebra*. Elsevier, 2001.

[51] P. Cangialosi, G. De Giacomo, R. De Masellis, and R. Rosati. Conjunctive artifact-centric services. In *Proc. of the 8th Int. Joint Conf. on Service Oriented Computing (ICSOC 2010)*, volume 6470 of *Lecture Notes in Computer Science*, pages 318–333. Springer, 2010.

[52] A. K. Chandra and D. Harel. Computable queries for relational data bases. *J. of Computer and System Sciences*, 21(2):156–178, 1980.

[53] P. P. Chen. The Entity-Relationship model: Toward a unified view of data. *ACM Trans. on Database Systems*, 1(1):9–36, Mar. 1976.

[54] J. Chomicki. Temporal inegrity constraints in relational databases. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 17(2):33–37, 1994.

[55] J. Chomicki. Temporal query languages: a survey. In *Proc. of the 1st Int. Conf. on Temporal Logic (ICTL'94)*, volume 827 of *Lecture Notes in Computer Science*, pages 506–534. Springer, 1994.

[56] J. Chomicki. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Trans. on Database Systems*, 20(2):149–186, 1995.

[57] J. Chomicki and T. Imielinski. Temporal deductive databases and infinite objects. In *Proc. of the 7th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'88)*, pages 61–73, 1988.

[58] J. Chomicki and D. Niwinski. On the feasibility of checking temporal integrity constraints. In *Proc. of the 12th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'93)*, pages 202–213, 1993.

[59] J. Chomicki and D. Niwinski. On the feasibility of checking temporal integrity constraints. *J. of Computer and System Sciences*, 51(3):523–535, 1995.

[60] D. Cohn and R. Hull. Business artifacts: A data-centric approach to modeling business operations and processes. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 32(3):3–9, 2009.

[61] E. Damaggio, A. Deutsch, R. Hull, and V. Vianu. Automatic verification of data-centric business processes. In *Proc. of the 9th Int. Conference on Business Process Management (BPM 2011)*, volume 6896 of *Lecture Notes in Computer Science*, pages 30–16. Springer, 2011.

[62] E. Damaggio, A. Deutsch, and V. Vianu. Artifact systems with data dependencies and arithmetic. In *Proc. of the 14th Int. Conf. on Database Theory (ICDT 2011)*, pages 66–77, 2011.

[63] E. Damaggio, A. Deutsch, and V. Vianu. Artifact systems with data dependencies and arithmetic. *ACM Trans. on Database Systems*, 37(3):22, 2012.

[64] S. B. Davidson, T. Milo, and S. Roy. A propagation model for provenance views of public/private workflows. In *Proc. of the 16th Int. Conf. on Database Theory (ICDT 2013)*, pages 165–176, 2013.

[65] H. Davulcu, M. Kifer, C. R. Ramakrishnan, and I. V. Ramakrishnan. Logic based modeling and analysis of workflows. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 25–33, 1998.

[66] G. De Giacomo, R. De Masellis, and R. Rosati. Verification of conjunctive artifact-centric services. *Int. J. of Cooperative Information Systems*, 21(2):111–139, 2012.

[67] G. De Giacomo, Y. Lesperance, and F. Patrizi. Bounded situation calculus action theories and decidable verification. In *Proc. of the 13th Int. Conf. on the Principles of*

Knowledge Representation and Reasoning (KR 2012), pages 467–477, 2012.

[68] S. Demri and R. Lazić. LTL with the Freeze quantifier and register automata. In *Proc. of the 21st IEEE Symp. on Logic in Computer Science (LICS 2006)*, pages 17–26, 1996.

[69] D. Deutch and T. Milo. A quest for beauty and wealth (or, business processes for database researchers). In *Proc. of the 30th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2011)*, pages 1–12, 2011.

[70] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic verification of data-centric business processes. In *Proc. of the 12th Int. Conf. on Database Theory (ICDT 2009)*, pages 252–267, 2009.

[71] A. Deutsch, M. Marcus, L. Sui, V. Vianu, and D. Zhou. A verifier for interactive, data-driven web applications. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 539–550, 2005.

[72] A. Deutsch, L. Sui, and V. Vianu. Specification and verification of data-driven web services. In *Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2004)*, pages 71–82, 2004.

[73] A. Deutsch, L. Sui, and V. Vianu. Specification and verification of data-driven web applications. *J. of Computer and System Sciences*, 73(3):442–474, 2007.

[74] A. Deutsch, L. Sui, V. Vianu, and D. Zhou. A system for specification and verification of interactive, data-driven web applications. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 772–774, 2006.

[75] A. Deutsch, L. Sui, V. Vianu, and D. Zhou. Verification of communicating data-driven web services. In *Proc. of the 25th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2006)*, pages 90–99, 2006.

[76] J. Esparza. Decidability of model checking for infinite-state concurrent systems. *Acta Informatica*, 34(2):85–107, 1997.

[77] J. Esparza. Decidability and complexity of Petri net problems – An introduction. In *Lectures on Petri Nets I*, Lecture Notes in Computer Science, pages 374–428. Springer, 1998.

[78] X. Fu, T. Bultan, R. Hull, and J. Su. Verification of Vortex workflows. In *Proc. of the 7th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2001)*, volume 2031 of *Lecture Notes in Computer Science*, pages 143–157. Springer, 2001.

[79] D. M. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 28 of *Oxford Logic Guides*. Oxford University Press, 1994.

[80] S. K. Gadia. Weak temporal relations. In *Proc. of the 5th ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS'86)*, pages 70–77, 1986.

[81] C. E. Gerede and J. Su. Specification and verification of artifact behaviors in business process models. In *Proc. of the 5th Int. Conf. on Service Oriented Computing (ICSOC 2007)*, volume 4749 of *Lecture Notes in Computer Science*, pages 181–192. Springer, 2007.

[82] M. H. Graham, N. D. Griffeth, and B. Smith-Thomas. Reliable scheduling of database transactions for unreliable systems. In *Proc. of the 3rd ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS'84)*, pages 300–310, 1984.

[83] D. Harel. Recurring dominoes: Making the highly undecidable highly understandable. *Ann. of Discrete Mathematics*, 24:51–72, 1985.

[84] J. M. Hellerstein. The declarative imperative: Experiences and conjectures in distributed logics. *SIGMOD Record*, 39(1):5–19, 2010.

[85] R. Hull. Artifact-centric business process models: Brief survey of research results and challenges. In *Proc. of the On the Move Confederated Int. Conf. (OTM 2008)*, volume 5332 of *Lecture Notes in Computer Science*, pages 1152–1163. Springer, 2008.

[86] R. Hull, M. Benedikt, V. Christophides, and J. Su. E-services: a look behind the curtain. In *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2003)*, pages 1–14, 2003.

[87] R. Hull and J. Su. The Vortex approach to integration and coordination of workflows. In *Proc. of the Workshop on Cross-Organisational Workflow Management and Co-ordination*, volume 17 of *CEUR Electronic Workshop Proceedings,* `http://ceur-ws.org/`, 1999.

[88] C. S. Jensen and R. T. Snodgrass. Temporal data management. *IEEE Trans. on Knowledge and Data Engineering*, 11(1):36–44, 1999.

[89] F. Kabanza, J.-M. Stévenne, and P. Wolper. Handling infinite temporal data. In *Proc. of the 9th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'90)*, pages 392–403, 1990.

[90] R. Karel, C. Richardson, and C. Moore. Warning: Don't assume your business processes use master data – Synchronize your business process and master data strategies. Report, Forrester, Sept. 2009.

[91] Z. M. Kedem and A. Tuzhilin. Relational database behavior: Utilizing relational discrete event systems and models. In *Proc. of the 8th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'89)*, pages 336–346, 1989.

[92] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *Proc. of the 24th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2005)*, pages 61–75, 2005.

[93] R. Krishnamurthy and U. Dayal. Theory of serializability for a parallel model of transactions. In *Proc. of the 1st ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS'82)*, pages 293–305, 1982.

[94] S. Kumaran, R. Liu, and F. Y. Wu. On the duality of information-centric and activity-centric models of business processes. In *Proc. of the 20th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2008)*, pages 32–47, 2008.

[95] R. Lazic, T. Newcomb, J. Ouaknine, A. W. Roscoe, and J. Worrell. Nets with tokens which carry data. *Fundamenta Informaticae*, 88(3):251–274, 2008.

[96] U. W. Lipeck. Transformation of dynamic integrity constraints into transaction specifications. In *Proc. of the Int. Conf. on Database Theory (ICDT'88)*, volume 326 of *Lecture Notes in Computer Science*, pages 322–337. Springer, 1988.

[97] R. Liu, K. Bhattacharya, and F. Y. Wu. Modeling business contexture and behavior using business artifacts. In *Proc. of the 19th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2007)*, volume 4495 of *Lecture Notes in Computer Science*, pages 324–339. Springer, 2007.

[98] N. A. Lynch. Concurrency control for resilient nested transactions. In *Proc. of the 2nd ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS'83)*, pages 166–181, 1983.

[99] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of aritificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[100] A. Meyer, S. Smirnov, and M. Weske. Data in business processes. Technical Report 50, Hasso-Plattner-Institut for IT Systems Engineering, Universität Potsdam, 2011. Available online at `http://opus.kobv.de/ubp/volltexte/2011/5304/`.

[101] A. Nigam and N. S. Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3):428–445, 2003.

[102] C. Ouyang, M. Dumas, A. H. M. ter Hofstede, and W. M. P. van der Aalst. From BPMN process models to BPEL web services. In *Proc. of the 4th IEEE Int. Conf. on Web Services (ICWS 2006)*, pages 285–292. IEEE Computer Society Press, 2006.

[103] P. Picouet and V. Vianu. Semantics and expressiveness issues in active databases. In *Proc. of the 14th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'95)*, pages 126–138, 1995.

[104] P. Picouet and V. Vianu. Expressiveness and complexity of active databases. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 155–172. Springer, 1997.

[105] M. Reichert. Process and data: Two sides of the same coin? In *Proc. of the On the Move Confederated Int. Conf. (OTM 2012)*, volume 7565 of *Lecture Notes in Computer Science*, pages 2–19. Springer, 2012.

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

[107] A. Reuter, S. Ceri, J. Gray, B. Salzberg, and G. Weikum. Databases and workflow management: What is it all about? (panel). In *Proc. of the 21st Int. Conf. on Very Large Data Bases (VLDB'95)*, page 632, 1995.

[108] C. Richardson. Warning: Don't assume your business processes use master data. In *Proc. of the 8th Int. Conference on Business Process Management (BPM 2010)*, volume 6336 of *Lecture Notes in Computer Science*, pages 11–12. Springer, 2010.

[109] K. Schmidt. LoLA: A low level analyser. In *Proc. of the 21st Int. Conf. on Application and Theory of Petri Nets (ICATPN 2000)*, Lecture Notes in Computer Science, pages 465–474. Springer, 2000.

[110] K. Schmidt. Using Petri net invariants in state space construction. In *Proc. of the 9th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2003)*, volume 2619 of *Lecture Notes in Computer Science*, pages 473–488. Springer, 2003.

[111] R. T. Snodgrass. The temporal query language TQuel. In *Proc. of the 3rd ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS'84)*, pages 204–213, 1984.

[112] E. Soisalon-Soininen and D. Wood. An optimal algorithm for testing for safety and detecting deadlocks in locked transaction systems. In *Proc. of the 1st ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS'82)*, pages 108–116, 1982.

[113] M. Spielmann. Verification of relational transducers for electronic commerce. In *Proc. of the 19th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2000)*, pages 92–103, 2000.

[114] M. Spielmann. Verification of relational transducers for electronic commerce. *J. of Computer and System Sciences*, 66(1):40–65, 2003.

[115] J. K. Strosnider, P. Nandi, S. Kumaran, S. P. Ghosh, and A. Arsanjani. Model-driven synthesis of SOA solutions. *IBM Systems Journal*, 47(3):415–432, 2008.

[116] J. Su. Letter from the editor of the special issue on management of data-centric business workïňĆows. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 32(3):2, Sept. 2009.

[117] J. van Benthem. Temporal logic. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 4*, pages 241–350. Oxford Scientific Publishers, 1996.

[118] W. M. P. van der Aalst. A decade of business process management conferences: Personal reflections on a developing discipline. In *Proc. of the 10th Int. Conference on Business Process Management (BPM 2012)*, volume 7481 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012.

[119] W. M. P. van der Aalst and C. Stahl. *Modeling Business Processes: a Petri Net-Oriented Approach*. The MIT Press, 2011.

[120] W. M. P. van der Aalst, M. Weske, and D. Grünbauer. Case handling: A new paradigm for business process support. *Data and Knowledge Engineering*, 53(2):129–162, 2005.

[121] M. Y. Vardi. A temporal fixpoint calculus. In *Proc. of the 15th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL'88)*, pages 250–259, 1988.

[122] M. Y. Vardi. Model checking for database theoreticians. In *Proc. of the 10th Int. Conf. on Database Theory (ICDT 2005)*, volume 3363 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.

[123] V. Vianu. Dynamic constraints and database evolution. In *Proc. of the 2nd ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS'83)*, pages 389–399, 1983.

[124] V. Vianu. *Dynamic Constraints and Database Evolution*. PhD thesis, University of Southern California, 1983.

[125] V. Vianu. Object projection views in the dynamic relational model. In *Proc. of the 3rd ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS'84)*, pages 214–220, 1984.

[126] V. Vianu. Automatic verification of database-driven systems: a new frontier. In *Proc. of the 12th Int. Conf. on Database Theory (ICDT 2009)*, pages 1–13, 2009.

[127] T. Yoneda, A. Shibayama, B.-H. Schlingloff, and E. M. Clarke. Efficient verification of parallel real-time systems. In *Proc. of the 5th Int. Conf. on Computer Aided Verification (CAV'93)*, volume 697 of *Lecture Notes in Computer Science*, pages 321–346. Springer, 1993.