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

Diego Calvanese

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy

Currently on sabbatical leave at Technical University Vienna, Austria

32nd ACM Symposium on Principles of Database Systems (PODS 2013)
23-26/6/2013 – New York, U.S.A.

Outline

"Sometimes I just feel like processing some data, but I have no data to process—other times I have the data, but I have nothing to process it with."

it

Outline

Processes and data

The information assets of an organization are constituted by:

- **data**, and
- **processes**, that determine how data changes and evolves over time.

Survey by Forrester investigates which of the two aspects should be given priority from the point of view of IT management [Karel *et al.*, 2009]:

- Business process management professionals: view data as subsidiary to processes manipulating them, and neglect importance of data quality.
- Data management experts: consider data as the driver of the organizational processes and are concerned about data quality only.

There is still a dichotomy

Dichotomy in the relative perception of importance has a negative impact:

- Little collaboration between the teams
  - running the master data management initiatives, and
  - managing the business processes.

  Forrester: 83% ... no interaction at all
- Little attention on the side of tool vendors to address the combined requirements:
  - Data management tools consider only the processes directly affecting the data in the tools, but not the actual business processes using the data,
  - Business process modeling suites do not allow for direct connection of data.

However, data and processes are tightly coupled together!

Two sides of the same coin

The need for overcoming this dichotomy is recognized by the (business) process modeling community as well [Reichert, 2012]:

"Process and data are just two sides of the same coin"

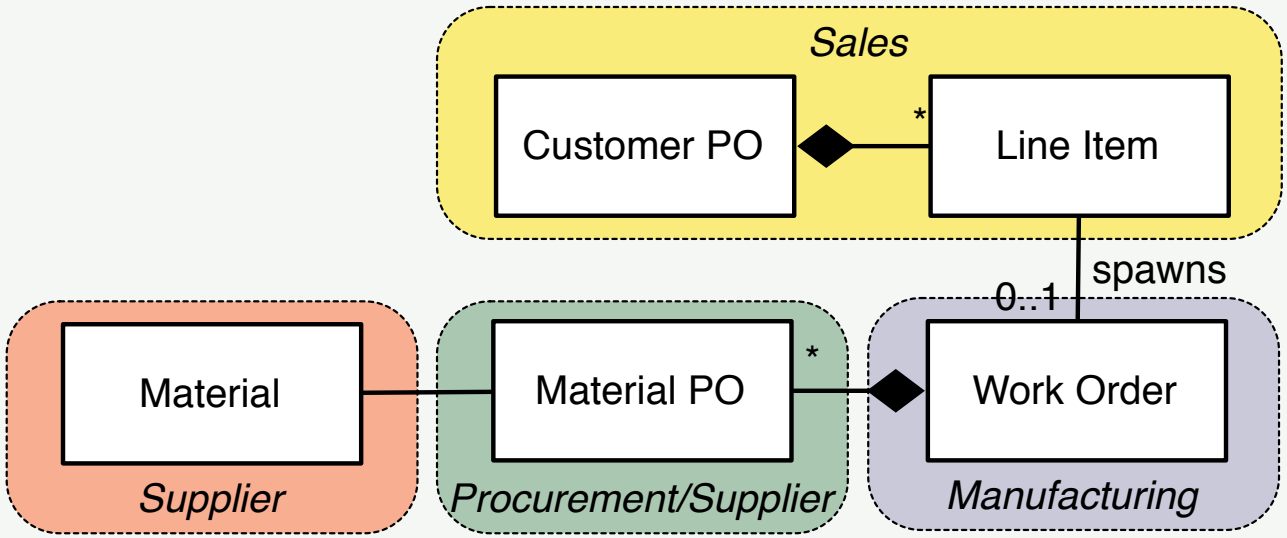
Two key areas where explicit representation of data in business processes is important [Meyer *et al.*, 2011]:

1. Modeling the core assets of an organization.
   - Data is crucial for the execution of business processes that create value.
   - Hence the business processes need to access the data, and this should be accounted for explicitly.
2. Business process controlling.
   - Key performance indicators and business goals are defined in terms of data.
   - To evaluate and control them, the data objects relevant for the activities contributing to the goals need to be identified.

Conventional data modeling

- Focus is on entities, relations, and static constraints that are relevant for the domain of interest.
- Result: **possibly distributed database**

Example: build-to-order (UML class diagram)

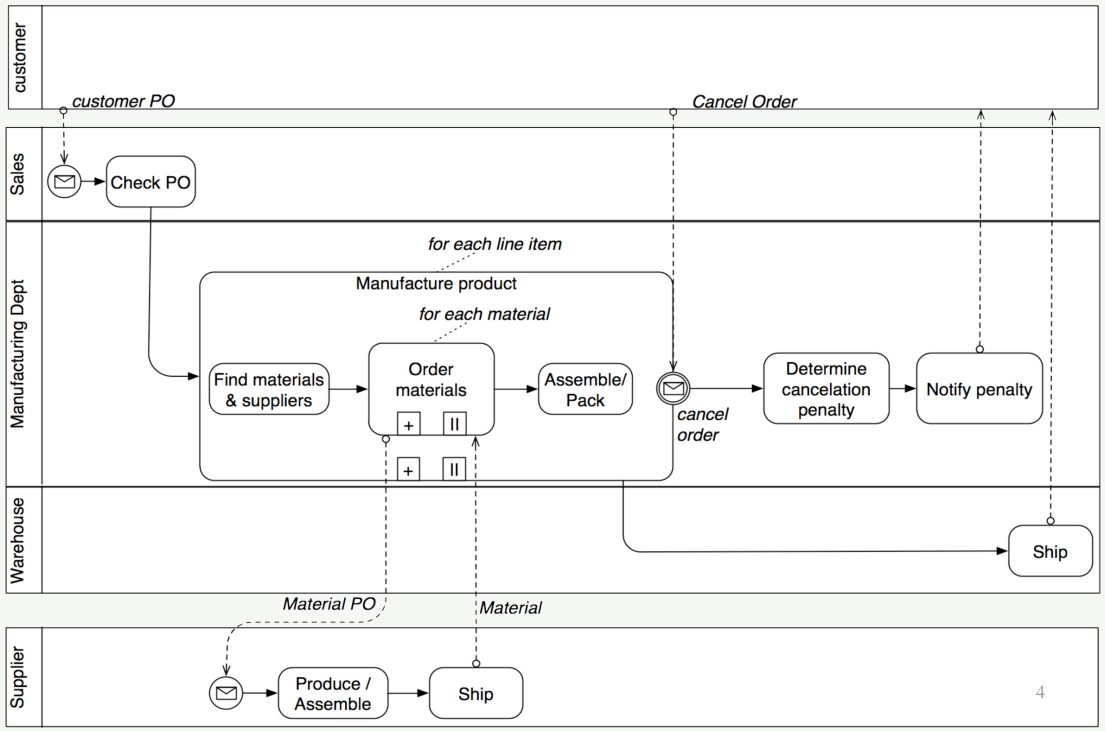


But how do data evolve?
Where can we see the "state" of an order?

Conventional process modeling

- Focus is on control flow of activities that realize the business objectives.
- Result: **executable process model**

Example: build-to-order (BPMN diagram)

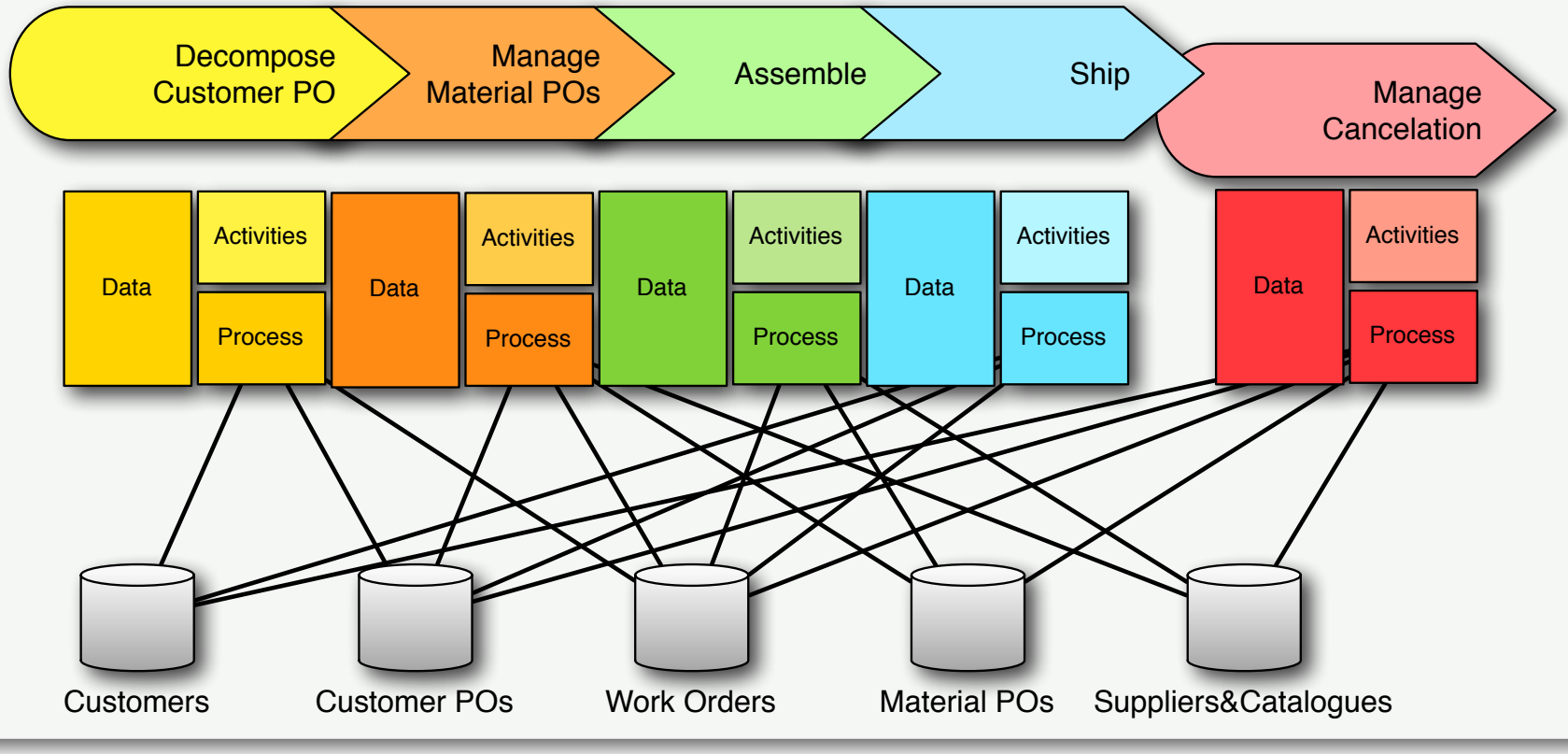


But how are data manipulated?
What impact does the cancellation of an order have?

Problem: interconnection between data and processes

- Conventional modeling: separation between data and processes.
- Connection is handled at the technological level.
  - ⇝ Lack of a conceptual view of data+processs that is holistic and coherent.
  - ⇝ It becomes difficult to manage and maintain the system.

Example: build-to-order

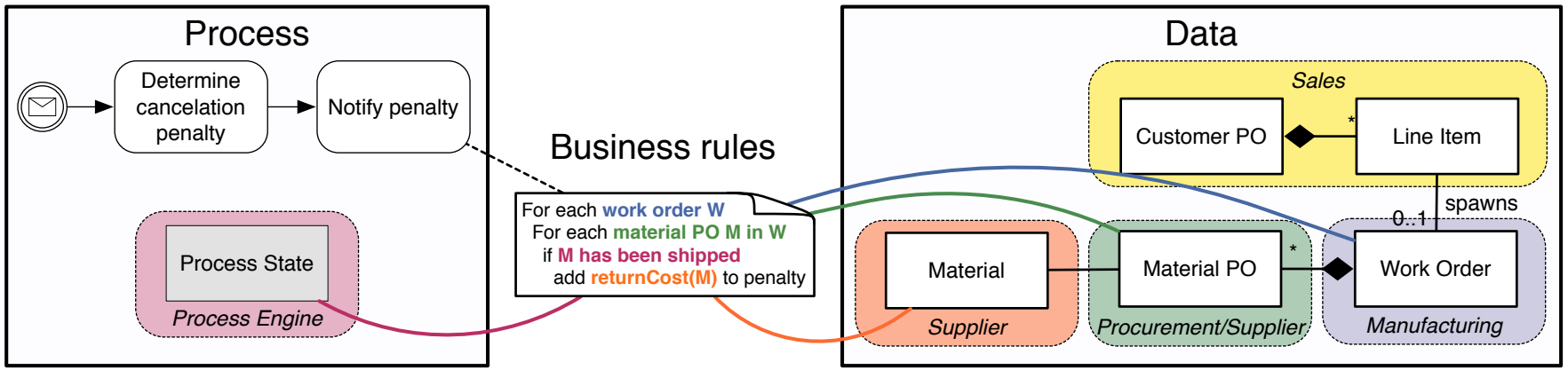


Problem: reconstructing the missing pieces

Lack of a holistic view of data+processs makes it difficult to:

- Reconstruct and aggregate the relevant information:
  - part of the data is in the DBs,
  - part of the data is hidden in the process execution engine.
- Decide how and where to model the domain relevant business rules:
  - At the DB level? Which DB? How to import the process data?
  - (Also) in the business model? How to import data from the DBs?

Example: build-to-order



Overcoming the dichotomy

Strong need for:

- suitable modeling formalisms supporting the integrated management of processes and data;
- methodologies for the design of systems based on such formalisms;
- systems and tools that implement these languages and methodologies.

This, in turn requires a foundational approach to:

- provide a clear understanding of (data-aware) process models w.r.t.
  - semantic properties, and
  - computational properties;
- enable **analysis** of such formalisms.

Analysis

Data-related analysis is well-established in DB theory:

- intensional reasoning over queries: containment, equivalence
- database dependencies: axiomatization, satisfaction, implication, ...
- semantic and conceptual data models
- reasoning over views
- ...

Analysis of system dynamics is also of great interest: **verification** of software and hardware systems **via model checking**

[2007 Turing Award: Clarke, Emerson, Sifakis]

- Dynamic properties of interest are formulated in a temporal logic (LTL, CTL, μ-calculus, ...).
- The transition system mathematically capturing the dynamics of the system of interest is (implicitly or explicitly) represented.
- The temporal logic formulas are checked (i.e., evaluated) over the transition system.

Model checking technology requires the transition system to be finite.

Analysis of data-aware processes

The presence of data complicates analysis significantly:

- **states** must be **modeled relationally** rather than propositionally;
- the resulting transition system is typically **infinite state**;
- query languages for analysis need to 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.

$\rightsquigarrow$ We need **first-order variants of temporal logics**

Model checking data-aware processes becomes immediately undecidable!

How can we mediate between:

- the expressiveness of the temporal property language, and
- the identification of classes of data-aware processes,

such that

1. we are to capture notable, real-world scenarios, but
2. analysis stays decidable, and possibly efficient.

Outline

Business process analysis

In BPM, **process model analysis** is considered the second most influential topic in the last decade (after process modeling languages) [van der Aalst, 2012].

However:

- Data has been abstracted away.
- Emphasis has been on the control-flow dimension:
  $\rightsquigarrow$ sophisticated techniques for absence of deadlocks, boundedness, soundness, or domain-dependent properties expressed in LTL or CTL.

Basic assumption: control-flow is captured by a (possibly infinite-state) propositional labeled transition system,

- labels represent the process tasks/activities
- concurrency is represented by interleaving
- transition system usually not represented explicitly, but is implicitly "folded" into a Petri net

Analysis of Petri nets

Verification of Petri nets:

- undecidable in general [Esparza, 1997; 1998]
- decidable for safe/bounded nets (transition graph is finite-state)

No satisfactory solution for specification and analysis of data-aware processes:

- colored Petri nets not suited to represent a DB
  - data are variables associated to tokens;
  - data are manipulated by procedural attachments to the transition in the net $\rightsquigarrow$ cannot be analyzed;
- BPMN (OMG standard) and BPEL (OASIS standard) suffer from similar problems:
  - they leave connection between data and process unspecified (e.g., do not capture atomic task behaviour)
  - hence, require to attach a program to every BPMN atomic task or BPEL service

Business artifacts to the rescue

- In early 2000, the **artifact-centric approach** emerged as a foundational proposal for merging data and processes together.
  - The emphasis is on data, which are modeled taking into account that they will be manipulated by processes.
  - Processes are modeled by considering how they manipulate data.
- Initial proposals by IBM [Nigam and Caswell, 2003], followed by [Bhattacharya *et al.*, 2007b; Deutsch *et al.*, 2007], ...
- See also EU project ACSI (for **Artifact-Centric Service Interoperation**), 2010–2013.

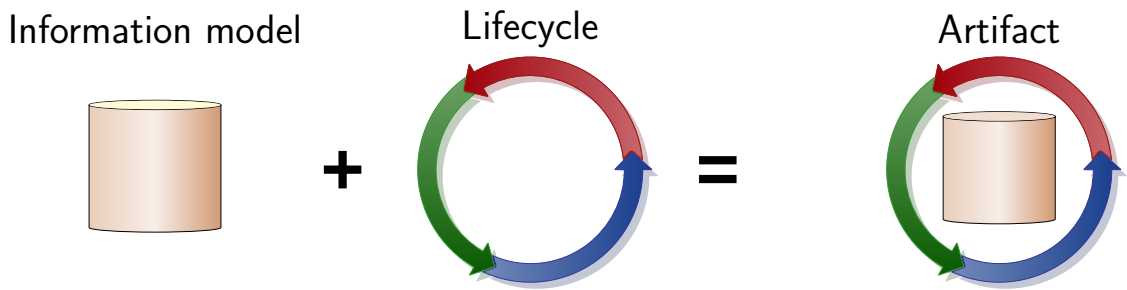
What is an artifact?

Definition

A key, business-relevant conceptual dynamic entity that is used in guiding the operation of a business.

Consists of:

- **Information model** - relevant data maintained by the artifact
- **Lifecycle model** - (implicit) description of the allowed information model evolutions through the execution of a process.

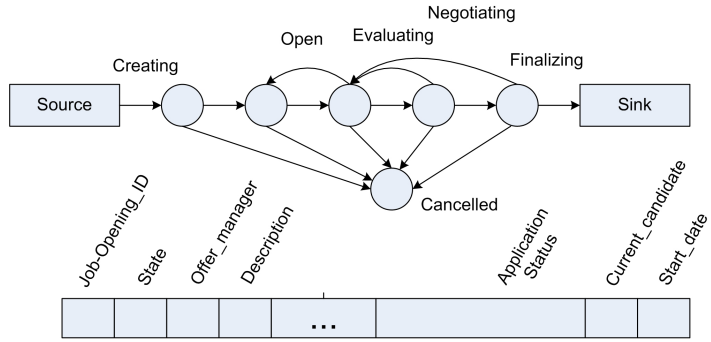


Goal: unified, end-to-end view of relevant entities and their possible evolutions.

Concrete models for artifacts

Key questions:

- How and where to store data maintained by their information models?
- How to specify the lifecycle of such artifacts?
- At which level of abstraction?



Some concrete information models:

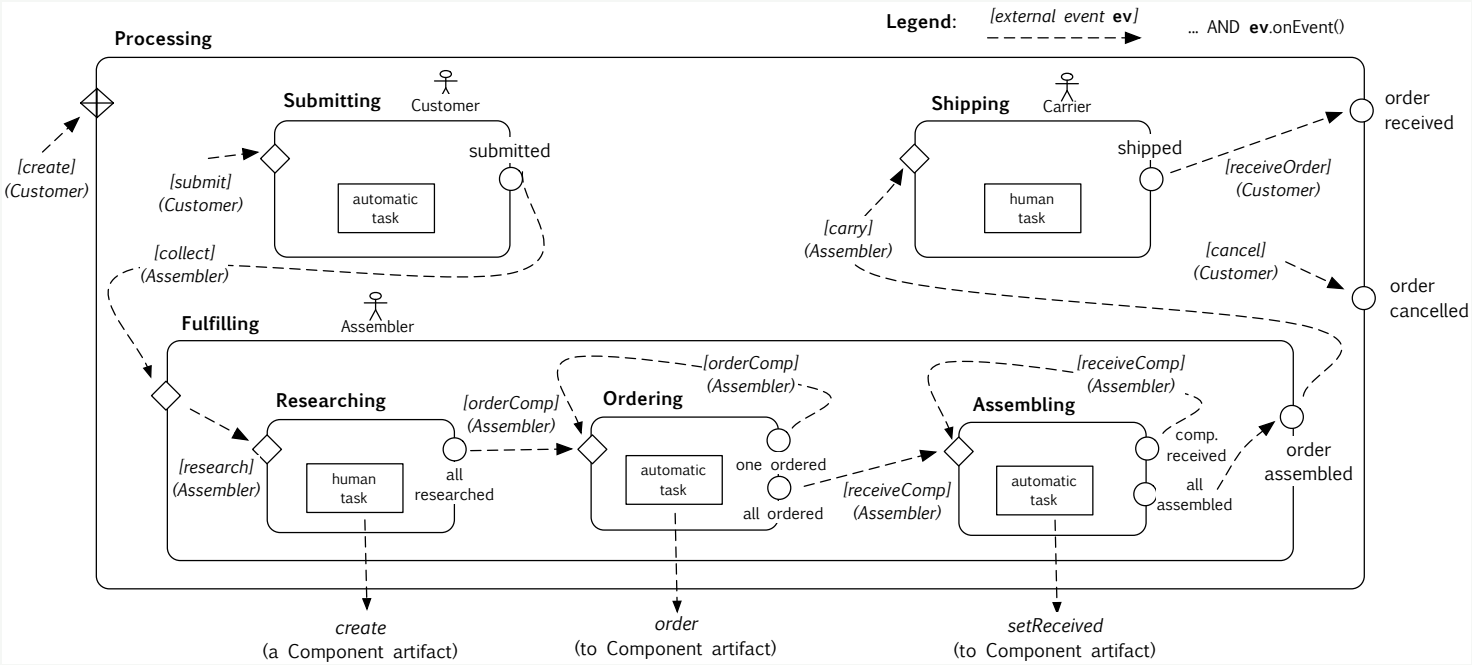
- **Relational database** (with nested records).
- **Knowledge base**, e.g., expressed in some ontology language.

Some concrete lifecycle models:

- **Finite-state machines**. State = phase; events trigger transitions.
  - Implemented in the *Siena* prototype by IBM.
- **Guard-Stage-Milestone** lifecycles, based on declarative (event-condition-action)-like rules.
  - Implemented in the *Barcelona* prototype by IBM.
- **Proclets** (interacting Petri nets).
  - Emphasise many-to-many relationships between artifacts.

Example: build-to-order using artifacts

Example: order as artifact (GSM diagram)



Glossary:

- *Stage* (rounded box): phase of the process.
- *Guard* (diamond): condition for activating a stage.
- *Milesone* (circle): condition for closing a stage.

Data model: information about domain and about state (e.g., achieved milestones)
Each guard/milestone is associated to a business rule on the data model.

Artifact approach: applications

Artifact-centric approach has been applied in several industrial settings:
[Bhattacharya *et al.*, 2005; Bhattacharya *et al.*, 2007a; Strosnider *et al.*, 2008]

Sigmod Tutorial (Thursday)

Richard Hull, Jianwen Su, Roman Vaculin
Data Management Perspectives on Business Process Management

Outline

Dimensions of the analysis problem space

We can consider variations of the analysis problem that differ along various dimensions:

1. Static information model
2. Dynamic component
3. Interaction between static and dynamic component
4. Interaction with environment
5. Analysis task

Outline

Approaches to dealing with processes and data

Key lines of research that have been pursued in DB theory:

- Database evolution and transactions
- Temporal data management
- Active databases
- Workflow formalisms and systems
- Temporal integrity constraints

Database evolution and transactions

"Dynamic" view of a DB seen as a **finite** sequence of instances investigated already in the 80's:

- Dynamic relational model [Vianu, 1983; 1984]
  - Dynamic constraints express dependencies between one state and the next.
  - Studies effect of dynamic constraints on static ones (FDs).
- Transactional DB schemas [Abiteboul and Vianu, 1985; 1986; 1987; 1988]
  - Valid DB states described using admissible (parameterized) transactions (i.e., finite sequences of inserts, deletes, updates, describing valid DB operations, later also iteration and new value creation)
  - Results concern expressive power (e.g., incomparable to constraint-based schemas), and undecidability of specific properties (e.g., soundness and completeness wrt constraints, reachability, erasability, equivalence).
  - Decidability is shown for restricted cases (insertions and deletions only, or under bounded-length transactions).

Temporal databases

Observation: temporal data management is difficult with conventional DBMSs.
$\rightsquigarrow$ Proposals for temporal extensions of query languages, e.g., relying on Allen's interval algebra [Snodgrass, 1984].

Problem to address: finite representation of infinite temporal data
$\rightsquigarrow$ query expressiveness vs. **data expressiveness** [Baudinet *et al.*, 1991]

1. Constraint based approach [Kabanza *et al.*, 1990]:
   - temporal attributes represent infinitely repeating points $z(n) = c + kn$;
   - constraints are conjunctions of linear (in)equalities on temporal attributes
   - queries are multi-sorted FO over relations with temporal attributes
   - queries are closed under relational algebra operations
2. Temporal deductive DBs, using successor function, and using sets of clauses to represent temporal query answers [Chomicki and Imielinski, 1988]
3. Templog instead uses temporal logic operators $\bigcirc$, $\Diamond$, $\Box$ (with restrictions) [Abadi and Manna, 1989; Baudinet, 1989] is a notational variant of (2)

Querying temporal databases

Two different approaches to querying a temporal DB using an FO-language.

1. Timestamped FO (TS-FO)

Consider the temporal DB as a single relational DB with a time-stamp column for each relation, plus a linear order over timestamps.

Example

$$\exists t_1 \exists t_2 \exists t_3 (t_1 < t_2 < t_2 \wedge \mathsf{Emp}(x, t_1) \wedge \neg \mathsf{Emp}(x, t_2) \wedge \mathsf{Emp}(x, t_3))$$

2. FO temporal logic (FO-LTL)

Use temporal operators ($\bigcirc$, $\Diamond$, $\Box$, $\mathcal{U}$, ...), and an implicit representation of time.

Example

$$\Diamond(\mathsf{Emp}(x) \wedge \Diamond(\neg \mathsf{Emp}(x) \wedge \Diamond \mathsf{Emp}(x)))$$

unibz.it

Querying temporal databases: comparison

FO-LTL is strictly less expressive than TS-FO [Abiteboul *et al.*, 1996].

Intuition: with an FO-LTL query, we can relate only a constant number of values in two different instances.

Example

The following TS-FO query cannot be expressed in FO-LTL:

$$\exists t_1 \exists t_2 (t_1 < t_2 \wedge \forall x (\mathsf{Employee}(x, t_1) \leftrightarrow \mathsf{Manager}(x, t_2)))$$

Note: $\Diamond(\forall x(\mathsf{Employee}(x) \leftrightarrow \Diamond \mathsf{Manager}(x)))$ fails, since there might not be a common timepoint where all employees have become managers.

Note: propositional LTL has the same expressive power as monadic FOL over (finite) linearly ordered structures [Kamp, 1968; Gabbay *et al.*, 1980].

Active databases

Studied from end of 80's:

- Described in terms of sets of event-condition-action rules operating on DB.
- Rules are triggered (under control of a condition) by external updates.
- Actions cause internal updates to the DB, which activate further rules.
- Control is passed back and forth between the external events and the trigger system.
- Initial research concentrated on execution by interpreters and their implementation [Kedem and Tuzhilin, 1989].

Several models and systems based on them have been defined, whose precise semantics was determined (procedurally) by the "execution model":

- immediate, deferred, or concurrent firing?
- which state should a previously triggered rule use?
- how to deal with events/conditions that do not hold any longer?

Active databases: formalization

[Picouet and Vianu, 1995; 1997] formalize and compare proposals using:

Relational machines (RTMs) [Abiteboul and Vianu, 1991; 1995]

- Turing Machines augmented with a **relational store** R (i.e., a finite set of fixed-arity relations).
- The TM interacts with R through FO queries and updates.

To formalize active DBs, pairs $\langle M_a, M_t \rangle$ of RTMs over the same R are used:

- M_t represents the active (trigger) DB.
- M_e represents the external update program, and has triggering states that transfer control to M_t.
- A run of $\langle M_t, M_e \rangle$ on a DB D for R computes an update of D by running M_t and M_e (each possibly multiple times) until they reach a final state.
- The semantics $M_t(M_e)$ of $\langle M_t, M_e \rangle$ is the computed update function.

Two active DBs M_t and M'_t over R are **equivalent**, if $M_t(M_e) = M'_t(M_e)$ for every external machine M_e over R.

Active databases: comparison of models

To compare active DB models, the previous definitions can be relativized:

- M_t and M'_t are **equivalent relative to an external language $\mathcal{E}$**, if $M_t(M_e) = M'_t(M_e)$ for every external machine M_e in $\mathcal{E}$.

By allowing M_t and M_e to vary within given languages $\mathcal{T}$ and $\mathcal{E}$, we can represent the combined expressive power:

$$\mathcal{T}(\mathcal{E}) \;=\; \{M_t(M_e) \mid M_t \in \mathcal{T} \text{ and } M_e \in \mathcal{E}\}$$

Based on this formalization and comparison criteria, [Picouet and Vianu, 1995; 1997]:

- provide a mechanism for comparing different proposals wrt expressive power;
- show that trigger machines capture various active DB models, e.g., Postgres, Sybase, Starburst, ARDL, HiPAC.

Outline

Data-aware business processes

We consider now approaches that tackle data-aware process analysis.

- Processes are considered at a higher level of abstraction.
- Explicit representation of the process as a separate component:
  - dynamic component, with procedural or declarative flavor
  - control portion of data
  - additional data manipulated by the process

Relational transducers

Is one of the most significant approaches proposed by the DB community to model high-level (business) processes:

- Model originally proposed to support forms of e-commerce [Abiteboul *et al.*, 1998; 2000].
- Explicitly accounts for a dynamic component on top of a full-fledged DB.

A relational transducer is a device transforming a sequence of input relations into a sequence of output relations:

- The state of a relational transducer contains
  - a static database, and
  - a memory dynamically changing over time.
- At each step of the computation, the transducer
  - receives from the environment input relations, and
  - produces output relations and updates the memory.

Relational transducers – Formalization

Relational transducer

Is a tuple $T = \langle S, \sigma, \omega \rangle$, where:

- $S = \langle db, mem, in, out, log \rangle$ is the schema, where $log \subseteq in \cup out$ maintains the meaningful portion of an input-output exchange;
- σ is a memory-update transition function mapping instances of $\langle db, mem, in \rangle$ to instances of mem;
- ω is an output-update transition function mapping instances of $\langle db, mem, in \rangle$ to instances of out.

Semantics is based on linear time: given database D for db:

- input sequence $I_1, \ldots, I_n$ represents interaction with external world.
- captures evolution of memory sequence $M_1, \ldots, M_n$ and output sequence $O_1, \ldots, O_n$, in response to input sequence:

$$S_i = \sigma(D, M_{i-1}, I_i) \qquad O_i = \omega(D, M_{i-1}, I_i) \qquad L_i = (I_i \cup O_i)|_{log}$$

Relational transducers – Example [Abiteboul *et al.*, 1998]

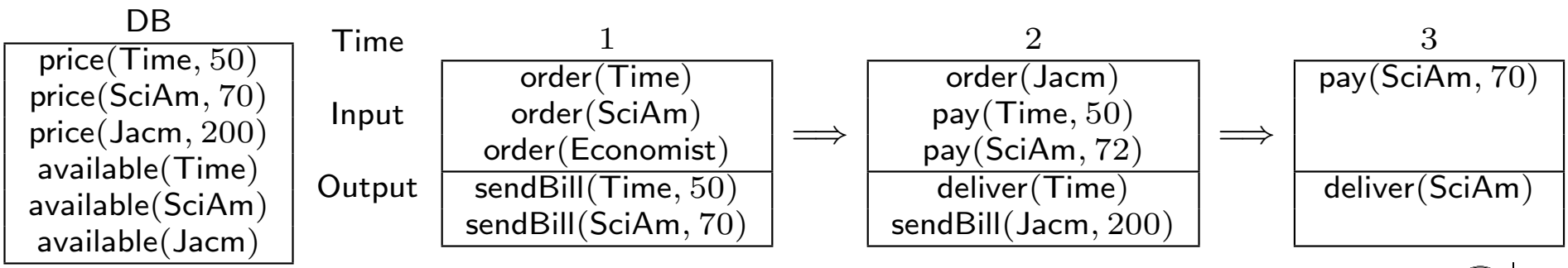
Relational transducer specification

Schema: db : price$_{/2}$, available$_{/1}$; in : order$_{/1}$, pay$_{/2}$;
mem : pastOrder$_{/1}$, pastPay$_{/2}$; out : sendBill$_{/2}$, deliver$_{/2}$;
log : pay, sendBill, deliver;

Memory: $\text{pastOrder}(X) \ +\!\leftarrow\ \text{order}(X)$;
$\text{pastPay}(X,Y) \ +\!\leftarrow\ \text{pay}(X,Y)$;

Output: $\text{sendBill}(X,Y) \leftarrow \text{order}(X), \text{price}(X,Y), \text{available}(X), \neg\text{pastPay}(X,Y)$;
$\text{deliver}(X) \leftarrow \text{pastOrder}(X), \text{price}(X,Y), \text{pay}(X,Y), \neg\text{pastPay}(X,Y)$.

When new input arrives, the rules are fired in parallel.
A run on a DB is completely determined by the sequence of inputs.



Relational transducers – Analysis tasks

Given a relational transducer T and an initial DB D:

- Finite log validity: Given a sequence $L_1, \ldots, L_n$ over log, is there an input sequence $I_1, \ldots, I_n$ generating it?

$$\exists I_1, \ldots, I_n((L_1, \ldots, L_n) = log(I_1, \ldots, I_n))$$

  Application: fraud detection

- Goal reachability: Is there a finite run of T on D such that a given sentence φ over out holds at the end of the run?

  Application: basic analysis task

Relational transducers – Analysis tasks (cont'd)

Given a relational transducer T and an initial DB D:

- Log containment (and equivalence): Given two transducers T, T' with the same log schema, is every finite log of T also a log of T' (and viceversa, for containment)?

$$\forall I_1, \ldots, I_n \exists J_1, \ldots, J_n (log(I_1, \ldots, I_n) = log(J_1, \ldots, J_n))$$

  Application: customization

Theorem

All the above problems are undecidable for unrestricted relational transducers.

Spocus transducers

A Spocus transducer is a relational transducer in which:

- The output relations are defined by non-recursive **s**emi-**po**sitive Datalog programs with inequality;
- The memory relations accumulate the inputs (**cu**mulative **s**tate).

Theorem ([Abiteboul *et al.*, 1998])

For Spocus transducers, log validity and reachability of conjunctive existential goals are decidable.

Proof: By a reduction to finite satisfiability for the Bernays-Schönfinkel prefix class $\exists^*\forall^* FO$, i.e., the set of FO sentences with relational vocabulary, constants, and equality of the form $\exists x_1 \cdots \exists x_m \forall x_{m+1} \cdots \forall x_n \, \varphi(x_1, \ldots, x_n)$.

Theorem ([Abiteboul *et al.*, 1998])

For Spocus transducers, log containment is undecidable, but becomes decidable if log contains in.

ASM transducers

Are defined by means of simple rule-based abstract state machine programs [Spielmann, 2000; 2003].

ASM transducers generalize Spocus transducers:

- Rule applications are guarded by FO formulas, as opposed to conjunctive formulas over literals.
- Memory is not necessarily cumulative (can be considered as an active DB, with immediate triggering of insertion and deletion actions).

ASM transducer

Is a relational transducer whose program is a finite set of rules of the form

$$\textbf{if } \varphi(\vec{x}) \textbf{ then } (\neg)R(\vec{x})$$

where

- the guard $\varphi(\vec{x})$ of the rule is a FO formula over $db \cup mem \cup in$;
- $R(\vec{x})$ is an atomic FO formula over $mem \cup out$;
- if $R \in out$, then it occurs positively in the rhs.

Uses one step-semantics of Datalog$^{\neg\neg}$, but ignoring inconsistent updates.

ASM transducers – Expressive power

ASM transducers are strictly more expressive than Spocus transducers.

Theorem ([Spielmann, 2000])

On ordered DBs, ASM transducers compute precisely the class of PSPACE-computable boolean queries.

However, this is **too powerful for analysis**: by Trakhtenbroth's theorem, log validity, verification of temporal properties (e.g., goal reachability) and containment are all undecidable for ASM transducers

Consider restrictions on analysis

1. The initial database is assumed fixed and given.
2. The amount of input data at each computation step is bounded a priori.

These restrictions **ensure decidability** of the considered analysis tasks. Complexity is PSPACE-complete for fixed arity, and in EXPSPACE otherwise.

From relational transducers to data driven web systems

Several kinds of web applications:

- provide distributed access to information stored on the web,
- are driven by underlying databases,
- are manipulated by complex web applications/services that interact with external services/users.

$\rightsquigarrow$ Lend themselves well for data-aware analysis.

First proposal by [Deutsch *et al.*, 2004; 2007]

- considers a single web service interacting with a user
- studies the trade-off between expressiveness of the specification and feasability of verification.
- Extend ASM transducers to model web services:
  - a database that remains fixed during the execution,
  - a set of state relations that evolve in response to user inputs,
  - a set of web page schemas that query the current database and state to generate user input choice, and
  - state transitions that are triggered by the input chosen by the user.

unibz.it

Data driven web systems

Again, restrictions are necessary to address analysis tasks:

- **Input-bounded quantification** in rules evolving system.
- Formulas of input rules are existential.
- Adds possibility of referring to input at the previous step in the run.

[Deutsch *et al.*, 2004] study verification for LTL with input-bounded quantification:

- in ExpSpace
- PSpace-complete for fixed arity

Upper bound: reduction to finite satisfiability of existential FO augmented with transitive closure.

Consider also CTL and CTL*, but decidability requires further restrictions:

- states and actions are propositional (inputs are still parametrized), and previous input relations are not allowed;

Effective implementation carried out in the WAVE system [Deutsch *et al.*, 2005]

unibz.it

Artifact-centric systems and AXML

Analysis has been studied also for artifact-centric systems:

- Artifacts based on finite-state-machines [Gerede and Su, 2007], with dynamic creation/modification/elimination of artifacts.
  - Verification of CTL with quantification across states.
  - Decidability shown under assumptions that ensure boundedness of the quantification domains.
- More declarative approach is followed in [Bhattacharya *et al.*, 2007b], based on business rules that activate services (with preconditions and non-deterministic effects).
  - Predefined reasoning tasks are addressed (path completion, path existence, ...), and decidability is shown under restrictions.

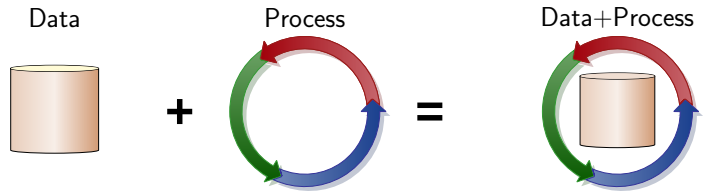
Active XML [Abiteboul *et al.*, 2004]

- Artifact-centric paradigm based on XML documents with parts that are intensionally specified, through calls to external services.
- Very rich setting, in which several analysis results have been established, [Abiteboul *et al.*, 2008; 2009a; 2009b]:

unibz.it

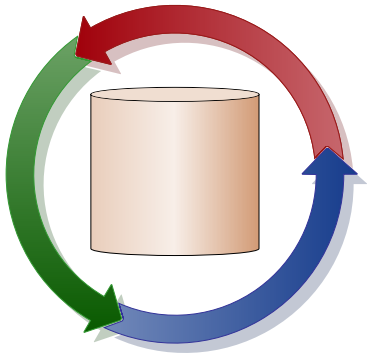
Data-Centric Dynamic Systems (DCDS)

- Abstract model underlying variants of artifact-centric systems.
- Semantically equivalent to the most expressive models for business process systems (e.g., GSM).



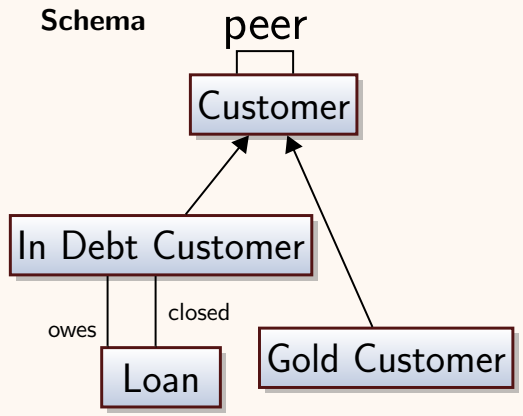
- Data Layer: Relational databases / ontologies
  - Data schema, specifying constraints on the allowed states
  - Data instance: state of the DCDS
- Process Layer: key elements are
  - Atomic actions
  - Condition-action-rules for application of actions
  - **Service calls:** communication with external environment, **new data!**

DCDS: Example



Data Layer

Schema



Instance

$\mathsf{Cust}(\mathsf{ann})$
$\mathsf{peer}(\mathsf{mark}, \mathsf{john})$
$\mathsf{Gold}(\mathsf{john})$
$\mathsf{owes}(\mathsf{mark}, @25)$
...

Process Layer

Condition Action Rules

$$\mathsf{peer}(x, y) \wedge \mathsf{Gold}(y) \longmapsto \mathsf{GetLoan}(x)$$

Service Calls

$$\mathit{UInput}(x)$$

Actions

$\mathsf{GetLoan}(x)$:

$$\begin{aligned}
\exists y.\mathsf{peer}(x, y) &\rightsquigarrow \{\mathsf{owes}(x, \mathit{UInput}(x))\},\\
\mathsf{Cust}(z) &\rightsquigarrow \{\mathsf{Cust}(z)\},\\
\mathsf{Loan}(z) &\rightsquigarrow \{\mathsf{Loan}(z)\},\\
\mathsf{InDebt}(z) &\rightsquigarrow \{\mathsf{InDebt}(z)\},\\
\mathsf{Gold}(z) &\rightsquigarrow \{\mathsf{Gold}(z)\}
\end{aligned}$$

unibz.it

Deterministic vs. non-deterministic services

DCDSs admit two different semantics for service-execution:

Deterministic services semantics

Along a run, when the same service is called again with the same arguments, it returns the same result as in the previous call.

Are used to model an environment whose behavior is completely determined by the parameters.

Example: temperature, given the location and the date and time

Non-deterministic services semantics

Along a run, when the same service is called again with the same arguments, it may return a different value than in the previous call.

Are used to model:

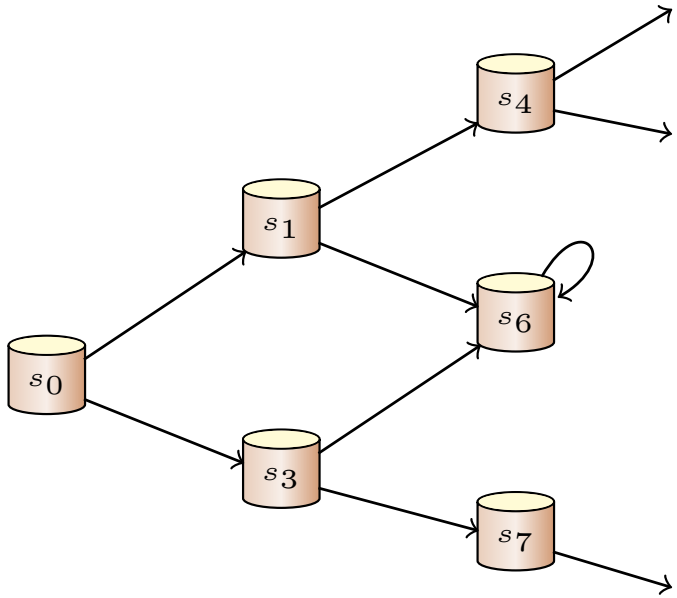
- an environment whose behavior is determined by parameters that are outside the control of the system;
- input of external users, whose choices depend on external factors.

Example: current temperature, given the location

Semantics via transition systems

Semantics of a DCDS $\mathcal{S}$ is given in terms of a **transition system** $\Upsilon_{\mathcal{S}}$:

- each state of $\Upsilon_{\mathcal{S}}$ has an associated DB over a common schema;
- the initial state is associated to the initial DB of the DCDS.



Note: $\Upsilon_{\mathcal{S}}$ is in general **infinite state**:

- infinite branching, due to the results of service calls,
- infinite runs, since infinitely many DBs may occur along a run;
- the DBs associated to the states are of unbounded size.

Verification for DCDSs

We are interested in the **verification** of temporal properties over $\Upsilon_{\mathcal{S}}$.

Idea to overcome infiniteness:

1. Devise a **finite-state** transition system $\Theta_{\mathcal{S}}$ that is a **faithful abstraction** of $\Upsilon_{\mathcal{S}}$ **independent of the formula** to verify.
2. Reduce the verification problem $\Upsilon_{\mathcal{S}} \models \Phi$ to the verification of $\Theta_{\mathcal{S}} \models \Phi$.

Problem: Verification of DCDSs is undecidable even for propositional reachability properties.
$\rightsquigarrow$ We need to pose restrictions on DCDSs.

We could draw inspiration from chase termination for tgds in data exchange, and specifically from weak-acyclicity.

Restrictions on DCDSs

Run-bounded DCDS

Runs cannot accumulate more than a fixed number of different values.

- Transition system is still infinite-state due to infinite branching.
- This is a semantic condition, whose checking is undecidable.
  $\leadsto$ Sufficient syntactic condition: Weak-acyclicity.
- Run-boundedness is very restrictive for DCDSs with nondeterministic services.

State-bounded DCDS

States cannot contain more than a fixed number of different values.

- Relaxation of run-boundedness.
- Infinite runs are possible.
- This is a semantic condition, whose checking is undecidable.
  $\leadsto$ Sufficient syntactic condition: GR-acyclicity.

Verification formalisms for DCDSs

Boundedness is not sufficient for decidability.
We introduce two extensions of $\mu\mathcal{L}$ with (restricted) first order quantification.

History-Preserving μ-calculus: $\mu\mathcal{L}_A$

FO quantification ranges over current active domain only.

Examples:

LTL_{FO} : $\forall x.\ \mathsf{Customer}(x) \rightarrow \mathbf{F}\ \mathsf{Gold}(x)$

$\mu\mathcal{L}_A$: $\forall x.\ \mathsf{Customer}(x) \rightarrow \mu Z.\mathsf{Gold}(x) \vee [-]Z$

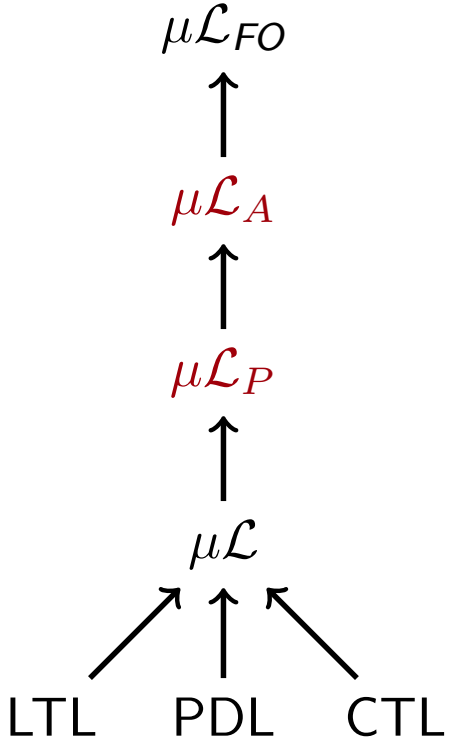
Persistence-Preserving μ-calculus: $\mu\mathcal{L}_P$

FO quantification ranges over persisting individuals only.

Examples:

LTL_{FO} : $\forall x.\ \mathsf{Gold}(x) \rightarrow \mathbf{G}\ \mathsf{Gold}(x)$

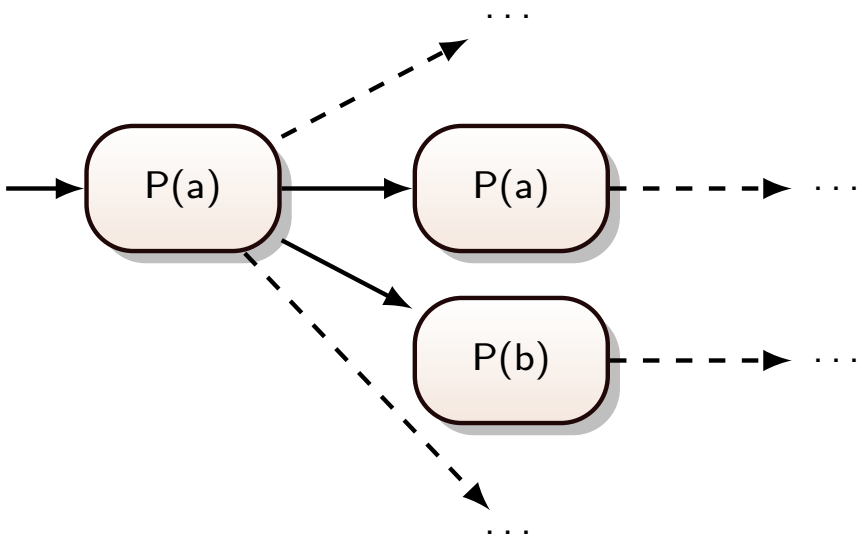
$\mu\mathcal{L}_P$: $\forall x.\ \mathsf{Gold}(x) \rightarrow \nu Z.\mathsf{Gold}(x) \wedge [-]Z$



Towards decidability

We need to tame the two sources of infinity in DCDSs:

- infinite branching
- infinite runs.



To prove decidability of model checking for a given restriction and verification formalism:

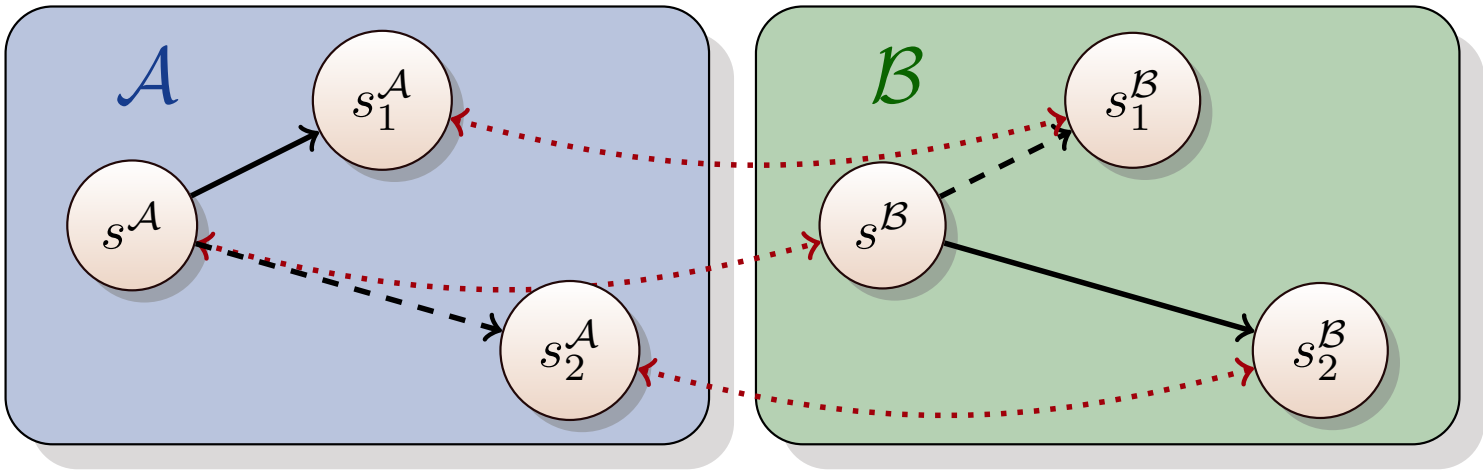
- We use bisimulation as a tool.
- We show that restricted DCDSs have a finite-state bisimilar transition system.

Bisimulation between transition systems

States $s^{\mathcal{A}}$ and $s^{\mathcal{B}}$ of transition systems $\mathcal{A}$ and $\mathcal{B}$ are bisimilar if:

1. $s^{\mathcal{A}}$ and $s^{\mathcal{B}}$ are isomorphic;
2. If there exists a state $s_1^{\mathcal{A}}$ of $\mathcal{A}$ such that $s^{\mathcal{A}} \Rightarrow_{\mathcal{A}} s_1^{\mathcal{A}}$, then there exists a state $s_1^{\mathcal{B}}$ of $\mathcal{B}$ such that $s^{\mathcal{B}} \Rightarrow_{\mathcal{B}} s_1^{\mathcal{B}}$, and $s_1^{\mathcal{A}}$ and $s_1^{\mathcal{B}}$ are bisimilar;
3. The other direction!

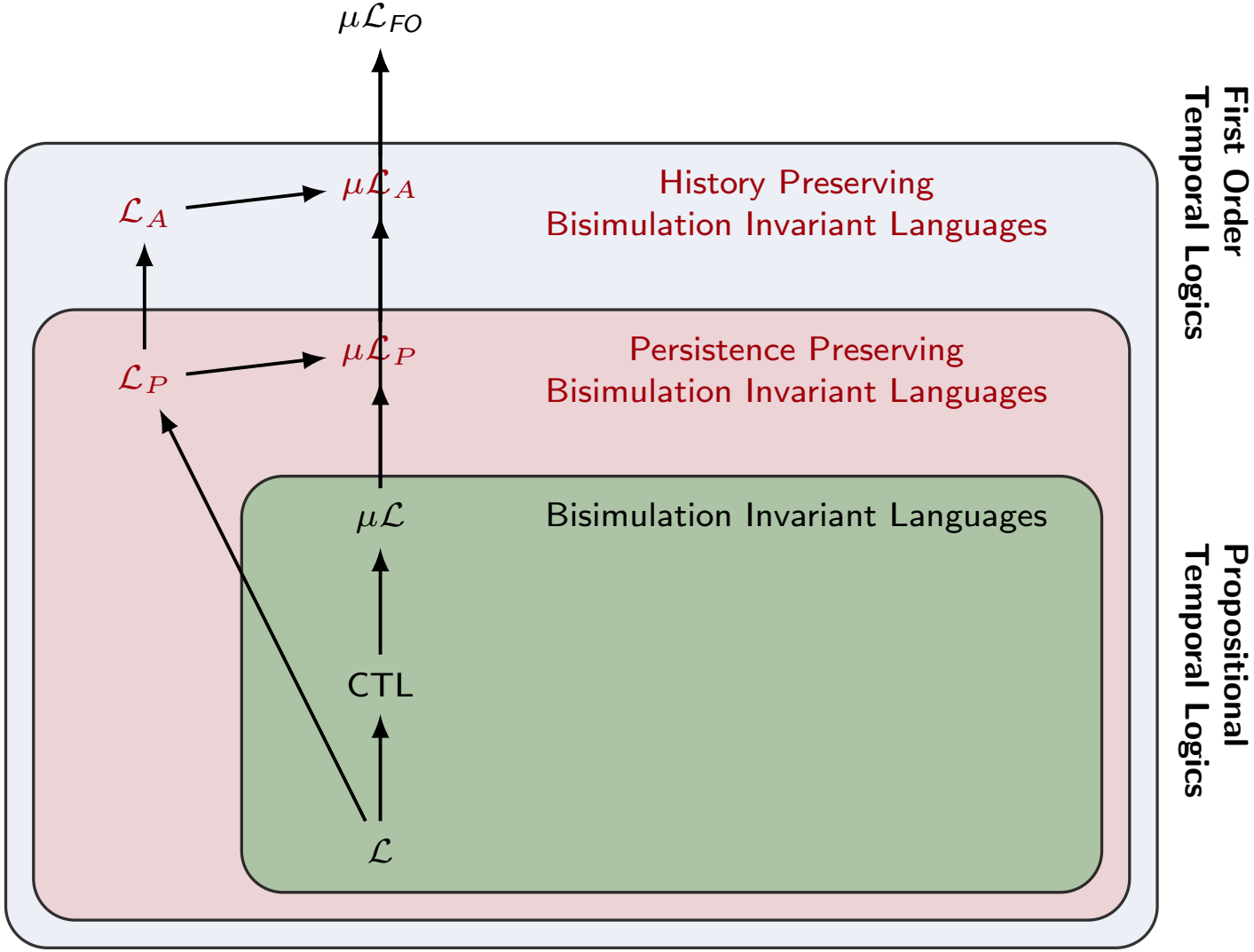
$\mathcal{A}$ and $\mathcal{B}$ are bisimilar, if their initial states are bisimilar.



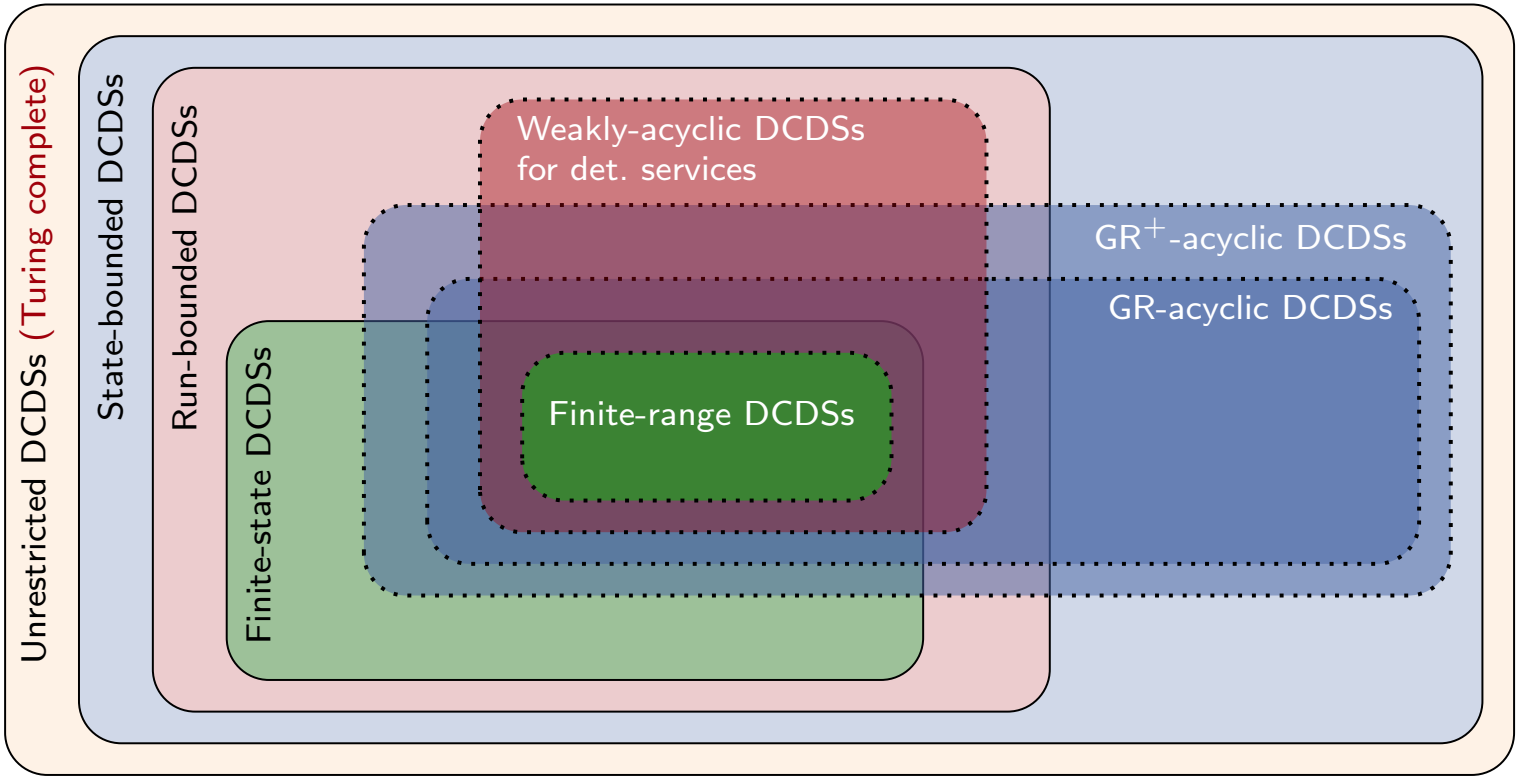
$\mu\mathcal{L}$ invariance property of bisimulation:

Bisimilar transition systems satisfy the same set of $\mu\mathcal{L}$ properties.

Adapting the notion of bisimulation



Results on verification for DCDSs



	Unrestricted	State-bounded	Run-bounded	Finite-state
$\mu\mathcal{L}_{FO}$	**U**	**U**	**N**	**D**
$\mu\mathcal{L}_{A}$	**U**	**U**	**D**	**D**
$\mu\mathcal{L}_{P}$	**U**	**D**	**D**	**D**
$\mu\mathcal{L}$	**U**	**D**	**D**	**D**

D: decidable; **U**: undecidable; **N**: no finite abstraction.

Uniformity

Due to the semantics of service calls, DCDS enjoy a sort of uniformity.

Uniformity (intuition)

A node of $\Upsilon_{\mathcal{S}}$ has as successor nodes all nodes obtained in all possible ways by substituting service calls with a resulting value.

It follows that successor states are "indistinguishable" from each other, modulo isomorphisms on the results of service calls.

Note: In [Belardinelli *et al.*, 2011; 2012], uniformity is exploited, to obtain decidability of verification of a form of CTL_A (i.e., CTL with **active domain** FO quantification) over **state**-bounded systems.

Compare this with undecidability of verification of $\mu\mathcal{L}_A$ over state-bounded systems.

Outline

Knowledge and Action Bases (KAB)

The data layer in an artifact system might be very complex, and difficult to interact with.

Hence we might exploit ontology-based technology and ontology-based data access techniques to support users:

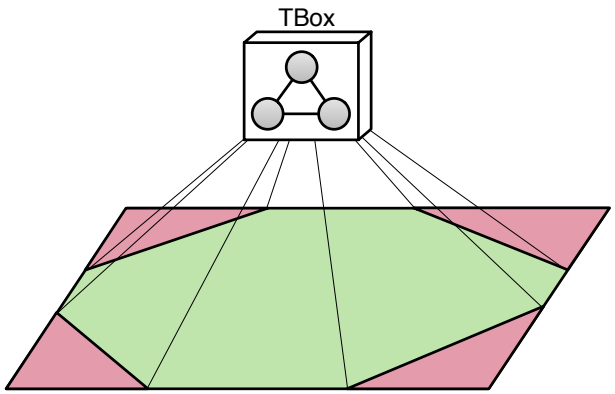
- We install "on top" of an artifact system an ontology, capturing the domain of interest at a higher level of abstraction.
- We connect the ontology to the underlying artifact system via declarative mappings.

Such a setting gives rise to a very rich and largely unexplored framework, depending on the various choices for:

- the language used to express the ontology;
- the form of the mappings, and the language used to express them;
- the assumptions we make about the dynamics of the system;
- the kind of analysis task we want to perfom.

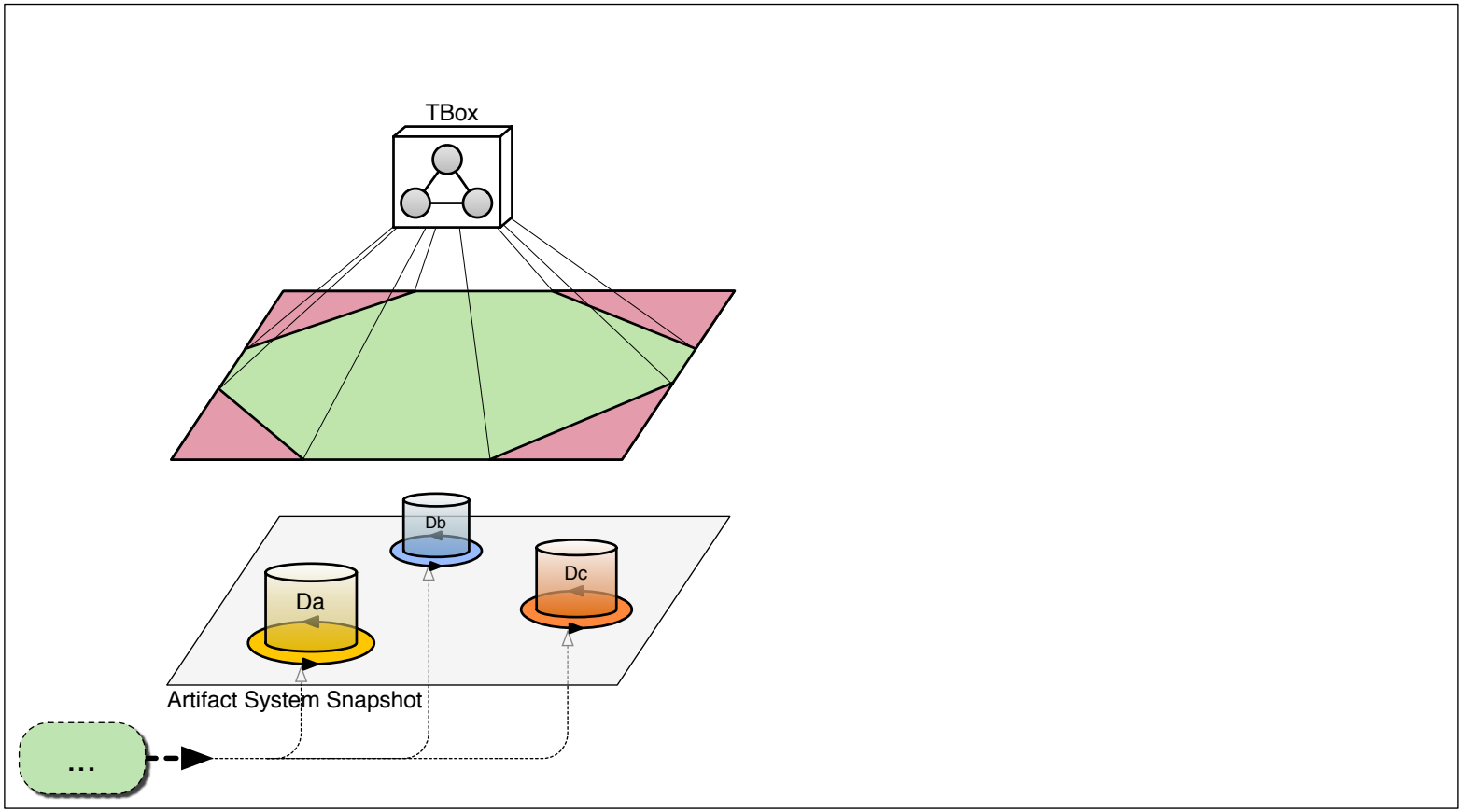
Knowledge and Action Bases (KAB)

Artifacts System conceptual schema (TBox) composed of semantic constraints that define the "data boundaries" of the artifact system.



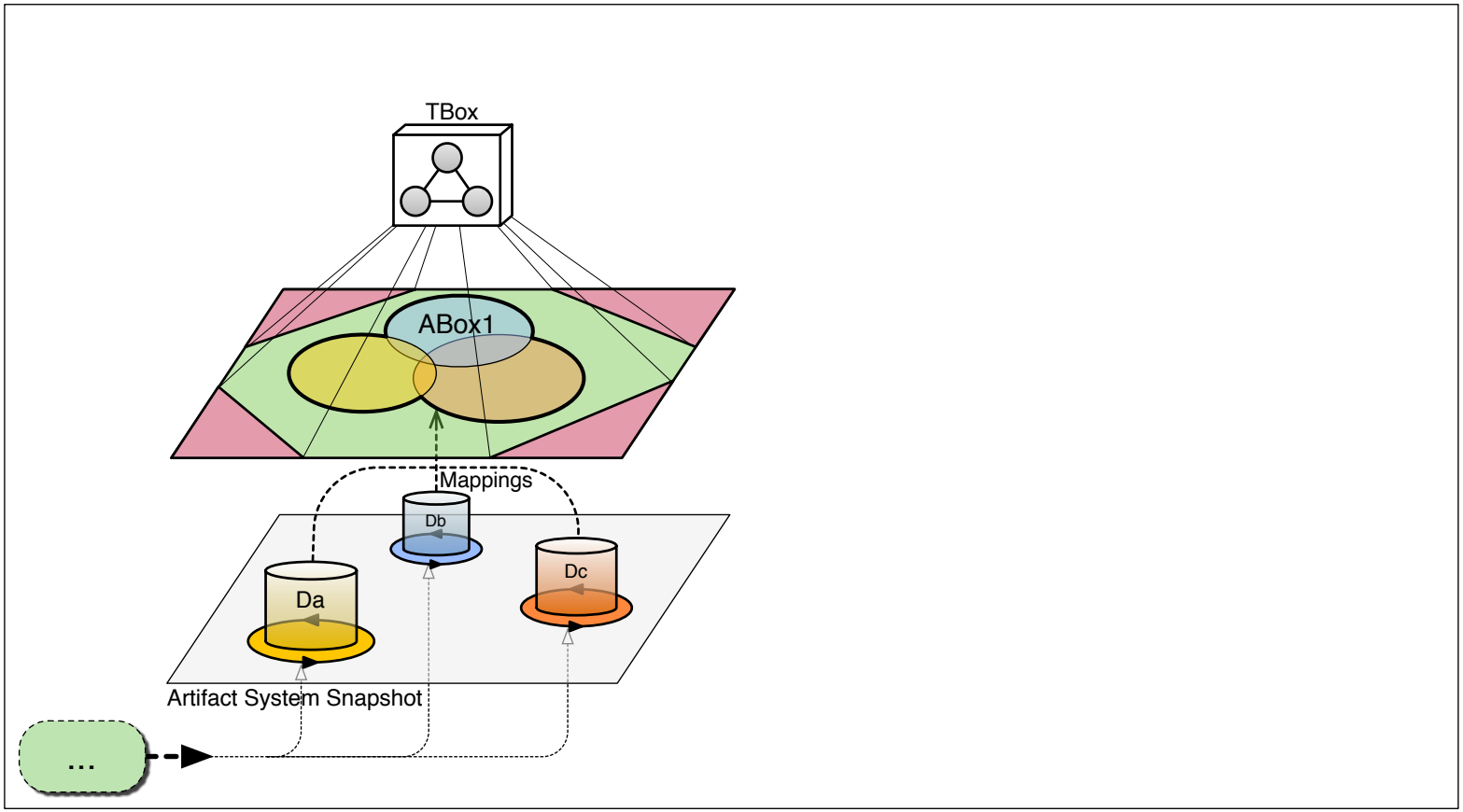
Semantic layer and snapshots

Actual data are concretely maintained at the artifact layer.
Snapshot: database instances of artifacts.



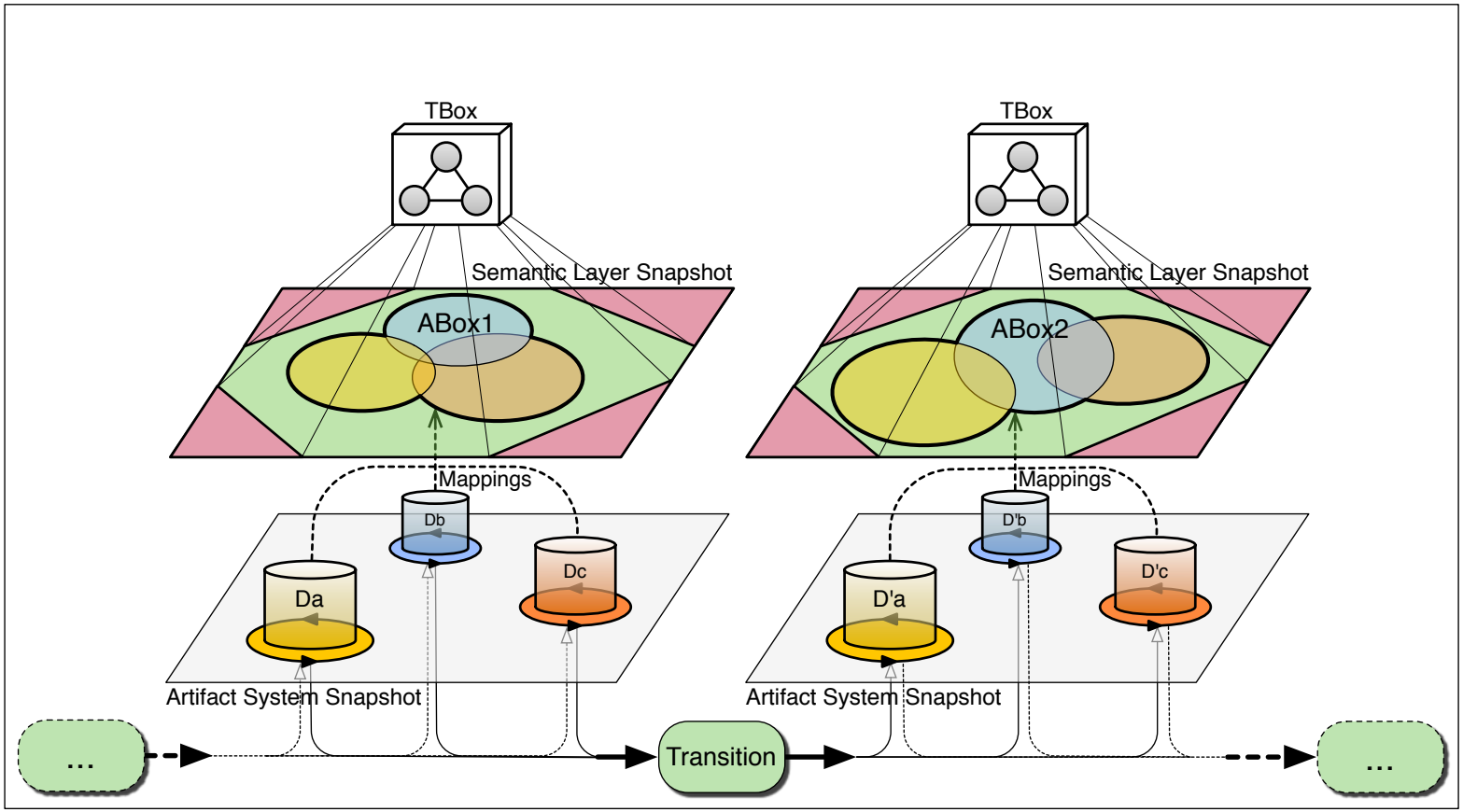
Mappings

Each snapshot is conceptualized in the ontology as instance data.
Mappings define how to obtain the virtual ABox from the source data.



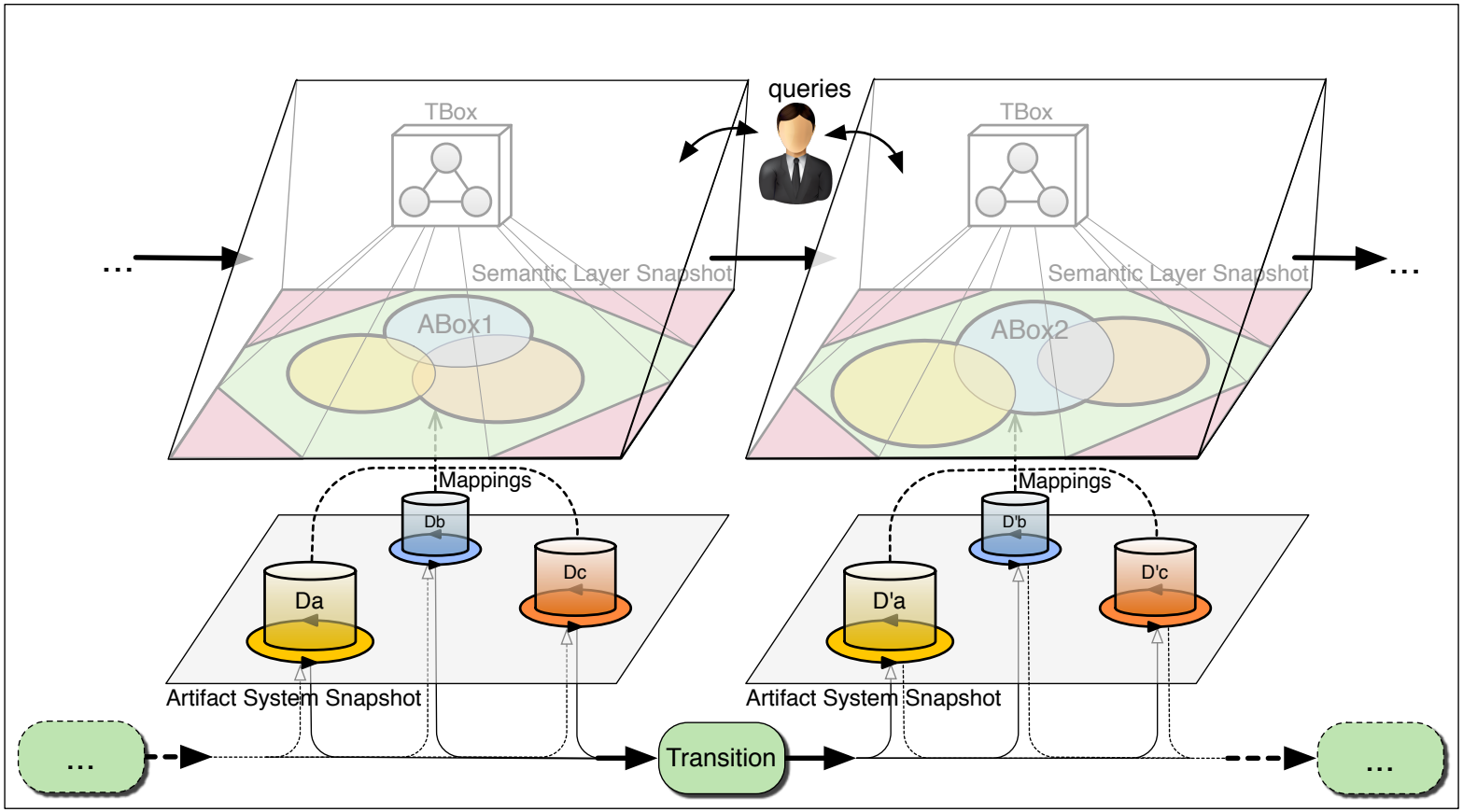
Action execution to evolve the system

The system evolves thanks to actions/process executed over the artifact layer, invoking external services to inject new data.



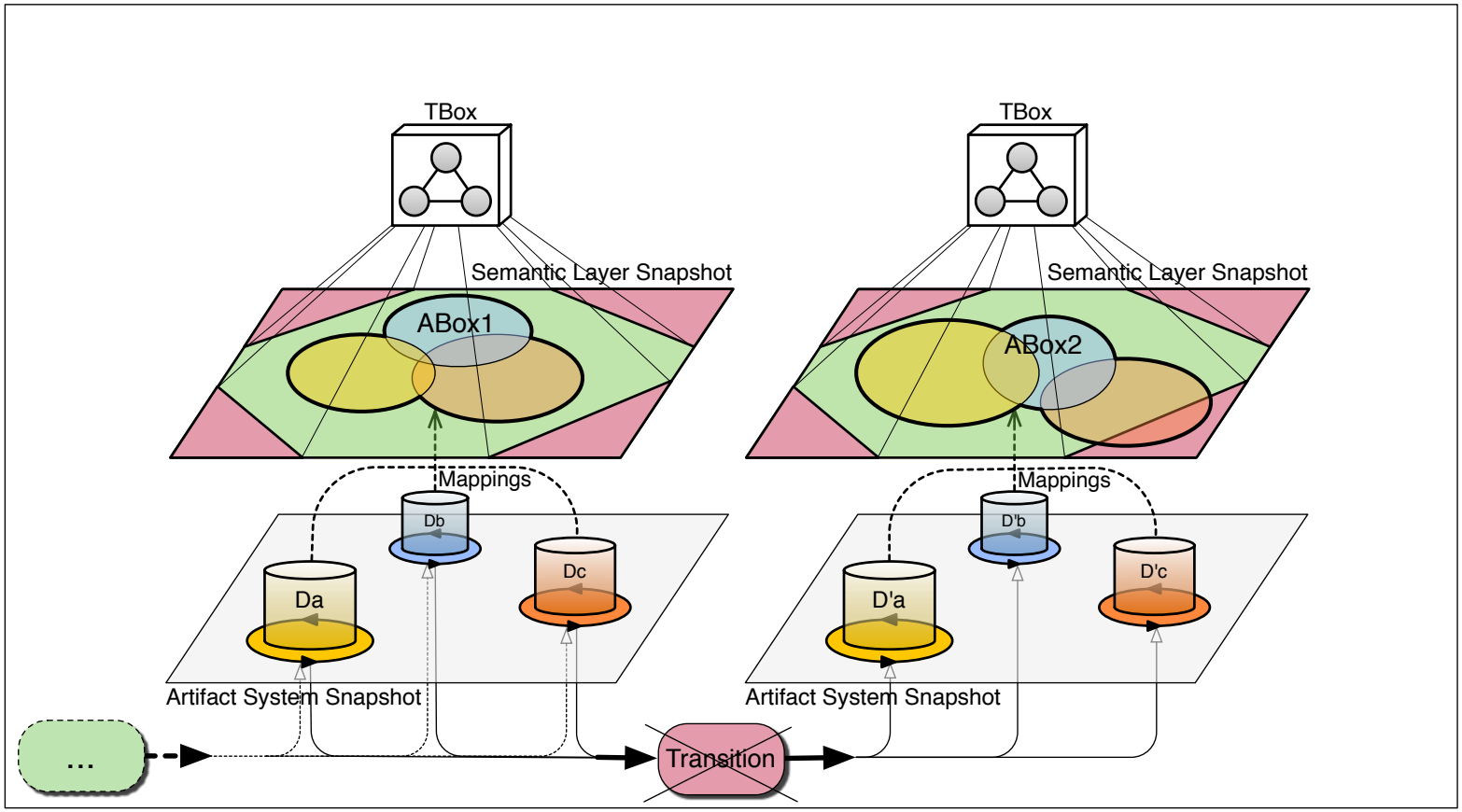
Understanding the evolution

Semantic layer used to **understand** the evolution at the conceptual level, by posing queries over the ontology.



Semantic Governance

Semantic layer used to regulate the execution of actions at the artifact layer by **rejecting actions that lead to violations of constraints** in the ontology.



Temporal Verification over Semantic Layer

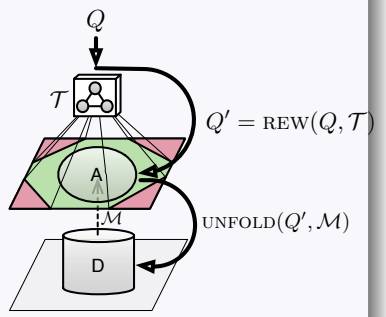
Temporal properties expressed as:

- queries over the ontology combined with
- temporal operators to talk about the dynamics of the system.

System evolves at the Artifact Layer.

Rewriting of temporal properties

- The temporal part is maintained unaltered, because the system evolves at the Artifact Layer.
- Faithful transformation of a temporal property over Semantic Layer:
  1. **Rewriting** of ontology queries to compile away the TBox.
  2. **Unfolding** of temporal property wrt mappings to obtain a corresponding temporal property over the Artifact Layer.



Hence, verification of temporal properties expressed over the ontology is reduced to verification of temporal properties over the artifacts.

Decidability of Verification over KABs

We obtain that, if the transition system at Artifact Layer satisfies suitable boundedness conditions, verification of (restricted) μ-calculus properties is decidable.

Theorem

The following are decidable, and can be reduced to model checking of propositional μ-calculus over a finite transition system:

- Verification of $\mu\mathcal{L}_A$ properties for weakly acyclic SASs with deterministic services.
- Verification of $\mu\mathcal{L}_P$ properties for generate-recall acyclic SASs (both with deterministic and with non-deterministic services).

Outline

Conclusions

- There is a huge amount of work carried out in database theory that is relevant to data-aware process analysis, using a plethora of techniques.
- The problem space has several dimensions that partly interact.
  $\rightsquigarrow$ Thorough systematization of the area is still missing.
- Many of the works are based on specific restrictions and assumptions that make them difficult to compare.
- Moreover, the positive results appear rather fragile.
- Analysis techniques are typically exponential in those data that "change"
  $\rightsquigarrow$ Circumscribing what can be changed is a key point.
- The assumptions would need validation also from the practical and business perspective.
  $\rightsquigarrow$ Requires making frameworks more robust.
- Some of the techniques are borrowed from different fields, although underlying assumptions and objectives might be different.
  $\rightsquigarrow$ Basic assumptions need to be reassessed.

Acknowledgements

Thanks to the many people that contributed interesting ideas, suggestions, discussions, and that collaborated to the presented results:

Giuseppe De Giacomo
Marco Montali

Babak Bagheri Hariri
Riccardo De Masellis
Alin Deutsch
Paolo Felli
Rick Hull
Maurizio Lenzerini
Alessio Lomuscio
Fabio Patrizi
Jianwen Su
Moshe Vardi

The ACSI Consortium

Thank you for your attention!

References I

[Abadi and Manna, 1989] Martín Abadi and Zohar Manna.

Temporal logic programming.

J. of Symbolic Computation, 8(3):277–295, 1989.

[Abiteboul and Vianu, 1985] Serge Abiteboul and Victor 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.

[Abiteboul and Vianu, 1986] Serge Abiteboul and Victor 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.

[Abiteboul and Vianu, 1987] Serge Abiteboul and Victor 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.

References II

[Abiteboul and Vianu, 1988] Serge Abiteboul and Victor 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.

[Abiteboul and Vianu, 1991] Serge Abiteboul and Victor Vianu.

Generic computation and its complexity.

In *Proc. of the 23rd ACM SIGACT Symp. on Theory of Computing (STOC'91)*, pages 209–219, 1991.

[Abiteboul and Vianu, 1995] Serge Abiteboul and Victor Vianu.

Computing with first-order logic.

J. of Computer and System Sciences, 50(2):309–335, 1995.

[Abiteboul *et al.*, 1996] Serge Abiteboul, Laurent Herr, and Jan 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.

References III

[Abiteboul *et al.*, 1998] Serge Abiteboul, Victor Vianu, Bradley Fordham, and Yelena 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.

[Abiteboul *et al.*, 2000] Serge Abiteboul, Victor Vianu, Bradley Fordham, and Yelena Yesha.
Relational transducers for electronic commerce.
J. of Computer and System Sciences, 61(2):236–269, 2000.

[Abiteboul *et al.*, 2004] Serge Abiteboul, Omar Benjelloun, and Tova 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.

[Abiteboul *et al.*, 2008] Serge Abiteboul, Luc Segoufin, and Victor 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.

References IV

[Abiteboul *et al.*, 2009a] Serge Abiteboul, Luc Segoufin, and Victor Vianu.

Modeling and verifying Active XML artifacts.

Bull. of the IEEE Computer Society Technical Committee on Data Engineering, 32(3):10–15, 2009.

[Abiteboul *et al.*, 2009b] Serge Abiteboul, Luc Segoufin, and Victor Vianu.

Static analysis of Active XML systems.

ACM Trans. on Database Systems, 34(4):23:1–23:44, 2009.

[Baudinet *et al.*, 1991] Marianne Baudinet, Marc Niézette, and Pierre 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.

[Baudinet, 1989] Marianne 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.

References V

[Belardinelli *et al.*, 2011] Francesco Belardinelli, Alessio Lomuscio, and Fabio 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.

[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 Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012)*, 2012.

[Bhattacharya *et al.*, 2005] Kamal Bhattacharya, Robert Guttman, Kelly Lyman, Fenno F. Heath, Santhosh Kumaran, Prabir Nandi, Frederick Y. Wu, Prasanna Athma, Christoph Freiberg, Lars Johannsen, and Andreas Staudt.

A model-driven approach to industrializing discovery processes in pharmaceutical research.

IBM Systems Journal, 44(1):145–162, 2005.

[Bhattacharya *et al.*, 2007a] Kamal Bhattacharya, Nathan S. Caswell, Santhosh Kumaran, Anil Nigam, and Frederick Y. Wu.

Artifact-centered operational modeling: Lessons from customer engagements.

IBM Systems Journal, 46(4):703–721, 2007.

unibz.it

References VI

[Bhattacharya *et al.*, 2007b] Kamal Bhattacharya, Cagdas Evren Gerede, Richard Hull, Rong Liu, and Jianwen 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.

[Chomicki and Imielinski, 1988] Jan Chomicki and Tomasz 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.

[Deutsch *et al.*, 2004] Alin Deutsch, Liying Sui, and Victor 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.

[Deutsch *et al.*, 2005] Alin Deutsch, Monica Marcus, Liying Sui, Victor Vianu, and Dayou 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.

unibz.it

References VII

[Deutsch *et al.*, 2007] Alin Deutsch, Liying Sui, and Victor Vianu.

Specification and verification of data-driven web applications.

J. of Computer and System Sciences, 73(3):442–474, 2007.

[Esparza, 1997] Javier Esparza.

Decidability of model checking for infinite-state concurrent systems.

Acta Informatica, 34(2):85–107, 1997.

[Esparza, 1998] Javier 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.

[Gabbay *et al.*, 1980] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi.

On the temporal analysis of fairness.

In *Proc. of the 7th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL'80)*, pages 163–173, 1980.

References VIII

[Gerede and Su, 2007] Cagdas E. Gerede and Jianwen 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.

[Kabanza *et al.*, 1990] Froduald Kabanza, Jean-Marc Stévenne, and Pierre 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.

[Kamp, 1968] H. Kamp.

On Tense Logic and the Theory of Order.

PhD thesis, University of California Los Angeles, 1968.

[Karel *et al.*, 2009] Rob Karel, Clay Richardson, and Connie Moore.

Warning: Don't assume your business processes use master data – Synchronize your business process and master data strategies.

Report, Forrester, September 2009.

References IX

[Kedem and Tuzhilin, 1989] Zvi M. Kedem and Alexander 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.

[Meyer *et al.*, 2011] Andreas Meyer, Sergey Smirnov, and Mathias 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/`.

[Nigam and Caswell, 2003] Anil Nigam and Nathan S. Caswell.

Business artifacts: An approach to operational specification.

IBM Systems Journal, 42(3):428–445, 2003.

[Picouet and Vianu, 1995] Philippe Picouet and Victor 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.

References X

[Picouet and Vianu, 1997] Philippe Picouet and Victor 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.

[Reichert, 2012] Manfred 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.

[Snodgrass, 1984] Richard 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.

[Spielmann, 2000] Marc 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.

References XI

[Spielmann, 2003] Marc Spielmann.

Verification of relational transducers for electronic commerce.

J. of Computer and System Sciences, 66(1):40–65, 2003.

[Strosnider *et al.*, 2008] Jay K. Strosnider, Prabir Nandi, Santhosh Kumaran, Sakti P. Ghosh, and Ali Arsanjani.

Model-driven synthesis of SOA solutions.

IBM Systems Journal, 47(3):415–432, 2008.

[van der Aalst, 2012] Wil 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.

[Vianu, 1983] Victor Vianu.

Dynamic Constraints and Database Evolution.

PhD thesis, University of Southern California, 1983.

References XII

[Vianu, 1984] Victor 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.