When Data, Processes and Knowledge Meet Together

Ario Santoso
Supervisor: Diego Calvanese

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano
lastname@inf.unibz.it

Acknowledgement: process and data team in KRDB for some slides in this presentation
Outline

1. Marriage Between Knowledge and Data
   - Ontology Based Data Access (OBDA)

2. Data Meet Processes
   - Artifact-Centric Approach
   - Data-Centric Dynamic Systems (DCDS)

3. Embracing Knowledge into Data-aware Processes
   - Semantically-Governed Data-Aware Processes (SGDAP)
   - SGDAP Formal Definition
   - SGDAP Semantics
   - SGDAP in Action

4. What’s Next?
Marriage Between Knowledge and Data
Query Answering: in Ontology Vs in Traditional Database

“Give me all persons who have a father”

\[ q(x) : = \exists y. \text{hasFather}(x, y) \]

- Person(Po), hasFather(Po, Tinkywinky)
- Person(Lala), hasFather(Lala, Tinkywinky)
- Person(Tinkywinky)
Query Answering: in Ontology Vs in Traditional Database

“All person must have a father”

“Give me all persons who have a father”

\[ q(x) : \exists y. \text{hasFather}(x, y) \]

Person(Po), hasFather(Po, Tinkywinky)
Person(Lala), hasFather(Lala, Tinkywinky)
Person(Tinkywinky)
Query Answering: in Ontology Vs in Traditional Database

```
Person ⊆ ∃hasParent
∃hasParent ⊆ Person

“Give me all persons who have a father”
q(x) :- ∃y.hasFather(x, y)
```

Conceptual Schema/Ontology

Data Store(s)

Query

Result(s)

Person(Po), hasFather(Po, Tinkywinky)
Person(Lala), hasFather(Lala, Tinkywinky)
Person(Tinkywinky)

Lala
Po
Tinkywinky
Outline

1. Marriage Between Knowledge and Data
   - Ontology Based Data Access (OBDA)

2. Data Meet Processes
   - Artifact-Centric Approach
   - Data-Centric Dynamic Systems (DCDS)

3. Embracing Knowledge into Data-aware Processes
   - Semantically-Governed Data-Aware Processes (SGDAP)
   - SGDAP Formal Definition
   - SGDAP Semantics
   - SGDAP in Action

4. What’s Next?
Ontology Based Data Access (OBDA)

- Idea: using ontologies as a conceptual view over data repositories.

Query Answering + Reasoning

- Ontologies provide a conceptualization of a domain interest.
- Unified view among various data sources.
- Imposes constraints on the data over the conceptual level.
- Helps to infer implicit information.
- Enhancing query answering service capability.
Ontology Based Data Access (OBDA)

- Idea: using ontologies as a conceptual view over data repositories.

What do we get?

- Ontologies provide a conceptualization of a domain interest.
- Unified view among various data sources.
- Imposes constraints on the data over the conceptual level.
- Helps to infer implicit information.
- Enhancing query answering service capability.
Outline

1. Marriage Between Knowledge and Data
   - Ontology Based Data Access (OBDA)

2. Data Meet Processes
   - Artifact-Centric Approach
   - Data-Centric Dynamic Systems (DCDS)

3. Embracing Knowledge into Data-aware Processes
   - Semantically-Governed Data-Aware Processes (SGDAP)
   - SGDAP Formal Definition
   - SGDAP Semantics
   - SGDAP in Action

4. What’s Next?
Data Meet Processes

Hi Process!!

Hi Data!!
Artifact-Centric Approach

- An approach for **business process modeling**.
- **Focuses** simultaneously on **data** and **processes**.

- Provides **richer** and **robust** model for business process development.
Artifact-Centric Approach (cont.)

• An Artifact System are characterized by:
  - Information model: Capture the artifact’s relevant information.
  - lifecycle: Characterize the evolution of the artifacts.

• Interesting task:
  - Design the system.
  - Understand temporal behavior (Verification of temporal properties).
    ✧ Verify if the evolution of the system satisfy the temporal properties.
Outline

1. Marriage Between Knowledge and Data
   - Ontology Based Data Access (OBDA)

2. Data Meet Processes
   - Artifact-Centric Approach
   - Data-Centric Dynamic Systems (DCDS)

3. Embracing Knowledge into Data-aware Processes
   - Semantically-Governed Data-Aware Processes (SGDAP)
   - SGDAP Formal Definition
   - SGDAP Semantics
   - SGDAP in Action

4. What’s Next?
Data-Centric Dynamic Systems (DCDS) [BCDDM-12]

- Abstract away an artifact system.
- Capture the essence of systems in which both data and process are first class citizens.

Data-Centric Dynamic Systems (DCDS) [BCDDM-12]

- Abstract away an artifact system.
- Capture the essence of systems in which both data and process are first class citizens.

Two components:
- Data Component: holds the relevant information to be manipulated by the system.
- Process Component: formed by the processes.
  - Consisting of invokable actions
  - Evolve the data and update the state of the system
  - Capture the interaction with external services to acquire new information from the environment (other systems, user choices, ...)

Formal Semantics of DCDS

- The semantics is defined in terms of transition system $\gamma$.
Formal Semantics of DCDS

- The semantics is defined in terms of transition system $\Gamma$.

- Intuitively:
  - The states in $\Gamma$ capture the snapshot of manipulated data.
  - The transitions in $\Gamma$ constitute the process executions, which evolve the system.

- Note:
  - $\Gamma$ is in general an infinite state space.
  - We only consider non-deterministic setting.
  - The same service call issued later may give a different result.
  - Useful to model user input and unpredictable external environment.
Formal Semantics of DCDS

- The semantics is defined in terms of transition system $\mathcal{\Sigma}$.

- Intuitively:
  - The states in $\mathcal{\Sigma}$ capture the snapshot of manipulated data.
  - The transitions in $\mathcal{\Sigma}$ constitute the process executions.
    - which evolve the system.

- Note:
  - $\mathcal{\Sigma}$ is in general infinite state.
  - We only consider non-deterministic setting.
    - The same service call issued later may give a different result.
    - Useful to model user input and unpredictable external environment.
Verification of Temporal Properties Over DCDSs

- Given a DCDS $S$ and a temporal logic formula $\Phi$
  - Construct the transition system $\gamma_S$ of $S$
  - Check whether $\gamma_S \models \Phi$

Verification of Temporal Properties Over DCDSs

- Given a DCDS $S$ and a temporal logic formula $\Phi$
  - Construct the transition system $\gamma_S$ of $S$
  - Check whether $\gamma_S \models \Phi$
- Challenge: $\gamma_S$ is in general infinite state

Verification of Temporal Properties Over DCDSs

- Given a DCDS $S$ and a temporal logic formula $\Phi$
  - Construct the transition system $\gamma_S$ of $S$
  - Check whether $\gamma_S \models \Phi$
- Challenge: $\gamma_S$ is in general infinite state

**Theorem (BCDDM-12)**

*There exists a DCDS $S$ with non-deterministic services, and a propositional LTL safety property $\Phi$, such that checking $\gamma_S \models \Phi$ is undecidable.*

---

One Decidability Result of Verification on DCDS

- Setting:
  - DCDS with non-deterministic service.
  - Temporal property is specified in a fragment of FO $\mu$-calculus (called $\mu L_P$).

$$\Phi ::= Q \mid \neg \Phi \mid \Phi_1 \land \Phi_2 \mid \exists x.\text{LIVE}(x) \land \Phi \mid$$
$$\langle \neg \rangle (\text{LIVE}(\bar{x}) \land \Phi) \mid [\neg ] (\text{LIVE}(\bar{x}) \land \Phi) \mid Z \mid \mu Z. \Phi$$

---

One Decidability Result of Verification on DCDS

- Setting:
  - DCDS with non-deterministic service.
  - Temporal property is specified in a fragment of FO $\mu$-calculus (called $\mu\mathcal{L}_P$).

$$\Phi ::= \begin{array}{c}
Q ~|~ \neg\Phi ~|~ \Phi_1 \land \Phi_2 ~|~ \exists x.\text{LIVE}(x) \land \Phi ~|\\
\langle -\rangle(\text{LIVE}(\bar{x}) \land \Phi) ~|~ \lceil -\rceil(\text{LIVE}(\bar{x}) \land \Phi) ~|~ Z ~|~ \mu Z.\Phi
\end{array}$$

- We need to restrict DCDSs to gain decidability:
  - state boundedness: there is a finite bound $b$ such that for each state $\mathcal{I}$ of $\mathcal{Y}_S$, $|\text{ADOM}(\mathcal{I})| < b$.

**Theorem (BCDDM-12)**

Verification of $\mu\mathcal{L}_P$ properties on state-bounded DCDSs with non-deterministic services is decidable.

DCDS example

- **Process:**
  \[ peer(x, y) \land Gold(y) \quad \rightarrow \quad \text{GetLoan}(x) \]

- **Service Call:**
  \[ UInput(x) \]

- **Actions:**
  \[ \text{GetLoan}(x) : \]
  \[
  \begin{align*}
  Customer(z) & \quad \leadsto \quad \{ Customer(z) \}, \\
  Peer(z_1, z_2) & \quad \leadsto \quad \{ Peer(z_1, z_2) \}, \\
  Owes(z_1, z_2) & \quad \leadsto \quad \{ Owes(z_1, z_2) \}, \\
  Gold(z) & \quad \leadsto \quad \{ Gold(z) \}, \\
  \exists y. Peer(x, y) & \quad \leadsto \quad \{ Owes(x, UInput(x)) \}
  \end{align*}
  \]
DCDS example

- **Process:**
  \( peer(x, y) \land Gold(y) \iff \text{GetLoan}(x) \)

- **Service Call:**
  \( UInput(x) \)

- **Actions:**
  \( \text{GetLoan}(x) : \)
  
  \[
  \begin{align*}
  \text{Customer}(z) & \rightsquigarrow \{\text{Customer}(z)\}, \\
  \text{Peer}(z_1, z_2) & \rightsquigarrow \{\text{Peer}(z_1, z_2)\}, \\
  \text{Owes}(z_1, z_2) & \rightsquigarrow \{\text{Owes}(z_1, z_2)\}, \\
  \text{Gold}(z) & \rightsquigarrow \{\text{Gold}(z)\}, \\
  \exists y. \text{Peer}(x, y) & \rightsquigarrow \{\text{Owes}(x, UInput(x))\}
  \end{align*}
  \]

- **Temporal property:**
  \[
  \mu Z. ([\exists xyz. Gold(x) \land Peer(y, x) \land Owes(y, z)] \lor \langle - \rangle Z) \\
  = F[\exists xyz. Gold(x) \land Peer(y, x) \land Owes(y, z)]
  \]

  Eventually there is someone who owes money from someone who is his peer.
Outline

1. Marriage Between Knowledge and Data
   - Ontology Based Data Access (OBDA)

2. Data Meet Processes
   - Artifact-Centric Approach
   - Data-Centric Dynamic Systems (DCDS)

3. Embracing Knowledge into Data-aware Processes
   - Semantically-Governed Data-Aware Processes (SGDAP)
     - SGDAP Formal Definition
     - SGDAP Semantics
     - SGDAP in Action

4. What’s Next?
Embracing Knowledge into Data-aware Processes

- We propose: *Semantically-Governed Data-Aware Processes (SGDAP).*
- Idea: OBDA + DCDS.
Semantically-Governed Data-Aware Processes (SGDAP)

The evolution of the system occurs at the relational layer.

- Processes are defined over the database schemas in the relational layer.
Semantically-Governed Data-Aware Processes (SGDAP)

The evolution of the system occurs at the relational layer.

- Processes are defined over the database schemas in the relational layer.

The semantic layer can be added on top of the relational layer to:

- Understand the system in terms of concepts and relationships relevant for the domain of interest.
  - Unified well-structured view of the whole information present in the system.
  - Interconnection of different databases that share information, though with different representation.
  - Understand how new component can be attach to the system.
  - Specification of queries and system constraint at the conceptual level.
Semantically-Governed Data-Aware Processes (SGDAP)

The evolution of the system occurs at the relational layer.

- Processes are defined over the database schemas in the relational layer.

The semantic layer can be added on top of the relational layer to:

- Understand the system in terms of concepts and relationships relevant for the domain of interest.
  - Unified well-structured view of the whole information present in the system.
  - Interconnection of different databases that share information, though with different representation.
  - Understand how new component can be attach to the system.
  - Specification of queries and system constraint at the conceptual level.

- Govern the system evolution:
  - Regulate the change of database and processes execution in the system
  - Ensure that the processes execution always manipulate data in accordance to the semantic layer.
Semantically-Governed Data-Aware Processes (SGDAP)

Semantic layer:
- Conceptual schema (Description Logic TBox) describing domain of interest as well as semantic constraints.
- **Data** are concretely maintained at the **relational layer**.
- **Semantic layer** defined over the **relational layer**.
- **We have layer separation.**
Semantically-Governed Data-Aware Processes (SGDAP)

- Each snapshot of database is conceptualized in the ontology, in terms of a virtual DL ABox.
- Mappings define how to obtain the virtual ABox from the concrete data sources.
Semantically-Governed Data-Aware Processes (SGDAP)

- The system evolves based on the actions execution in the relational layer.
- Semantic layer is used to understand the evolution at the conceptual level.
Semantic governance:

- Semantic layer used to **regulate the actions’ execution** at the relational layer.
- Actions execution leading to **violate** the semantic **constraints** are rejected.
Semantically-Governed Data-Aware Processes (SGDAP)

- External services invoked to feed the relational layer with new, fresh data.
Outline

1. Marriage Between Knowledge and Data
   - Ontology Based Data Access (OBDA)

2. Data Meet Processes
   - Artifact-Centric Approach
   - Data-Centric Dynamic Systems (DCDS)

3. Embracing Knowledge into Data-aware Processes
   - Semantically-Governed Data-Aware Processes (SGDAP)
   - SGDAP Formal Definition
   - SGDAP Semantics
   - SGDAP in Action

4. What’s Next?
SGDAP - Formal Definition

SGDAP $S = \langle O, P, D_0 \rangle$, where

- $O = \langle R, T, M \rangle$ is an OBDA system

- $D_0$ is an initial database conforming to $R$.
- $P = \langle F, A, \pi \rangle$ is a process component
SGDAP - Formal Definition

SGDAP \( S = \langle \mathcal{O}, \mathcal{P}, D_0 \rangle \), where

- \( \mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle \) is an OBDA system constituted by:
  - a database schema \( \mathcal{R} \) (i.e., a set of relation schemas);
  - a \( DL-Lite_A \) conceptual schema (TBox) \( \mathcal{T} \);
  - a set of mappings \( \mathcal{M} \), each relating a query over \( \mathcal{R} \) with a query over \( \mathcal{T} \).
- \( D_0 \) is an initial database conforming to \( \mathcal{R} \).
- \( \mathcal{P} = \langle \mathcal{F}, \mathcal{A}, \pi \rangle \) is a process component
SGDAP $\mathcal{S} = \langle \mathcal{O}, \mathcal{P}, D_0 \rangle$, where

- $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$ is an OBDA system constituted by:
  - a database schema $\mathcal{R}$ (i.e., a set of relation schemas);
  - a $DL$-Lite$_A$ conceptual schema (TBox) $\mathcal{T}$;
  - a set of mappings $\mathcal{M}$, each relating a query over $\mathcal{R}$ with a query over $\mathcal{T}$.
- $D_0$ is an initial database conforming to $\mathcal{R}$.
- $\mathcal{P} = \langle \mathcal{F}, \mathcal{A}, \pi \rangle$ is a process component constituted by
  - a set of functions $\mathcal{F}$, each representing an external service;
  - a set of actions $\mathcal{A}$, defined over $\mathcal{R}$ and possibly invoking external services;
  - a process $\pi$ specifying the control flow, i.e., when and how actions can be executed.
A set of relation schemas.
Database Schema

- A set of relation schemas.
- Example:
  - ENROLLED(id, name, surname, type, endDate)
    - (endDate = NULL) for current student
    - (type='Bachelor') for the bachelor student.
    - (type='Master') for the master student.
  - TRANSF_M(name, surname)
    - Abbreviation of TRANSFER_MASTER
    - Students recently transferred from a Master in another University.
    - They must still be associated to a matriculation number.
  - GRAD(id, mark, type)
    - Graduated students.
Conceptual Schema

- Specified using $DL$-$Lite_A$ (lightweight description logic suitable for data access and management).
Conceptual Schema

- Specified using $DL$-$Lite_A$ (lightweight description logic suitable for data access and management).
- Example:

```
Bachelor ⊆ Student
Master ⊆ Student
Graduated ⊆ Student
δ(MNum) ⊆ Student
Student ⊆ δ(MNum)
(funct MNum)
(id Student MNum)
```
• Progression mechanism for the system (at the relational layer).
  ▶ Finite set of condition-action rules

\[ Q \rightarrow \alpha \]

▶ Intuitively tells at any moment which actions can be executed.
Process

• Progression mechanism for the system (at the relational layer).
  ▶ Finite set of condition-action rules
    \[ Q \rightarrow \alpha \]
    ▶ Intuitively tells at any moment which actions can be executed.
• Example:
  ▶ Matriculation of a transferred student.
    \( \text{TRANSF}_M(\text{name, surname}) \rightarrow \text{COMPL-ENR(name, surname)} \)
  ▶ Graduation of a student.
    \( \text{ENROLLED(id,_,_,_, NULL)} \rightarrow \text{GRADUATE(id)} \)
Actions and Services

- An action manipulates data.
- Possibly importing fresh values by calling external services.
Actions and Services

• An action manipulates data.
• Possibly importing fresh values by calling external services.
• It is of the form

\[ \alpha(p_1, \ldots, p_n) : \{ e_1, \ldots, e_m \} \]

▶ an action signature (name + set of parameters);
▶ a set of effect specifications taking place simultaneously, each of the form

\[ q^+ \land Q^- \rightsquigarrow E \]

where

★ \( q^+ \) is a UCQ over the database schema \( \mathcal{R} \);
★ \( Q^- \) is a FO query that acts as a filter
  (its free variables are included in those of \( q^+ \)).
★ \( E \) is a set of facts over \( \mathcal{R} \).
Actions and Services - Example

Service calls:

- `today();`
- `getID(student_name, student_surname, type);`
- `getMark(student_id, degree_type).`
Actions and Services - Example

Service calls:

- \textit{today}();

- \texttt{getID(student\_name, student\_surname, type)};

- \texttt{getMark(student\_id, degree\_type)}.

\textsc{compl-enr}(n, s):

- \texttt{GRAD(id, m, t)} \mapsto \texttt{GRAD(id, m, t)}

- \texttt{ENROLLED(id, n_2, s_2, t, d)} \mapsto \texttt{ENROLLED(id, n_2, s_2, t, d)}

- \texttt{TRANSF\_M(n_2, s_2) \land (n_2 \neq n \lor s_2 \neq s)} \mapsto \texttt{TRANSF\_M(n_2, s_2)}

- \texttt{TRANSF\_M(n, s)} \mapsto \texttt{ENROLLED(getID(n, s, 'Master'), n, s, 'Master', NULL)}
Actions and Services - Example

Service calls:

- `today()`;
- `getID(student_name, student_surname, type)`;
- `getMark(student_id, degree_type)`.

COMPL-ENR\((n, s)\):

- `GRAD(id, m, t) \leadsto GRAD(id, m, t)`
- `ENROLLED(id, n_2, s_2, t, d) \leadsto ENROLLED(id, n_2, s_2, t, d)`
- `TRANSF_M(n_2, s_2) \land (n_2 \neq n \lor s_2 \neq s) \leadsto TRANSF_M(n_2, s_2)`
- `TRANSF_M(n, s) \leadsto ENROLLED(getID(n, s, 'Master'), n, s, 'Master', NULL)`

GRADUATE\((id)\):

- `GRAD(id_2, m, t) \leadsto GRAD(id_2, m, t)`
- `TRANSF_M(n, s) \leadsto TRANSF_M(n, s)`
- `ENROLLED(id_2, n, s, t, d) \land id_2 \neq id \leadsto ENROLLED(id_2, n, s, t, d)`
- `ENROLLED(id, n, s, t, NULL) \leadsto ENROLLED(id, n, s, t, today())`
- `ENROLLED(id, _, _, t, NULL) \leadsto GRADUATED(id, getMark(id, t), t)`
m1: SELECT name, surname, type FROM ENROLLED WHERE type = 'Bachelor'
  \[\rightarrow\text{Bachelor}(s_1(name, surname, type))\]

m2: SELECT name, surname, type FROM ENROLLED WHERE type = 'Master'
  \[\rightarrow\text{Master}(s_1(name, surname, type))\]

m3: SELECT name, surname, type, id FROM ENROLLED
    \[\rightarrow\text{MNum}(s_1(name, surname, type), val(id))\]

m4: SELECT name, surname FROM TRANSF_M
    \[\rightarrow\text{Master}(s_1(name, surname, 'Master'))\]

m5: SELECT e.name, e.surname, e.type FROM ENROLLED e, GRAD g WHERE e.id = g.id
    \[\rightarrow\text{Graduated}(s_1(name, surname, type))\]
Outline

1. Marriage Between Knowledge and Data
   - Ontology Based Data Access (OBDA)

2. Data Meet Processes
   - Artifact-Centric Approach
   - Data-Centric Dynamic Systems (DCDS)

3. Embracing Knowledge into Data-aware Processes
   - Semantically-Governed Data-Aware Processes (SGDAP)
   - SGDAP Formal Definition
   - SGDAP Semantics
   - SGDAP in Action

4. What’s Next?
SGDAP Semantic via Transition System

- **Relational transition system** $\gamma^R_S$.
  - Whose states are pair (state id, DB).
  - $\gamma^R_S$ is semantically governed by $\mathcal{T}$.

- **Semantic transition system** $\gamma^S_S$.
  - Whose states are pair (state id, ABox).
  - $\gamma^S_S$ just mirrors the structure of $\gamma^R_S$, applying the mappings state by state.
Relational Transition System Construction

**Example:** application of COMPL-ENR(A,B)

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>surname</th>
<th>type</th>
<th>endDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>D</td>
<td>Master</td>
<td>NULL</td>
</tr>
<tr>
<td>0</td>
<td>A</td>
<td>B</td>
<td>Master</td>
<td>NULL</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>surname</th>
<th>type</th>
<th>endDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>D</td>
<td>Master</td>
<td>NULL</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>B</td>
<td>Master</td>
<td>NULL</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>surname</th>
<th>type</th>
<th>endDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>D</td>
<td>Master</td>
<td>NULL</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>B</td>
<td>Master</td>
<td>NULL</td>
</tr>
</tbody>
</table>
```

...
Semantic Transition System Construction

Example: application of COMPL-ENR(A,B)

\[
\begin{align*}
\text{TRANSF}_M & \quad \text{GRAD} \\
\text{ENROLLED} & \quad \text{ENROLLED}
\end{align*}
\]
Outline

1. Marriage Between Knowledge and Data
   - Ontology Based Data Access (OBDA)

2. Data Meet Processes
   - Artifact-Centric Approach
   - Data-Centric Dynamic Systems (DCDS)

3. Embracing Knowledge into Data-aware Processes
   - Semantically-Governed Data-Aware Processes (SGDAP)
   - SGDAP Formal Definition
   - SGDAP Semantics
   - SGDAP in Action

4. What’s Next?
Dynamic Properties/Constraints

- Specified using temporal logics formulas whose atoms are epistemic queries over the ontology.
  - Formalism for temporal properties: $\mu L$.
  - Epistemic queries according to EQL-Lite.
- We investigate $\mu L_C$ with closed EQL-Lite queries.
Dynamic Properties/Constraints

- Specified using temporal logics formulas whose atoms are epistemic queries over the ontology.
  - Formalism for temporal properties: $\mu L$.
  - Epistemic queries according to EQL-Lite.
- We investigate $\mu L_C$ with closed EQL-Lite queries.
- Example in $\mu L_C$:

$$\mu Z. (\forall s. \text{Student}(s) \rightarrow \text{Grad}(s)) \lor [\neg]Z$$

Every execution leads to a state in which all the students present in that state are graduated.
From the Semantic Back to the Relational Layer

Temporal property mixes:
- temporal operators to talk about the dynamics of the system;
- queries over the ontology (TBox + virtual ABoxes).

But the system evolves at the relational layer.

How to reconcile all these pieces?
From the Semantic Back to the Relational Layer

Temporal property mixes:

- temporal operators to talk about the dynamics of the system;
- queries over the ontology (TBox + virtual ABoxes).

But the system evolves at the relational layer.

How to reconcile all these pieces?

- The temporal part is maintained unaltered, because the system evolves at the relational layer.
- Faithful transformation of the queries:
  1. **Rewriting** to compile away the TBox;
  2. **Unfolding** to obtain a corresponding property over the relational schema \( \mathcal{R} \).
Verification and Abstraction

Proved for $\mu\mathcal{L}_C$.

\[
\begin{align*}
\forall S & \forall A_0 D_0 S_0 A_0 \\
\forall S & \forall A_1 D_1 S_1 A_1 \\
\forall S & \forall A_2 D_2 S_2 A_2 \\
\forall S & \forall A_3 D_3 S_3 A_3 \\
\Phi &= \Phi' \\
\Phi' &= \text{UNFOLD}(\text{REW}(\Phi, T), M)
\end{align*}
\]
Semantically-Governed Data-Aware Processes


Outline

1. Marriage Between Knowledge and Data
   - Ontology Based Data Access (OBDA)

2. Data Meet Processes
   - Artifact-Centric Approach
   - Data-Centric Dynamic Systems (DCDS)

3. Embracing Knowledge into Data-aware Processes
   - Semantically-Governed Data-Aware Processes (SGDAP)
   - SGDAP Formal Definition
   - SGDAP Semantics
   - SGDAP in Action

4. What’s Next?
What’s next?

- Various setting/situations of knowledge, data and process integration:
  - Process are specified in the high level over the semantic layer.
    - Leads to process synthesis problem.
  - Process are specified partially.
    - How to synthesize additional processes while satisfying the given temporal property.
What’s next? (cont.)

- Accommodating update on semantic layer.
  - There is a change explicitly over the ontology.
  - How changes done at the semantic layer affect the relational layer.
  - Related to view updates problem.
  - Plan: investigate bidirectional mapping [Pierce12] to formalize the relation between the two layers.

---

What’s next? (cont.)

• Quantitative property specification.
  ▶ Verification and synthesis in presence of quantitative requirements.
  ▶ Close to [CS12].

• Considering classes of Petri Nets equipped with data for data-aware process formalism.
  ▶ e.g., Colored Petri Nets or Petri Data Nets.

• Deal with complexity issues.

• Come up with the real use cases.

Thank you

شكرا  хвала  謝謝  Merci  спасибо  Dank u wel  teşekkür ederim  ขอบคุณคุณ  شكرا  köszönöm  děkuji  감사합니다  mulţumesc