Complex Event Recognition in Logic and AI

Diego Calvanese

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

Department of Computing Science
Umeå University, Sweden

Dagstuhl Seminar 20071 on
Complex Event Recognition (CER)

10–14 February 2020 – Dagstuhl, Germany

Outline

Disclaimer (shamelessly copied from Cristian and Martin)

This tutorial is NOT

an overview of existing approaches, logics, formalisms

about how to write "things" in 50 different languages

about architectures/technologies

exhaustive in any way

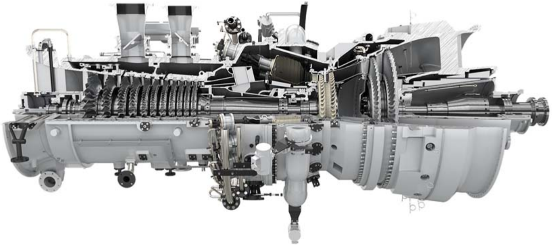
. . . probably not even fun

and definitely with too many slides!

Outline

Siemens Energy Services

- Monitor gas and steam turbines.
- Collect data from 50 remote diagnostic centers around the world.
- Centers linked to a common central DB.
- Turbines are highly complex, with 5 000–50 000 sensors each.
- Engineers compute KPIs, and extract data from maintenance reports using ETL tools.



Objective: retrospective diagnostics

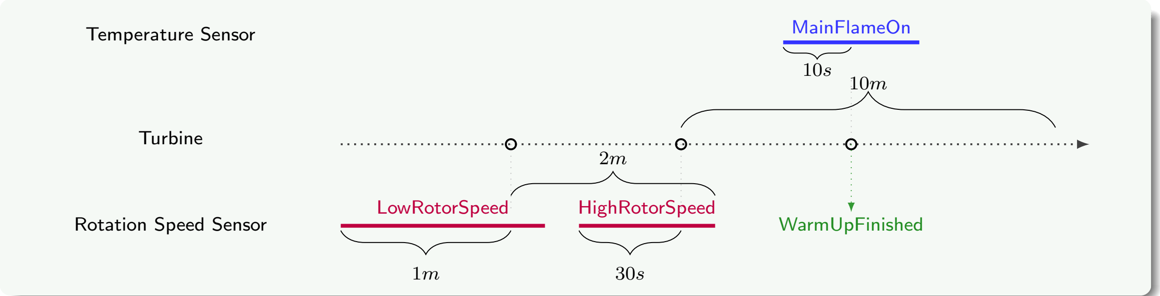
i.e., recognize complex events that are abnormal or potentially dangerous.

Events

- Involve a number of sensor measurements.
- Have a certain temporal duration.
- Occur in a certain temporal sequence.

Recognizing complex patterns

Find the **time periods of "Warm up"**
of the gas turbines deployed in the train with ID T001.



We need to **model** the dynamic aspects, i.e., **the process**:

- relations between temporal events
- metric constraints (e.g., at least 1 min)

Modeling static knowledge about the domain

We need also to capture the **static knowledge** about

- machines and their deployment profiles
- component hierarchies
- sensor configurations
- functional profiles

Devices consist of parts, which are monitored by different kinds of sensors (rotation, pressure, ...).

- A gas turbine is a turbine.
- A steam turbine is a turbine.
- A power unit is a turbine-part.
- A burner is a turbine-part.
- A temperature sensor is a sensor.
- A rotation-speed sensor is a sensor.
- ...
- Turbines are deployed in a train.
- A turbine-part is part-of a turbine.
- A turbine-part is monitored by a sensor.
- ...

Dealing with the data

On the **data** side

- no fixed periodicity,
- different periodicities for different data sources,
- incomplete data (NULLs in the tables).

ts	*value*
12:20:00	570
12:20:00.001	570.5
12:00:00.002	570.7
...	...
12:20:05	575
12:20:05.001	575.12
12:00:05.002	575.34
...	...
12:20:08.023	589.16
12:00:08.024	589.32
...	...

deadbanded →

tb_measurement

ts	*value*
12:20:00	570
12:20:05	575
12:20:08.023	589.16
...	...

We also need to "**connect**" the data to the static and dynamic models:

- deal with structural information, fully taking into account the semantics of the data as conveyed by its model;
- deal with temporal information (e.g., through timestamps), again taking into account the temporal/process model.

Dealing with the data

On the **data** side

- no fixed periodicity,
- different periodicities for different data sources,
- incomplete data (NULLs in the tables).

ABMN6

ts	*temp*
01:25:00	12.78
02:25:00	12.78
03:25:00	12.22
04:25:00	NULL
05:25:00	10.56
...	...

periodicity = 1hour

AP064

ts	*temp*
00:00:00	10.01
00:15:00	10.01
00:30:00	NULL
00:45:00	8.9
01:00:00	8.7
...	...

periodicity = 15mins

We also need to "**connect**" the data to the static and dynamic models:

- deal with structural information, fully taking into account the semantics of the data as conveyed by its model;
- deal with temporal information (e.g., through timestamps), again taking into account the temporal/process model.

The challenge for Complex Event Recognition

CER requires to deal together with

temporal information / processes

data

knowledge

We are interested in **modeling** these three components, and **inferring** relevant properties about them.

Outline

Processes and data

They constitute the information assets of an organization:

- **data**: determine the information of interest
- **processes**: determine how data change and evolve over time

Conceptual Modeling

Both aspects can be modelled at the conceptual level, but traditionally this is done:

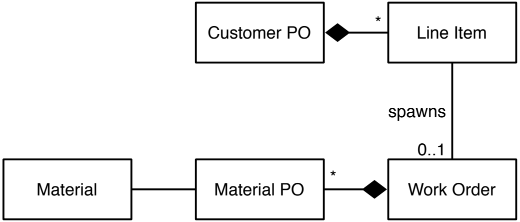
- using different modeling tools,
- by different teams with different competences, and
- their connection is NOT modelled conceptually, but it should!

Consequence

Automated inference (e.g., for verification) combining both processes and data, is not possible!

Conventional data modeling

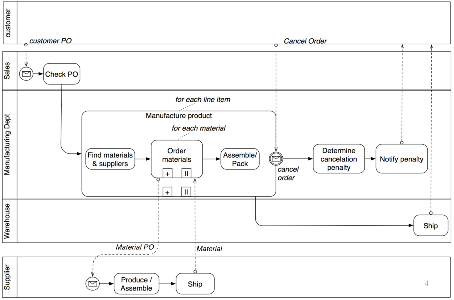
- Produce a structural model of the domain of interest
- Focus: entities, relations, and static constraints that are relevant for the domain of interest.
- Formalisms: UML, ER, ORM, ...
- Result: conceptual model of a database schema



But how do data evolve?

Conventional process modeling

- Produce a model of the dynamics of the domain of interest
- Focus: control flow of activities realizing the business objectives
- Formalisms: BPMN, UML AD, ...
- Result: executable process model



But how are data manipulated?

Consequences of the dichotomy

Survey by Forrester: 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.

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!

Overcoming the dichotomy

Strong need for:

- Suitable modeling formalisms supporting the **integrated management** of processes and data **at the conceptual level**.
- A clear understanding of semantic and **computational properties** of such formalisms, so as to enable their **analysis**.

Let us adopt the standard formalism for capturing at the conceptual level the structural aspects of the domain of interest, namely **Description Logics**.

Outline

Description Logics

- **Description Logics (DLs)** stem from early days (1970') KR formalisms, and assumed their current form in the late 1980's & 1990's.
- Are logics specifically designed to represent and reason on **structured knowledge**.
- Technically they can be considered as well-behaved (i.e., decidable) **fragments of first-order logic**.
- Semantics given in terms of first-order interpretations.
- Come in hundreds of variations, with different semantic and computational properties.
- Provide the formal foundations for the W3C standard **Web Ontology Language** OWL.

Representing knowledge in Description Logics

The domain of interest is composed of objects and is structured into:

- **concepts**, which correspond to classes, and denote sets of objects
- **roles**, which correspond to (binary) relationships, and denote binary relations on objects.

DL language

Each DL is characterized by a **description language**, specifying the constructs to form complex concept and role expressions, starting from concept and role names.

Description language: Concept and role constructs

Construct	Syntax	Example	Semantics
atomic concept	A	Turbine	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
atomic role	P	isMonitoredBy	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
atomic negation	$\neg A$	¬SpeedSensor	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
conjunction	$C \sqcap D$	Sensor ⊓ TurnedOn	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
(unqual.) existential res.	$\exists R$	∃isMonitoredBy	$\{o \mid \exists o'. (o,o') \in R^{\mathcal{I}}\}$
value restriction	$\forall R.C$	∀turbineCode.Integer	$\{o \mid \forall o'. (o,o') \in R^{\mathcal{I}} \to o' \in C^{\mathcal{I}}\}$
bottom	$\bot$		$\emptyset$

(C, D denote arbitrary concepts and R an arbitrary role)

The above constructs form the basic language $\mathcal{AL}$ of the family of $\mathcal{AL}$ languages.

Additional concept and role constructs

Construct	$\mathcal{AL}\cdot$	Syntax	Semantics
disjunction	$\mathcal{U}$	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$
top		$\top$	$\Delta^\mathcal{I}$
qualified existential restriction	$\mathcal{E}$	$\exists R.C$	$\{ o \mid \exists o'. (o,o') \in R^\mathcal{I} \land o' \in C^\mathcal{I} \}$
(full) negation	$\mathcal{C}$	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
number restrictions	$\mathcal{N}$	$\geq k\, R$	$\{ o \mid \#\{o' \mid (o,o') \in R^\mathcal{I}\} \geq k \}$
		$\leq k\, R$	$\{ o \mid \#\{o' \mid (o,o') \in R^\mathcal{I}\} \leq k \}$
qualified number restrictions	$\mathcal{Q}$	$\geq k\, R.C$	$\{ o \mid \#\{o' \mid (o,o') \in R^\mathcal{I} \land o' \in C^\mathcal{I}\} \geq k \}$
		$\leq k\, R.C$	$\{ o \mid \#\{o' \mid (o,o') \in R^\mathcal{I} \land o' \in C^\mathcal{I}\} \leq k \}$
nominal	$\mathcal{O}$	$\{c\}$	$\{c^\mathcal{I}\}$
inverse role	$\mathcal{I}$	R^-	$\{ (o,o') \mid (o',o) \in R^\mathcal{I} \}$

Many different DL constructs and their combinations have been investigated.

Further examples of DL constructs

- Disjunction: $\mathsf{TempSensor} \sqcup \mathsf{SpeedSensor}$
- Qualified existential restriction: $\exists \mathsf{isMonitoredBy}.\mathsf{Sensor}$
- Full negation: $\neg(\mathsf{TempSensor} \sqcup \mathsf{SpeedSensor})$
- Number restrictions: $\geq 3\, \mathsf{isMonitoredBy} \sqcap \leq 6\, \mathsf{isMonitoredBy}$
- Qualified number restrictions: $\geq 2\, \mathsf{isMonitoredBy}.\mathsf{TempSensor}$
- Nominal: $\exists \mathsf{isMonitoredBy}.\{\mathtt{mf05}\}$
- Inverse role: $\forall \mathsf{isMonitoredBy}^{-}.\mathsf{TurbinePart}$

Asserting knowledge in Description Logics

Intensional knowledge about the domain is specified through **inclusion axioms**:

TBox (for "terminological box")

Concept inclusion:	$C_1 \sqsubseteq C_2$	$\forall x. C_1(x) \rightarrow C_2(x)$
Role inclusion:	$R_1 \sqsubseteq R_2$	$\forall x, y. R_1(x, y) \rightarrow R_2(x, y)$

Extensional knowledge is specified through **membership assertions**, denoting facts involving individuals:

ABox (for "assertional box")

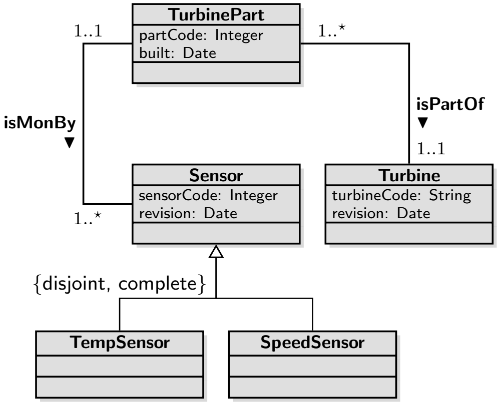
Concept membership assertion:	$A(c)$	TurbinePart(`b01`), TempSensor(`mf05`)
Role membership assertion:	$P(c_1, c_2)$	isMonitoredBy(`b01`, `mf05`)

Description Logic **knowledge base**

Constituted by a TBox and ABox together.

DLs capture UML class diagrams

There is a close correspondence between DLs and conceptual modeling formalisms [Lenzerini and Nobili 1990; Bergamaschi and Sartori 1992; Borgida 1995; C., Lenzerini, et al. 1999; Borgida and Brachman 2003; Berardi et al. 2005; Queralt et al. 2012].



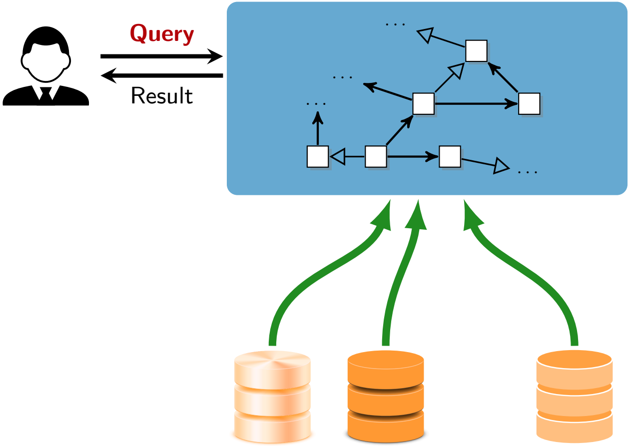
$$
\begin{array}{rcl}
\mathsf{TempSensor} & \sqsubseteq & \mathsf{Sensor}\\
\mathsf{SpeedSensor} & \sqsubseteq & \mathsf{Sensor}\\
\mathsf{TempSensor} & \sqsubseteq & \neg\mathsf{SpeedSensor}\\
\mathsf{Sensor} & \sqsubseteq & \mathsf{TempSensor} \sqcup \mathsf{SpeedSensor}\\
\mathsf{Turbine} & \sqsubseteq & \forall \mathsf{turbineCode}.\mathsf{String} \sqcap\\
 & & \exists \mathsf{turbineCode} \sqcap \leq 1\, \mathsf{turbineCode}\\
\exists \mathsf{isMonBy} & \sqsubseteq & \mathsf{TurbinePart}\\
\exists \mathsf{isMonBy}^- & \sqsubseteq & \mathsf{Sensor}\\
\mathsf{TurbinePart} & \sqsubseteq & \exists \mathsf{isMonBy}\\
\mathsf{Sensor} & \sqsubseteq & \exists \mathsf{isMonBy}^-\\
\exists \mathsf{isPartOf} & \sqsubseteq & \mathsf{TurbinePart}\\
\exists \mathsf{isPartOf}^- & \sqsubseteq & \mathsf{Turbine}
\end{array}
$$

$$(\textbf{funct}\ \mathsf{isMonBy}^-)$$
$$(\textbf{funct}\ \mathsf{isPartOf})$$
$$\ldots$$

DLs capture conceptual modeling formalisms

Modeling construct	DL axiom	FOL formalization
ISA on classes	$A_1 \sqsubseteq A_2$	$\forall x(A_1(x) \rightarrow A_2(x))$
... and on relations	$R_1 \sqsubseteq R_2$	$\forall x, y(R_1(x,y) \rightarrow R_2(x,y))$
Disjointness of classes	$A_1 \sqsubseteq \neg A_2$	$\forall x(A_1(x) \rightarrow \neg A_2(x))$
... and of relations	$R_1 \sqsubseteq \neg R_2$	$\forall x, y(R_1(x,y) \rightarrow \neg R_2(x,y))$
Domain of relations	$\exists P \sqsubseteq A_1$	$\forall x(\exists y(P(x,y)) \rightarrow A_1(x))$
Range of relations	$\exists P^- \sqsubseteq A_2$	$\forall x(\exists y(P(y,x)) \rightarrow A_2(x))$
Mandatory participation *(min card = 1)*	$A_1 \sqsubseteq \exists P$ $A_2 \sqsubseteq \exists P^-$	$\forall x(A_1(x) \rightarrow \exists y(P(x,y)))$ $\forall x(A_2(x) \rightarrow \exists y(P(y,x)))$
Functionality of relations *(max card = 1)*	$A_1 \sqsubseteq \leq 1\, P$ $A_2 \sqsubseteq \leq 1\, P^-$	$\forall x, y, y'(A_1(x) \wedge P(x,y) \wedge P(x,y') \rightarrow y = y')$ $\forall x, x', y(A_2(y) \wedge P(x,y) \wedge P(x',y) \rightarrow x = x')$
...	...	...

Virtual Knowledge Graphs – DL ontologies for data access (OBDA)



Ontology $\mathcal{O}$
conceptual view of data, convenient vocabulary

Mapping $\mathcal{M}$
how to populate the ontology from the data

Data Sources $\mathcal{S}$
autonomous and heterogeneous

Simplifies the access to information, and allows one to abstract away the precise structure of data sources.

Outline

DLs and processes

Description logics have been one of the cornerstones of the Semantic Web

- Formalisms of choice for modeling ontologies, i.e., conceptualizations of the domain of interest.
- Represent statics aspects of knowledge: classes, relationships between classes, ISA, ...

Semantic web services are the other cornerstone of the Semantic Web

- High-level descriptions of computations abstracting from the technological issues of the actual programs that realize them.
- Deal with dynamic aspects: action effects, change, knowledge evolution, ...

Combining data and processes

- Has been considered critical for the Semantic Web enterprise.
- Many attempts in mid 2000: OWL-S, SWSO, WSMO, ...
- But resisted automated reasoning ... for good reasons.

(A partial) history of DLs and processes

- Early 1990s: Starting point
  - [Baader 1991]: Extends $\mathcal{ALC}$ with regular expressions on roles ($\mathcal{ALC}_{reg}$).
  - [Schild 1991]: DLs $\leftrightarrow$ PDLs/Modal Logic: $\mathcal{ALC}_{reg} =$ PDL.
- Mid 1990s: High hopes
  - [Schild 1993]: DLs + temporal logics (for processes – point based).
  - [De Giacomo and Lenzerini 1994a; Schild 1994]: DLs $\leftrightarrow$ μ-calculus.
  - [De Giacomo and Lenzerini 1994a,b, 1995a, 1996; C., De Giacomo, and Lenzerini 1995]: DLs as rich modal logics, ExpTime-complete.
  - [De Giacomo and Lenzerini 1995b]: Use knowledge in DLs to capture Reiter's Propositional Situation Calculus.
- End of 1990: Everything collapses
  - [Baader and Laux 1995]: DLs+modal logics = Multi-Dimensional Modal Log.
  - [Wolter and Zakharyaschev 1998, 1999a,b,c; Gabbay et al. 2003]: Multi-Dimensional Modal Logics are computationally nasty.

Satisfiability of a KB where a role extension persists is undecidable!

Multi-dimensional (Description) Logics

Example: We want to express that a potential customer (for a car salesman) is an adult who eventually wants to own a car:

$$\text{PotentialCustomer} \equiv \text{Adult} \sqcap \exists \text{eventually-wants-own}.\text{Car}$$

- The role eventually-wants-own is a new role different from own and wants-own.
- But then there is no interaction between these roles, which intuitively misses something (e.g., wants-own should imply eventually-wants-own).

Proposal by [Baader and Ohlbach 1995]:

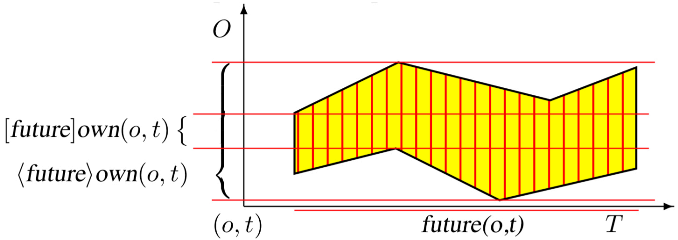
- Use modal operators with an appropriate modal theory of time (or other modalities):

$$\text{PotentialCustomer} \equiv \text{Adult} \sqcap \exists (\langle \text{future} \rangle [\text{wants}] \text{own}).\text{Car}$$

- Note that ordinary DL roles are just one of the modalities, namely the one operating on objects, while others operate on time-points, intentional worlds, etc.
- In fact, each modal operator is equipped with a dimension $\rightsquigarrow$ **Multi-dimensional** logic $\mathcal{M}\text{-}\mathcal{ALC}$.

Semantics of $\mathcal{M}$-$\mathcal{ALC}$

- Similar to the Kripke style possible worlds semantics for many-dimensional modal logic.
- One carrier set $\Delta_i^\mathcal{I}$ for each dimension i. The domain is $\Delta_1^\mathcal{I} \times \cdots \times \Delta_n^\mathcal{I}$.
- Concepts are subsets of the domain, i.e., n-tuples of elements.
- A role of dimension i is a function $\Delta_1^\mathcal{I} \times \cdots \times \Delta_n^\mathcal{I} \to 2^{\Delta_i^\mathcal{I}}$.
- Semantics of complex concept and role expressions:
  - concepts: standard DL semantics (where $\langle p \rangle C$ is as $\exists p.C$, and $[p]C$ as $\forall p.C$).
  - complex role terms:



Reasoning in $\mathcal{M}$-$\mathcal{ALC}$ [Baader and Ohlbach 1995]

$\mathcal{M}$-$\mathcal{ALC}$ is a rather complex logic!

Theorem

Concept satisfiability in $\mathcal{M}$-$\mathcal{ALC}$ is decidable, under the restriction that no $[\cdots]$-operator occurs in role terms.

Note: The algorithm is based on constraint propagation (i.e., it is a tableaux algorithm).

Independence of some dimensions:

- Consider a role future with dimension time, and an object dimension.
- A interpretation $\mathcal{I}$ may interpret future as an arbitrary function $\text{future}^{\mathcal{I}} : \Delta^{\mathcal{I}}_{\text{object}} \times \Delta^{\mathcal{I}}_{\text{time}} \to 2^{\Delta^{\mathcal{I}}_{\text{time}}}$
- Hence, for john and mary in $\Delta^{\mathcal{I}}_{\text{object}}$, and t_0 in $\Delta^{\mathcal{I}}_{\text{time}}$, the future time points reached by (john, t_0) may be different from those reached by (mary, t_0).
- If the role future is independent of the object dimension, this may not happen.

Theorem

Concept satisfiability is **undecidable** in $\mathcal{M}$-$\mathcal{ALC}$ extended with the possibility of declaring some roles to be independent of some dimensions.

Situation Calculus and DLs

To see which is the source of undecidability when enriching DLs with a temporal dimention, and how profound this is, let's combine DL KBs and Situation Calculus action theories into a single logical theory.

- SitCalc is possibly the best known formalism for Reasoning about Actions [McCarthy 1963], [McCarthy and Hayes 1969; Reiter 2001].
- DL KB describing an ontology can be seen as FOL "static constraints" in SitCalc.
- We use the single theory obtained by combining the DL KB and the SitCalc action theory to represent and reason on actions over the ontology.

Situation Calculus

SitCalc (Reiter's Basic Action Theories) [Reiter 2001]

- First-Order multi sorted language – but over inductively defined situations (i.e., situations are defined in **Second-Order** Logic). Sorts:
  - Objects: represent data.
  - Situations: denote the current state and the history that leads to that state.
  - Actions: progress the system – finite action types, but with infinitely many possible object parameters
  - Fluents: assert properties of objects in situations.
- Precondition axioms: define when (in which situations) actions (with parameters) can be executed

$$\mathsf{Poss}(a(\vec{x}), s) \equiv \Phi(\vec{x}, s)$$

- Successor state axioms: define the effects of action execution – include solution to the frame problem

$$F(\vec{x}, \mathsf{do}(a, s)) \equiv \Phi_F^+(\vec{x}, a, s) \;\vee\; (F(\vec{x}, s) \wedge \neg\Phi_F^-(\vec{x}, a, s))$$

- Initial situation S_0: description of the initial state of the system

SitCalc: Reasoning

Key feature: Regression

Regression allows for reducing reasoning about a given future situation to reasoning about the initial situation — Greatly simplifies reasoning as progression/executability.

However, more sophisticated temporal properties are also of interest:

- There exists a future situation such that α
- For all (future) situations α holds
- Eventually whatever actions we do we have α
- Always when α then eventually β
- ...

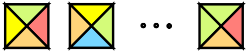
When we deal with such properties virtually all decidability results are based on assuming a **finite number of states**:

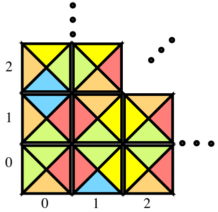
- Propositional SitCalc
- Assume finite object domain

$\mathcal{ALC}$ KBs + SitCalc BAT

However, the **combination of DLs and SitCalc is problematic!** [C., De Giacomo, and Soutchanski 2015]

We use a reduction from quadrant tiling problem.

- Given a finite set of tile types: 
- Can one tile the first quadrant respecting adjacency conditions?



Quadrant tiling problem

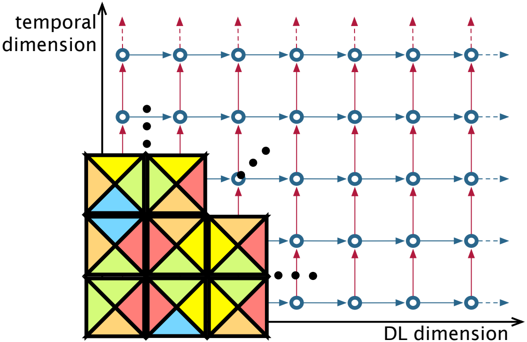
- Given
  - finite set of tile (or domino) types: $D_1, \ldots, D_m$
  - horizontal adjacency relation: H
  - vertical adjacency relation: V
- A tiling is a total function $T : \mathbb{N} \times \mathbb{N} \longrightarrow \{D_1, \ldots, D_m\}$
- A tiling is correct if $(T(i,j), T(i+1,j)) \in H$
  $(T(i,j), T(i,j+1)) \in V$

Undecidability of $\mathcal{ALC}$ + SitCalc

Theorem

Satisfiability of $\mathcal{ALC}$ + SitCalc is undecidable.

Proof: Reduction from quadrant tiling problem



Undecidability of $\mathcal{ALC}$ + SitCalc

Theorem

Satisfiability of $\mathcal{ALC}$ + SitCalc is undecidable.

Proof: Reduction from quadrant tiling problem

- Origin of the grid: $\mathsf{G}(0, S_0)$ *(or* $\{\mathsf{G}(0)\}$*)*
- Grid propagation: $\begin{cases} \mathsf{G} \sqsubseteq \exists \mathsf{right} \sqcap \forall \mathsf{right}.\mathsf{G} \\ \mathsf{G}(x, \mathsf{do}(\mathsf{up}, s)) \equiv \mathsf{G}(x, s) \\ \mathsf{right}(x, y, \mathsf{do}(\mathsf{up}, s)) \equiv \mathsf{right}(x, y, s) \end{cases}$
- Tiling of the whole grid: $\begin{cases} \mathsf{D}_\alpha \sqsubseteq \neg \mathsf{D}_\beta, & \text{for each } \alpha, \beta \in \{1, \ldots, m\},\ \alpha \neq \beta \\ \mathsf{G} \sqsubseteq \bigsqcup_{\alpha \in \{1,\ldots,m\}} \mathsf{D}_\alpha \end{cases}$
- For horizontal adjacency relation: $\mathsf{D}_\alpha \sqsubseteq \bigsqcup_{\beta:(D_\alpha, D_\beta) \in H} \forall \mathsf{right}.\mathsf{D}_\beta$
- For vertical adjacency relation: $\begin{cases} \mathsf{PD}_\beta(x, \mathsf{do}(\mathsf{up}, s)) \equiv \bigvee_{\alpha:(D_\alpha, D_\beta) \in V} \mathsf{D}_\alpha(x, s) \\ \mathsf{D}_\beta \sqsubseteq \mathsf{PD}_\beta \end{cases}$

Deep undecidability of DLs + Actions Theory

DLs + SitCalc undecidability result can be strengthened to:

- The simplest kind of DLs:
  - $\textit{DL-Lite}_{core}$ (i.e., OWL 2 QL profile of OWL 2)
  - $\mathcal{EL}$ (i.e., OWL 2 EL profile of OWL 2)
- The simplest kind of SitCalc Basic Action Theories, those satisfying these 2 properties:
  - "Local Effect", where successor state axioms change only the properties of objects mentioned explicitly in the action parameters"
  - "Context Free", where successor state axioms changes do not depend on properties holding in the current situation

Theorem

Satisfiability of $\textit{DL-Lite}_{core}/\mathcal{EL}$ KBs + SitCalc "Local Effect" and "Context Free" BATs is undecidable.

Deep undecidability of DLs + Actions Theory

Rationale

DLs + Action theories are undecidable even in the simplest cases.

How to regain decidability?

- Allow changes only of concepts (not roles)
  e.g., [Gabbay et al. 2003; Artale and Franconi 1999; Gutiérrez-Basulto et al. 2012; Jamroga 2012]
- Drop TBox (or make it acyclic)
  e.g., [Baader, Lutz, et al. 2005; Gu and Soutchanski 2007]
- Drop persistence of TBox (ontology is not maintained by actions)
  e.g., [Gu and Soutchanski 2010]

All these restrictions are unsuitable for conceptual models of data + processes!

Are there other options?

YES: as we will see next

Outline

Integrating DLs with action formalisms

Proposal by [Milicic 2008] (and different co-authors).

- Approach is not restricted to a particular DL.
- Hence, they study a very expressive DL: $\mathcal{ALCQIO}$
  - TBox is a set of concept definitions (i.e., $A \equiv C$), assumed to be acyclic.
  - ABox consists of positive and negative instance assertions on (possibly complex) concepts and roles.
- An atomic **action** has the form $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$, where:
  - pre is a finite set of pre-conditions, each being an ABox assertion;
    - specify the conditions for the application of the action;
  - occ is a finite set of occlusions, each being an ABox assertion of the form $A(c)$ or $P(c, c')$ (with A a concept name, P a role name, and c, c' constants);
    - the literals in the occlusion may change arbitrarily;
  - post is a finite set of conditional post-conditions, each of the form φ/ψ, where φ is an ABox assertion and ψ has the form $A(c)$, $\neg A(c)$, $P(c, c')$, or $\neg P(c, c')$;
    - if φ holds before applying the action, then ψ holds after its application;
    - the law of inertia applies;

A **complex action** is a finite sequence of atomic actions.

Conditional actions – Example

Opening a bank account: $openBA = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$

- The pre-condition is that the customer c is eligible for a bank account and holds a proof of address:

$$\mathsf{pre} : \{\ \mathsf{EligibleBAcc}(c), \exists\mathsf{holds}.\mathsf{ProofAddress}(c)\ \}$$

- If a letter from the employer is available, then the bank account comes with a credit card, otherwise not:

$$\begin{aligned}\mathsf{post} : \{\ & \top(c)/\mathsf{holds}(c,a),\\ & \exists\mathsf{holds}.\mathsf{Letter}(c)/\mathsf{BAccCredit}(a),\\ & \neg\exists\mathsf{holds}.\mathsf{Letter}(a)/\mathsf{BAccNoCredit}(a)\ \}\end{aligned}$$

- Nothing is free to change:

$$\mathsf{occ} : \{\ \}$$

The meaning of the concepts is defined in an acyclic TBox:

$$\begin{aligned}\mathsf{EligibleBAcc} &\equiv \exists\mathsf{permanentResident}.\{\mathsf{UK}\}\\ \mathsf{ProofAddress} &\equiv \mathsf{ElectricityContract}\end{aligned}$$

Semantics of actions and reasoning

Semantics of actions

- States of the world correspond to interpretations.
- Thus, the semantics of actions can be defined by means of a transition relation on interpretations.
- The transition relation ensures that what is specified in the conditional post-condition holds.
- Since the TBox is acyclic, defined concepts can be ignored in the definition, and actions with empty occlusion are deterministic.
- When there are conflicting post-conditions for $\mathcal{I}$ (i.e., φ_1/ψ and $\varphi_2/\neg\psi$, and both φ_1 and φ_2 hold in $\mathcal{I}$), then $\mathcal{I}$ has no successor interpretation.

Reasoning problems: Given an acyclic TBox $\mathcal{T}$, an ABox $\mathcal{A}$, and a composite action $\alpha_1 \cdots \alpha_k$

Executability: all models of $\langle\mathcal{T},\mathcal{A}\rangle$ satisfy pre_1, and at each step i, the reached models satisfy pre_{i+1}.

Projection: φ is consequence of applying $\alpha_1 \cdots \alpha_k$ in $\mathcal{A}$ and $\mathcal{T}$, if for each model $\mathcal{I}$ of $\langle\mathcal{T},\mathcal{A}\rangle$, the models reached from $\mathcal{I}$ satisfy φ.

Results on reasoning

Theorem

For all languages $\mathcal{L}$ between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$, executability and projection in $\mathcal{L}$ can be polynomially reduced to non-satisfiability in $\mathcal{LO}$ of an ABox relative to an acyclic TBox.

Theorem

Projection and executability are

- PSpace-complete for $\mathcal{ALC}$, $\mathcal{ALCO}$, $\mathcal{ALCQ}$, $\mathcal{ALCQO}$;
- ExpTime-complete for $\mathcal{ALCI}$, $\mathcal{ALCIO}$;
- coNExpTime-complete for $\mathcal{ALCQI}$, $\mathcal{ALCQIO}$.

Notes:

- The additional complexity caused by the introduction of nominals in the reduction of projection to ABox inconsistency cannot be avoided.
- Removing the restriction on acyclic TBoxes, or on possibly negated atomic concepts in post-conditions, leads to semantics and computational problems (notably, undecidability).
- For actions without occlusions, the approach is an instance of Reiter's Situation Calculus.

Combining LTL and DLs

We are in a case of logics with a two-dimensional semantics (objects and time).

The problem has different **dimensions**:

- temporal operators applied to concept expressions and/or TBox axioms and/or ABox assertions;
- no rigid symbols vs. rigid concepts and/or rigid roles

Already many results on the combinations of DLs and LTL:

- Rich settings, combining (extensions of) $\mathcal{ALC}$ with LTL, with rigid concepts, but no rigid roles:
  - temporal operators on concepts only: satisfiability is ExpTime-complete [Schild 1993]
  - temporal operators on concept expressions and (TBox + ABox) axioms: satisfiability is ExpSpace-complete [Wolter and Zakharyaschev 1999a; Gabbay et al. 2003].

  Satisfiability becomes undecidable with rigid roles [Gabbay et al. 2003].
- With rigid roles, decidability can be obtained by strongly restricting:
  - the temporal component (S5, instead of LTL) [Artale, Lutz, et al. 2007], or
  - the DL component (*DL-Lite*, instead of $\mathcal{ALC}$) [Artale, Kontchakov, et al. 2007]

LTL over $\mathcal{ALC}$ axioms [Baader, Ghilardi, et al. 2012]

They propose to consider a different way of combining LTL and $\mathcal{ALC}$:
temporal operators only on (TBox + ABox) axioms, but not on concept constructors.

Theorem

Satisfiability in $\mathcal{ALC}$-LTL, where temporal operators are applied only on (TBox + ABox) axioms is

- 2ExpTime-complete with rigid roles;
- NExpTime-complete with rigid concepts, but no rigid roles;
- ExpTime-complete without rigid symbols.

Idea for decidability and complexity:

- Build **propositional abstraction** of an $\mathcal{ALC}$-LTL formula, by replacing each $\mathcal{ALC}$ axiom by a propositional variable.
- Encode in LTL that at each time-point some (guessed) subset of the set of $\mathcal{ALC}$ axioms holds.
- Do some additional checks to account for rigid concepts and roles.

Runtime verification

In model checking, one wants to check whether all possible traces of a transition system describing a system's behavior satisfy a given formula.

Runtime verification

- One observes the system behavior, having at any time point a finite prefix u of a trace.
- The task is to check whether all continuations of u to a trace satisfy/violate an LTL formula φ.
- Given (u, φ), the monitor might return three possible answers:
  - $\top$, if all continuations of u to an infinite trace satisfy φ;
  - $\bot$, if all continuations of u to an infinite trace violate φ;
  - ?, if none of the above holds, i.e., there is a continuation satisfying φ, and one violating φ.

Example: $\varphi_{tv} = \Box(\text{turnOff} \rightarrow \mathbf{X}(\text{off} \land (\mathbf{X}\text{off})\,\mathbf{U}\,\text{turnOn}))$

- For prefix $u_1 = \{\text{on}, \neg\text{turnOn}, \text{turnOff}\}$, the answer to (u_1, φ_{tv}) is ?.
- For prefix $u_2 := \{\text{on}, \neg\text{turnOn}, \text{turnOff}\}\ \{\text{on}, \neg\text{turnOn}, \neg\text{turnOff}\}$, the answer to (u_2, φ_{tv}) is $\bot$.

Runtime verification in $\mathcal{ALC}$-LTL

In the $\mathcal{ALC}$-LTL setting:

- the LTL formula is built over $\mathcal{ALC}$ axioms, and
- the prefix describing the system's behavior is a sequence of $\mathcal{ALC}$ ABoxex.

Hence:

- We have incomplete information also in the prefix, since neither an axiom nor its negation may follow from an ABox.
- We have another source of uncertainty, which may cause the monitor to answer ?.

Results by [Baader and Lippmann 2014]

- How to construct Büchi automata for checking satisfiability of $\mathcal{ALC}$-LTL formulae.
- How to construct a monitor (which is a deterministic Moore automaton) for an $\mathcal{ALC}$-LTL specification and a formula φ.
- Tight complexity bounds for various monitoring tasks (e.g., 2ExpTime for liveness and for deciding monitorability).

Temporalizing ontology-mediated query answering

In many settings, the query capabilities provided by DLs are not sufficient.
E.g., to extract complex patterns from the data, one may want to use database-inspired queries.

Setting inspired by an application of situation awareness [Baader, Borgwardt, et al. 2013]:

- A DL TBox $\mathcal{T}$ describes the structural properties of the domain of interest that always hold.
  Example:

$$\begin{aligned}\exists \mathsf{systolicPressure}.\mathsf{HighPressure} &\sqsubseteq \exists \mathsf{finding}.\mathsf{Hypertension}\\ \exists \mathsf{finding}.\mathsf{Hypertension} \sqcap \exists \mathsf{history}.\mathsf{Hypertension} &\sqsubseteq \exists \mathsf{risk}.\mathsf{MyocardialInfarction}\end{aligned}$$

- A finite sequence $\mathcal{A}_0, \ldots, \mathcal{A}_n$ of ABoxes (partially) describe the n previous states of the world.
- Queries are expressed in LTL, in which atoms are conjunctive queries over the TBox vocabulary.
  Example: male patients with a history of hypertension:

$$\mathsf{Male}(x) \land \bigcirc^{-} \Diamond^{-} (\exists y.\mathsf{finding}(x, y) \land \mathsf{Hypertension}(y))$$

Temporalizing ontology-mediated query answering – Results

In this setting, it makes sense to distinguish

- data complexity, i.e., the complexity measured in the size of the ABoxes only, and
- combined complexity, i.e., the complexity measured in the size of the whole input.

Results for LTL conjunctive query entailment for $\mathcal{ALC}$ [Baader, Borgwardt, et al. 2013]:

	Data complexity	Combined complexity
no rigid symbols	coNP-complete	ExpTime-complete
rigid concepts only	coNP-complete	coNExpTime-complete
rigid concepts and roles	coNP-hard, in ExpTime	2ExpTime-complete

Note: the tight upper bounds for data complexity and combined complexity, require different algorithms.

Temporalizing ontology-mediated query answering – More results

The previous results have been extended to many more DLs [Baader, Borgwardt, et al. 2015]:

	Data complexity			Combined complexity		
	no	RC	RCR	no	RC	RCR
$\mathcal{ALC}$– $\mathcal{ALCHQ}$	coNP	coNP	$\leq$Exp	Exp	coNExp	2Exp
$\mathcal{ALCO}$ – $\mathcal{ALCHOQ}/\mathcal{ALCHOI}$	coNP	coNP	$\leq$Exp	$\geq$coNExp	?	2Exp
$\mathcal{S}$ – $\mathcal{SQ}$	coNP	coNP	$\leq$Exp	$\geq$coNExp	?	2Exp
$\mathcal{SO}$ – $\mathcal{SOQ}$	$\geq$coNP	?	$\leq$Exp	$\geq$coNExp	?	2Exp
$\mathcal{SH}/\mathcal{ALCI}$ – $\mathcal{SHIQ}$	coNP	coNP	$\leq$Exp	2Exp	2Exp	2Exp
$\mathcal{SHO}$ – $\mathcal{SHOQ}/\mathcal{SHOI}$	$\geq$coNP	?	$\leq$Exp	2Exp	2Exp	2Exp
$\mathcal{ALCOIQ}$ – $\mathcal{ALCHOIQ}$	$\geq$coNP	?	decidable	$\geq$coN2Exp	?	decidable
$\mathcal{SOIQ}$ – $\mathcal{SHOIQ}$	$\geq$coNP	?	?	$\geq$coN2Exp	?	?

Outline

Levesque's functional approach

Adopt a radical solution: assume a **functional view of ontologies** [Levesque 1984]:

Levesque's functional approach

View KB as systems that allow for two kinds of operations

- $\text{ASK}(q, s)$, which returns the answers to a query q that are logically implied by the ontology s,
- $\text{TELL}(a, s)$, which produces a new ontology s' as a result of the application of an action a to the ontology s.

Note:

1. Essentially, this amounts to applying actions/temporal operators to DL axioms only.
2. Also related to update.
   But result of TELL remains in the same language as the original ontology.

Levesque's functional approach – Pros and Cons

Major advantage:

- Strong decoupling of reasoning on the static knowledge from reasoning on the dynamics of the computations over such knowledge.
- As a result, we can lift to DLs many notions and results developed over the years in Reasoning about Actions, Process Modeling, and Verification.

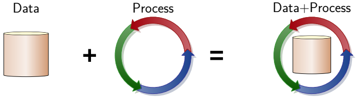
Disadvantages:

- We don't have a single theory anymore for representing and reasoning on actions over ontologies:
  - The ontology represents what is known.
  - The actions change what is known (i.e., the ontology), but they are not represented in the (same) ontology.
- We lose the possibility of distinguishing between "knowledge" and "truth".

Knowledge and Action Base (KAB)

[Bagheri Hariri, C., De Giacomo, et al. 2013; Bagheri Hariri, C., Montali, et al. 2013]

Representing both structural and behavioral aspects:



KAB

- Data Layer: Description logic KB
  - Data schema: TBox
  - Data instance: ABox
- Process Layer:
  - Atomic actions: access and update data;
  - Process: finite state control over conditional action invocation;
  - External calls: communication with external environment
    - Insert new data objects possibly depending on already present objects.

Actions and processes in KABs

Actions

- Have input parameters.
- Action execution results in a new KB: ABox changes while TBox remains fixed.
- Resulting KBs may contain new objects that come from the environment outside the system.

Processes

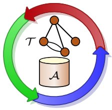
A process over a KAB is a specification of when actions can be executed.

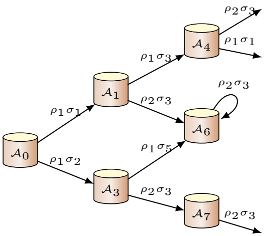
- Must be based on querying the current ontology.
- With additional non-deterministic finite control state programs.

KAB transition system

KAB behavior captured by the infinite-state transition system Υ, defined as follows:

1. Start from the given initial state $\mathcal{A}_0$.
2. Use process specification to choose instantiated action $\rho\sigma$ to execute.
3. If resulting state/ABox is **consistent w.r.t. TBox**, it becomes **a new state**.
4. Repeat forever.





Sources of infinity:

- Infinite branching – infinitely many parameter instantiations.
- Infinite runs – usage of values obtained from action steps.
- Unbounded Aboxes – accumulation of such values.

Verification of KABs

Check sophisticated temporal properties over KABs.

Example

- Eventually all people in this room will be sleeping.
- From now on all people in this room will be interested in KABs.
- There exists a possible future situation where all people in this room will be interested in KABs.
- Always, when a presentation starts, it eventually ends.

Verification logics

We are interested in checking very powerful dynamic/temporal properties.
$\leadsto$ FOL variants of LTL and μ-calculus.

Verification logic

FOL variant of μ-calculus

$$\Phi ::= Q \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \exists x.\Phi \mid [-]\Phi \mid \langle - \rangle\Phi \mid \mu Z.\Phi \mid \nu Z.\Phi \mid Z$$

- Q is an ECQ (first-order like syntax, but with restricted quantification) [C., De Giacomo, Lembo, Lenzerini, et al. 2007]
- Existential quantification ranges over terms in the current KB.
- Fixpoint operators are fully general.

Example

- $\forall x.PeopleInRoom(x) \rightarrow \mu Z.(Sleeping(x) \vee [-]Z)$
  Eventually all people in this room will be sleeping (liveness).
- $\forall x.PeopleInRoom(x) \rightarrow \nu Z.(InterestedInKABs(x) \wedge [-]Z)$
  From now on all people in this room wil be interested in KABs (safety).
- $\forall x.PeopleInRoom(x) \rightarrow \mu Z.(InterestedInKABs(x) \vee \langle - \rangle Z)$
  There exists a possible future situation where all people in this room will be interested in KABs.

Verification of KABs – Undecidability

Model checking

Given a KAB, and a verification formula Φ, check whether Φ holds in the KAB Transition System Υ, denoted by $\Upsilon \models \Phi$.

- KABs transition system has infinitely many states since the number of different KBs that can be obtained by executing actions according to the process is unbounded.
- The usual techniques, e.g., model checking, used for finite-state systems don't work off-the-shelf.

Theorem

Model checking verification formulas on KABs is **undecidable**.
(Even for trivial eventualities with no quantification across on a KAB with empty TBox, no equality, and actions without negation).

Proof: By reduction from answering boolean UCQs in relational DBs under a set of TGDs [Beeri and Vardi 1981].

Technical challenges in verification

Challenge 1: Combining DLs and actions

- Adopt Levesque's functional view of KBs: KBs are objects that can be:
  - queried through logical reasoning, returning certain answers;
  - updated through an extra-logical mechanism, generating a new KB.
- Actions make system transit from the current KB to the next one, specified according to the effects of actions.

Challenge 2: Dealing with infinite state transition systems

- External calls insert new objects in the KB.
- Even under Levesque's functional view, the number of KBs to be considered in the system evolution is in general infinite.
- Getting a solution to this is the crux of success.

Challenge 3: Allowing quantification across in the verification logic

- Verification properties query the current state/KB *through certain answers*.
- Quantifying across: we retrieve objects in one state and check their properties in future states.

History-preserving and persistence-preserving μ-calculus

We introduce two extensions of the modal μ-calculus / LTL with (controlled) forms of first order quantification.

History-preserving quantification: $\mu\mathcal{L}_A$ / LTL-FO$_A$

FO quantification ranges over current active domain only.

Examples:

LTL-FO$_A$: $\forall x.\ Customer(x) \rightarrow \mathbf{F}\ Gold(x)$

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

Persistence-preserving quantification: $\mu\mathcal{L}_P$ / LTL-FO$_P$

FO quantification ranges over persisting individuals only.

Examples:

LTL-FO$_P$: $\forall x.\ Gold(x) \rightarrow \mathbf{G}\ Gold(x)$

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

$\mu\mathcal{L}_{FO}$/LTL-FO

$\uparrow$

$\mu\mathcal{L}_A$/LTL-FO$_A$

$\uparrow$

$\mu\mathcal{L}_P$/LTL-FO$_P$

$\uparrow$

$\mu\mathcal{L}$

$\uparrow$

LTL

Bisimulation between transition systems

The local condition of the bisimulation is isomorphism between states.

History-preserving bisimilarity

History-preserving bisimilarity requires that isomorphism in the new states extends old ones.

Persistence-preserving bisimilarity

Persistence-preserving bisimilarity requires that isomorphism in the new states extends old ones only on objects in the active domain.

Theorem (bisimulation invariance)

- History-preserving bisimilar transition systems satisfy the same $\mu\mathcal{L}_A$ (history-preserving) formulas.
- Persistence-preserving bisimilar transition systems satisfy the same $\mu\mathcal{L}_P$ (persistence-preserving) formulas.

Two key semantical conditions for decidability

Run-boundedness

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

- Transition system accumulates finite information along a run.
- But accumulates infinite information through infinite branching.
- This is a semantic condition, whose checking is undecidable.
  $\leadsto$ Easy to check syntactic conditions needed: Weak-acyclicity.

State-boundedness

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

- Relaxation of run-boundedness.
- Infinite accumulation of information along each run is possible.
- This is a semantic condition, whose checking is undecidable.
  $\leadsto$ Easy to check syntactic conditions needed: GR-acyclicity.

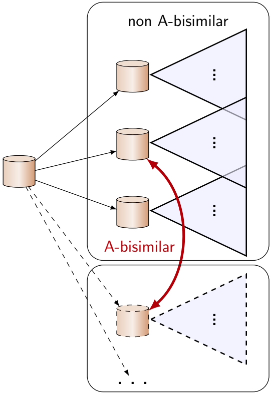
Decidability results for run-bounded KABs

Theorem

Verification of $\mu\mathcal{L}_A$ over run-bounded KABs is decidable and can be reduced to model checking of propositional μ-calculus over a finite transition system.

Idea: use **isomorphic types** instead of actual values.

Remember: runs are bounded!



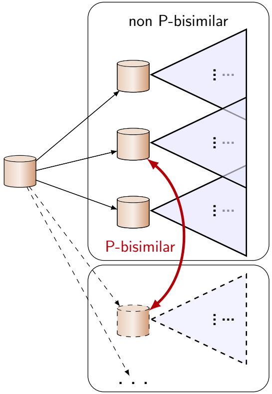
Decidability results for state-bounded KABs

Theorem

Verification of $\mu\mathcal{L}_P$ over state-bounded KABs is decidable and can be reduced to model checking of propositional μ-calculus over a finite transition system.

Steps:

1. **Prune** infinite branching (isomorphic types).
2. Finite abstraction along the runs:
   - $\mu\mathcal{L}_P$ looses track of previous values that do not exist anymore.
   - New values can be replaced with old, non-persisting ones.
   - This eventually leads to **recycle** the old values without generating new ones.



Outline

Semantically-Governed Artifact Systems (SGASs)

The data layer in a dynamic system constituted of evolving artifacts might be very complex, and difficult to interact with.

Hence we can resort to 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 system via declarative mappings.

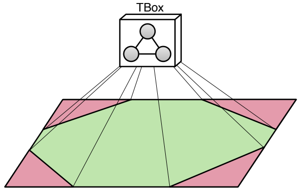
Such a setting gives rise to a very rich and still largely unexplored framework, in which we have various choices for:

- 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 tasks we want to perform.

Initial results reported in [C., De Giacomo, Lembo, Montali, et al. 2012].

Semantically-Governed Artifact Systems (SGASs)

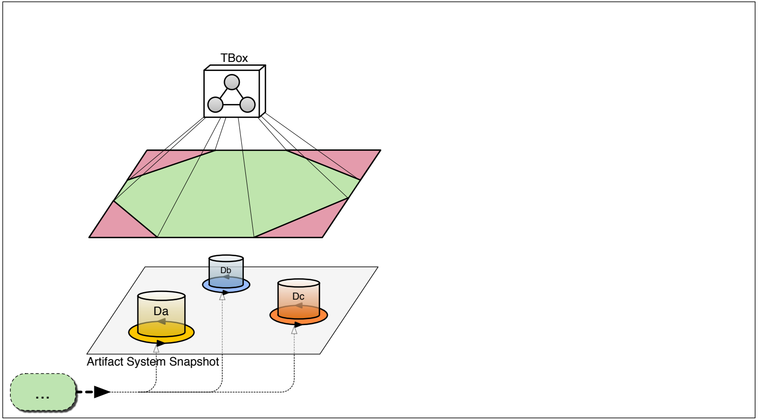
The system's conceptual schema (TBox) composed of semantic constraints that define the "data boundaries" of the underlying artifact system.



Semantic layer and snapshots

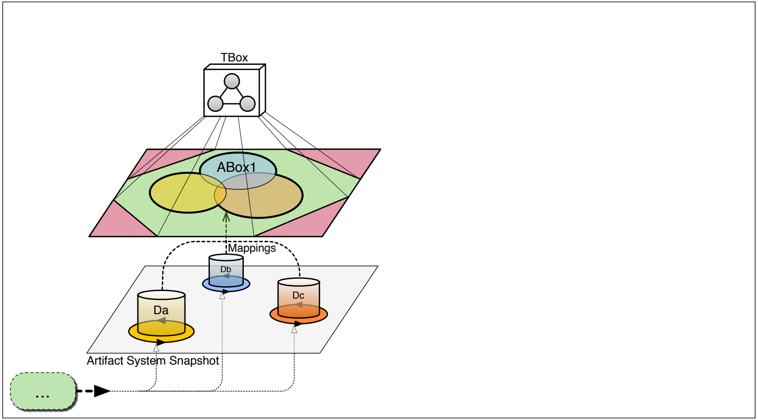
Actual data are concretely maintained at the artifact layer.

Snapshot: database instances of the involved artifacts.



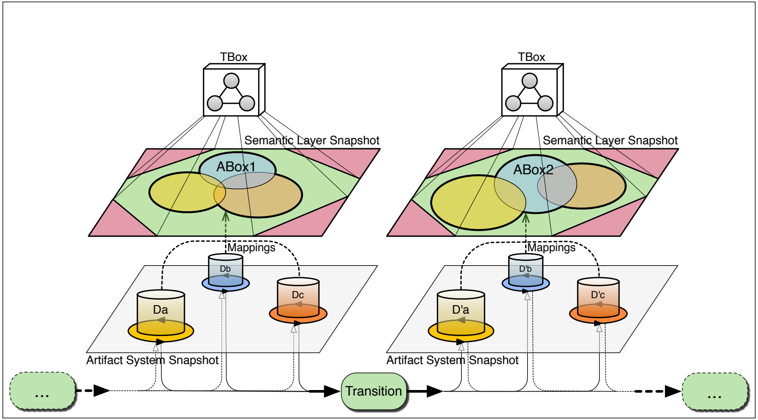
Mappings

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



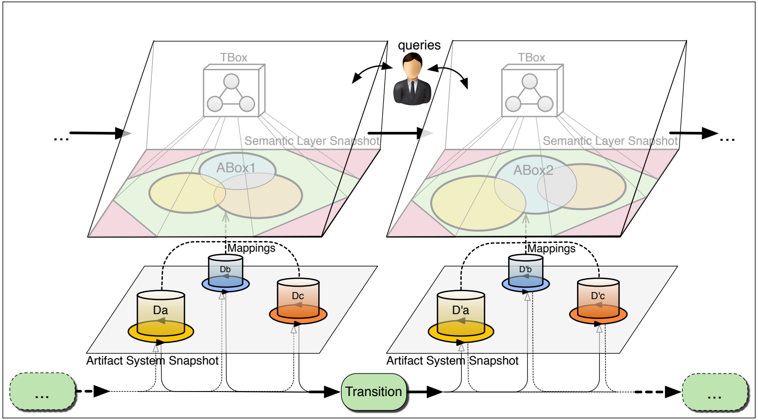
Action execution to evolve the system

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



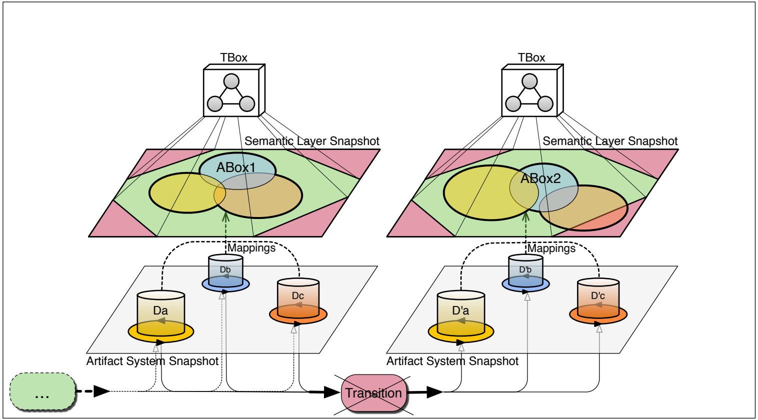
Understanding the evolution

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

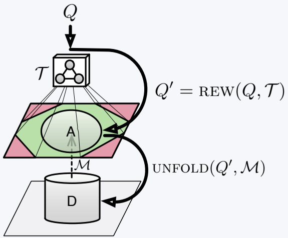
Temporal properties expressed as:

- 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 SGASs

We obtain that the verification of (restricted first-order) temporal properties is decidable, provided the transition system at the Artifact Layer satisfies suitable boundedness conditions.

Results

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

- Verification of LTL-FO$_A/\mu\mathcal{L}_A$ properties over run-bounded SGASs with deterministic services.
- Verification of LTL-FO$_P/\mu\mathcal{L}_P$ properties over state-bounded SGASs (both with deterministic and with non-deterministic services).

Outline

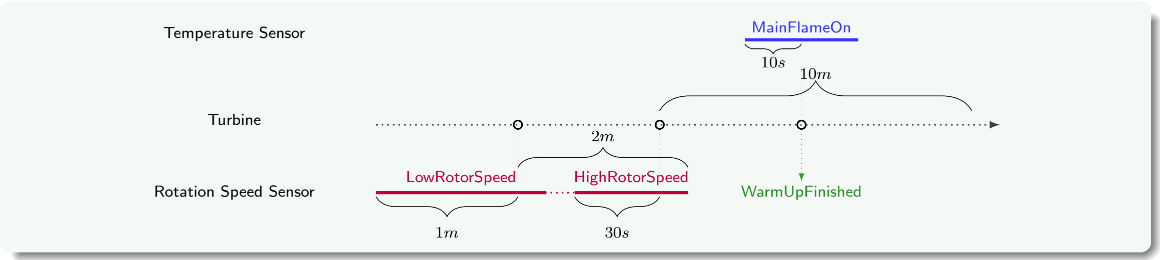
Modeling static domain knowledge using an ontology

We can model the static domain knowledge using one of the standard ontology languages, e.g., **OWL 2 QL**.

Devices consist of parts, which are monitored by different kinds of sensors (rotation, pressure, ...).

$$
\begin{aligned}
\mathsf{GasTurbine} &\sqsubseteq \mathsf{Turbine} \\
\mathsf{SteamTurbine} &\sqsubseteq \mathsf{Turbine} \\
\mathsf{PowerUnit} &\sqsubseteq \mathsf{TurbinePart} \\
\mathsf{Burner} &\sqsubseteq \mathsf{TurbinePart} \\
\mathsf{TemperatureSensor} &\sqsubseteq \mathsf{Sensor} \\
\mathsf{RotationSpeedSensor} &\sqsubseteq \mathsf{Sensor} \\
\mathsf{PressureSensor} &\sqsubseteq \mathsf{Sensor} \\
&\cdots
\end{aligned}
\qquad
\begin{aligned}
\exists \mathsf{isDeployedIn} &\sqsubseteq \mathsf{Turbine} \\
\exists \mathsf{isDeployedIn}^- &\sqsubseteq \mathsf{Train} \\
\exists \mathsf{isPartOf} &\equiv \mathsf{TurbinePart} \\
\exists \mathsf{isPartOf}^- &\sqsubseteq \mathsf{Turbine} \\
\exists \mathsf{isMonitoredBy} &\sqsubseteq \mathsf{TurbinePart} \\
\exists \mathsf{isMonitoredBy}^- &\sqsubseteq \mathsf{Sensor} \\
&\cdots
\end{aligned}
$$

Modeling complex patterns



We can model this temporal pattern, e.g., with **metric temporal rules**:

$$
\begin{aligned}
\mathsf{LowRotorSpeed}(tb) &\leftarrow \mathsf{rotorSpeed}(tb, rs) \land rs < 1000.\\
\mathsf{HighRotorSpeed}(tb) &\leftarrow \mathsf{rotorSpeed}(tb, rs) \land rs > 1260.\\
\mathsf{MainFlameOn}(tb) &\leftarrow \mathsf{mainflame}(tb, mf) \land mf > 1.\\
\mathsf{WarmUpFinished}(tb) &\leftarrow \boxminus_{[0s,10s]}\mathsf{MainFlameOn}(tb) \land\\
&\quad \diamondminus_{(0,10m]}\big(\boxminus_{(0,30s]}\mathsf{HighRotorSpeed}(tb) \land\\
&\qquad \diamondminus_{(0,2m]}\boxminus_{(0,1m]}\mathsf{LowRotorSpeed}(tb)\big)
\end{aligned}
$$

We use *DatalogMTL*

DatalogMTL is a Horn fragment of Metric Temporal Logic (MTL).

A DatalogMTL *program* is a finite set of rules of the form

$$A^+ \leftarrow A_1 \wedge \cdots \wedge A_k \qquad \text{or} \qquad \bot \leftarrow A_1 \wedge \cdots \wedge A_k,$$

where

- each A_i is either $\tau \neq \tau'$, or defined by the grammar

$$A ::= P(\tau_1, \ldots, \tau_m) \mid \boxplus_\varrho A \mid \boxminus_\varrho A \mid \diamondplus_\varrho A \mid \diamondminus_\varrho A$$

  where ϱ denotes a (left/right open or closed) interval with non-negative endpoints,
- A^+ does not contain $\diamondplus_\varrho$ or $\diamondminus_\varrho$ (since this would lead to undecidability).

Query evaluation in *DatalogMTL*

[Brandt, Kalayci, et al. 2017; Brandt, Güzel Kalayci, et al. 2018]

Theorem

Answering *DatalogMTL* queries is ExpSpace-complete in combined complexity.

We consider the **nonrecursive fragment Datalog$_{nr}$MTL** of *DatalogMTL*:

- sufficient expressive power for many real-world situations
- computationally well-behaved

Answering *Datalog$_{nr}$MTL* queries:

- Is PSpace-complete in combined complexity.
- Is in AC^0 in data complexity.
- Can be **reduced to SQL query evaluation**.

Hence, *Datalog$_{nr}$MTL* is well suited as a temporal rule language for OBDA.

Data sources: schema and data

Data sources often contain temporal information in the form of time-stamps.

Example data schema $\mathcal{S}$ for the Siemens data

It includes time-stamped sensor measurements and deployment details:

tb_measurement(timestamp, sensor_id, value),
tb_sensors(sensor_id, sensor_type, mnted_part, mnted_tb),
tb_components(turbine_id, component_id, component_type).

A corresponding data instance $\mathcal{D}_0$:

tb_measurement

timestamp	*sensor_id*	*value*
2017-06-06 12:20:00	rs01	570
2017-06-06 12:22:50	rs01	1278
2017-06-06 12:23:40	rs01	1310
...	...	...
2017-06-06 12:32:30	mf01	2.3
2017-06-06 12:32:50	mf01	1.8
2017-06-06 12:33:40	mf01	0.9
...	...	...

tb_sensors

sensor_id	*sensor_type*	*mnted_part*	*mnted_tb*
rs01	0	pt01	tb01
mf01	1	b01	tb01
...	...	...	...

tb_components

turbine_id	*component_id*	*component_type*
tb01	pt01	0
tb01	b01	1
...	...	...

Static mapping assertions in $\mathcal{M}_s$

Static mapping assertions: $\Phi(\vec{x}) \rightsquigarrow \Psi(\vec{x})$

- $\Phi(\vec{x})$ is a query over the source schema $\mathcal{S}$
- $\Psi(\vec{x})$ is an atom with predicate in Σ_s

Example

```
SELECT sensor_id AS X FROM tb_sensors
WHERE sensor_type = 1                      ⇝ TemperatureSensor(X)

SELECT component_id AS X FROM tb_components
WHERE component_type = 1                   ⇝ Burner(X)

SELECT mnted_part AS X, sensor_id AS Y FROM tb_sensors ⇝ isMonitoredBy(X, Y)
```

These mappings retrieve from the database ordinary facts.

$$\text{Burner}(\texttt{b01}), \quad \text{TemperatureSensor}(\texttt{mf01}),$$
$$\text{isMonitoredBy}(\texttt{pt01}, \texttt{rs01}), \quad \text{isMonitoredBy}(\texttt{b01}, \texttt{mf01}).$$

Temporal mapping assertions in $\mathcal{M}_t$

Temporal mapping assertions: $\Phi(\vec{x}, \mathtt{begin}, \mathtt{end}) \rightsquigarrow \Psi(\vec{x})@\langle t_{\mathtt{begin}}, t_{\mathtt{end}}\rangle$

- begin and end are variables returning a date/time.
- '$\langle$' is either '(' or '[', and similarly for '$\rangle$'.
- $\Psi(\vec{x})$ is an atom with predicate in Σ_t.
- $t_{\mathtt{begin}}$ is either begin or a date-time constant, and similarly for $t_{\mathtt{end}}$.

Example

```
SELECT * FROM (
  SELECT sensor_id, value, timestamp AS begin,
         LEAD(timestamp,1) OVER W AS end
  FROM tb_measurement, tb_sensors
  WINDOW W AS (PARTITION BY sensor_id ORDER BY timestamp)
  WHERE tb_measurement.sensor_id = tb_sensors.sensor_id AND sensor_type = 0
) SUBQ WHERE value > 1260
```

$\rightsquigarrow$ HighRotorSpeed(sensor_id)@[begin,end)

These mappings retrieve from the database temporal facts.

HighRotorSpeed(rs01)@[2017-06-06 12:22:50, 2017-06-06 12:23:40)

Concrete syntax for temporal OBDA specifications

Temporal OBDA specification $\mathcal{P}_t = \langle \Sigma_s, \Sigma_t, \mathcal{O}, \mathcal{R}_s, \mathcal{R}_t, \mathcal{M}_s, \mathcal{M}_t, \mathcal{S} \rangle$

- Σ_s is a static vocabulary,
- Σ_t is a temporal vocabulary,
- $\mathcal{O}$ is an ontology,
- $\mathcal{R}_s$ is a set of static rules,
- $\mathcal{R}_t$ is a set of temporal rules,
- $\mathcal{M}_s$ is a set of static mapping assertions,
- $\mathcal{M}_t$ is a set of temporal mapping assertions,
- $\mathcal{S}$ is a database schema.

Component	defines predicates in	in terms of predicates in	Adopted language
$\mathcal{O}$	Σ_s	Σ_s	OWL 2 QL
$\mathcal{R}_s$	Σ_s	Σ_s	non-recursive Datalog
$\mathcal{R}_t$	Σ_t	$\Sigma_s \cup \Sigma_t$	$Datalog_{nr}MTL$
$\mathcal{M}_s$	Σ_s	$\mathcal{S}$	R2RML / Ontop
$\mathcal{M}_t$	Σ_t	$\mathcal{S}$	R2RML / Ontop

Temporal OBDA instance

A **temporal OBDA instance** is a pair $\mathcal{J} = \langle \mathcal{P}_t, \mathcal{D} \rangle$

- $\mathcal{P}_t$ is a temporal OBDA specification,
- $\mathcal{D}$ is a database instance compliant with the database schema $\mathcal{S}$ in $\mathcal{P}_t$.

To represent the facts generated by $\mathcal{J}$, we use **RDF Datasets**.

(We follow the model proposed by the W3C RSP Community Group.)

The temporal fact $\mathsf{HighRotorSpeed}(\mathtt{rs01})@[\mathtt{2017\text{-}06\text{-}06\ 12{:}22{:}50}, \mathtt{2017\text{-}06\text{-}06\ 12{:}23{:}40})$

is modeled as the **named graph** $\mathsf{GRAPH}\ g_0\ \{(\mathtt{rs01}, \mathtt{a}, \mathtt{HighRotorSpeed})\}$

and a set of triples in the default graph, to model the time interval:

$$
\begin{array}{l}
(g_0, \mathtt{a}, \mathtt{time{:}Interval}),\\
(g_0, \mathtt{time{:}isBeginningInclusive}, \mathtt{true}),\ (g_0, \mathtt{time{:}isEndInclusive}, \mathtt{false}),\\
(g_0, \mathtt{time{:}hasBeginning}, b_0),\ (g_0, \mathtt{time{:}hasEnd}, e_0),\\
(b_0, \mathtt{time{:}inXSDDateTimeStamp}, \text{`}\mathtt{2017\text{-}06\text{-}06\ 12{:}22{:}50}\text{'}),\\
(e_0, \mathtt{time{:}inXSDDateTimeStamp}, \text{`}\mathtt{2017\text{-}06\text{-}06\ 12{:}23{:}40}\text{'}).
\end{array}
$$

Query answering

Example query

"Find the time periods of "Warm up" of the gas turbines deployed in the train with ID T001".

This corresponds to the query: $Q(tb)@\iota$, where $Q(tb)$ is defined as

$$\mathsf{Q}(tb) \leftarrow \mathsf{Train}(\mathtt{T001}),\ \mathsf{GasTurbine}(tb),\ \mathsf{isDeployedIn}(tb, \mathtt{T001}),\ \mathsf{WarmupFinished}(tb).$$

In "temporal" SPARQL syntax:

```
PREFIX ss: <http://siemens.com/ns#>
PREFIX st: <http://siemens.com/temporal/ns#>

SELECT ?tb ?left_edge ?begin ?end ?right_edge
WHERE {
  ?tb a ss:GasTurbine ;
      ss:isDeployedIn ss:train_T001 .
  {?tb a st:PurgingIsOver}@<?left_edge,?begin,?end,?right_edge>
}
```

Equivalent standard SPARQL query

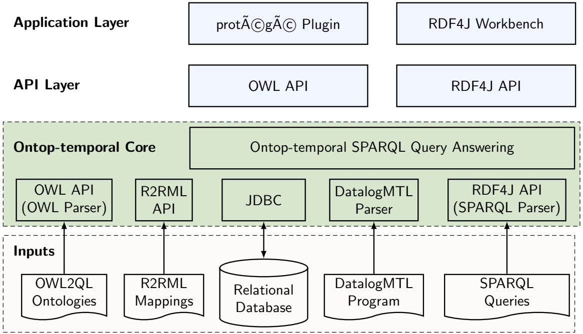
```
PREFIX ss: <http://siemens.com/ns#>
PREFIX st: <http://siemens.com/temporal/ns#>

SELECT ?tb ?left_edge ?begin ?end ?right_edge
WHERE {
  ?tb a ss:GasTurbine ;  ss:isDeployedIn ss:train_T001 .
  {?tb a st:WarmupFinished}@<?left_edge,?begin,?end,?right_edge>
}
```

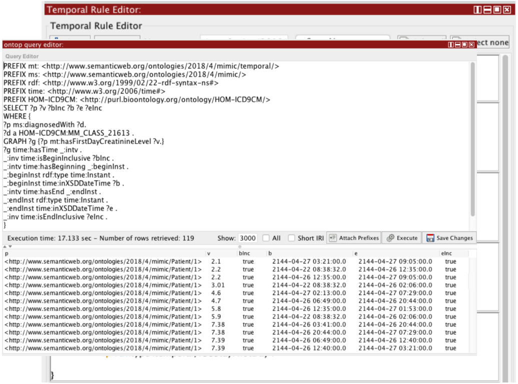
... is translated into the following standard SPARQL query:

```
PREFIX time: <http://www.w3.org/2006/time#>
...
SELECT ?tb ?left_edge ?begin ?end ?right_edge
WHERE {
  ?tb a ss:GasTurbine ;  ss:isDeployedIn ss:train_T001 .
  GRAPH ?g { ?tb a st:WarmupFinished . }
  ?g a time:Interval ;
     time:isBeginningInclusive ?left_edge ;
     time:hasBeginning [ time:inXSDDateTimeStamp ?begin ] ;
     time:hasEnd [ time:inXSDDateTimeStamp ?end ] ;
     time:isEndInclusive ?right_edge .
}
```

The Ontop-temporal system

We have implemented the temporal OBDA framework in the **Ontop-temporal** system [Güzel Kalayci et al. 2018], as an extension of the OBDA system *Ontop*.





Outline

Conclusions

- There is a huge amount of work in knowledge representation and in database theory, using a variety of techniques, that is **potentially relevant** to Complex Event Recognition.
- 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.
- Some of the assumptions made would need validation also from the practical point of view, in real-world use-cases.
  $\rightsquigarrow$ Requires making frameworks more robust.
- Moreover, the positive results on reasoning and verification appear rather fragile.
- I am curious to see which of the works/results/techniques find your interest, and concrete application in CER.

Acknowledgements

Thanks to the many people that contributed interesting ideas, suggestions, discussions, and with whom I collaborated on some of the presented results:

Babak Bagheri Hariri
Giuseppe De Giacomo
Riccardo De Masellis
Alin Deutsch
Paolo Felli
Elema Güzel Kalayci
Roman Kontchakov
Rick Hull
Maurizio Lenzerini
Alessio Lomuscio
Marco Montali
Fabio Patrizi
Vladislav Ryzhikov
Ario Santoso
Moshe Vardi
Guohui Xiao
Michael Zakharyaschev

Thank you for your attention!

References I

[1] 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.

[2] M. Lenzerini and P. Nobili. "On the Satisfiability of Dependency Constraints in Entity-Relationship Schemata". In: *Information Systems* 15.4 (1990), pp. 453–461.

[3] S. Bergamaschi and C. Sartori. "On Taxonomic Reasoning in Conceptual Design". In: *ACM Trans. on Database Systems* 17.3 (1992), pp. 385–422.

[4] A. Borgida. "Description Logics in Data Management". In: *IEEE Trans. on Knowledge and Data Engineering* 7.5 (1995), pp. 671–682.

[5] D. C., M. Lenzerini, and D. Nardi. "Unifying Class-Based Representation Formalisms". In: *J. of Artificial Intelligence Research* 11 (1999), pp. 199–240.

[6] A. Borgida and R. J. Brachman. "Conceptual Modeling with Description Logics". In: *The Description Logic Handbook: Theory, Implementation and Applications*. Ed. by F. Baader, D. C., D. McGuinness, D. Nardi, and P. F. Patel-Schneider. Cambridge University Press, 2003. Chap. 10, pp. 349–372.

References II

[7] D. Berardi, D. C., and G. De Giacomo. "Reasoning on UML Class Diagrams". In: *Artificial Intelligence* 168.1–2 (2005), pp. 70–118.

[8] A. Queralt, A. Artale, D. C., and E. Teniente. "OCL-Lite: Finite Reasoning on UML/OCL Conceptual Schemas". In: *Data and Knowledge Engineering* 73 (2012), pp. 1–22.

[9] F. Baader. "Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles". In: *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. 1991.

[10] K. Schild. "A Correspondence Theory for Terminological Logics: Preliminary Report". In: *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. 1991, pp. 466–471.

[11] K. Schild. "Combining Terminological Logics with Tense Logic". In: *Proc. of the 6th Portuguese Conf. on Artificial Intelligence (EPIA)*. Vol. 727. Lecture Notes in Computer Science. Springer, 1993, pp. 105–120.

[12] G. De Giacomo and M. Lenzerini. "Concept Language with Number Restrictions and Fixpoints, and its Relationship with μ-Calculus". In: *Proc. of the 11th Eur. Conf. on Artificial Intelligence (ECAI)*. 1994, pp. 411–415.

References III

[13] K. Schild. "Terminological Cycles and the Propositional μ-Calculus". In: *Proc. of the 4th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*. 1994, pp. 509–520.

[14] G. De Giacomo and M. Lenzerini. "Boosting the Correspondence between Description Logics and Propositional Dynamic Logics". In: *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI)*. 1994, pp. 205–212.

[15] G. De Giacomo and M. Lenzerini. "What's in an Aggregate: Foundations for Description Logics with Tuples and Sets". In: *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. 1995, pp. 801–807.

[16] G. De Giacomo and M. Lenzerini. "TBox and ABox Reasoning in Expressive Description Logics". In: *Proc. of the 5th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*. 1996, pp. 316–327.

[17] D. C., G. De Giacomo, and M. Lenzerini. "Structured Objects: Modeling and Reasoning". In: *Proc. of the 4th Int. Conf. on Deductive and Object-Oriented Databases (DOOD)*. Vol. 1013. Lecture Notes in Computer Science. Springer, 1995, pp. 229–246.

References IV

[18] G. De Giacomo and M. Lenzerini. "PDL-Based Framework for Reasoning about Actions". In: *Proc. of the 4th Conf. of the Italian Assoc. for Artificial Intelligence (AI*IA)*. Vol. 992. Lecture Notes in Artificial Intelligence. Springer, 1995, pp. 103–114.

[19] F. Baader and A. Laux. "Terminological Logics with Modal Operators". In: *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. 1995, pp. 808–814.

[20] F. Wolter and M. Zakharyaschev. "Satisfiability Problem in Description Logics with Modal Operators". In: *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*. 1998, pp. 512–523.

[21] F. Wolter and M. Zakharyaschev. "Temporalizing Description Logics". In: *Proc. of the 2nd Int. Workshop on Frontiers of Combining Systems (FroCoS)*. Ed. by M. de Rijke and D. Gabbay. Amsterdam: Wiley, 1999.

[22] F. Wolter and M. Zakharyaschev. "Multi-Dimensional Description Logics". In: *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. 1999, pp. 104–109.

[23] F. Wolter and M. Zakharyaschev. "Modal Description Logics: Modalizing Roles". In: *Fundamenta Informaticae* 39.4 (1999), pp. 411–438.

References V

[24] D. Gabbay, A. Kurusz, F. Wolter, and M. Zakharyaschev. *Many-dimensional Modal Logics: Theory and Applications*. Studies in Logic. Elsevier Science Publishers, 2003.

[25] F. Baader and H. J. Ohlbach. "A Multi-Dimensional Terminological Knowledge Representation Language". In: *Journal of Applied Non-Classical Logics* 5.2 (1995).

[26] J. McCarthy and P. J. Hayes. "Some Philosophical Problems from the Standpoint of Aritificial Intelligence". In: *Machine Intelligence* 4 (1969), pp. 463–502.

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

[28] D. C., G. De Giacomo, and M. Soutchanski. "On the Undecidability of the Situation Calculus Extended with Description Logic Ontologies". In: *Proc. of the 24th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. AAAI Press, 2015, pp. 2840–2846.

[29] A. Artale and E. Franconi. "Temporal ER Modeling with Description Logics". In: *Proc. of the 18th Int. Conf. on Conceptual Modeling (ER)*. Vol. 1728. Lecture Notes in Computer Science. Springer, 1999, pp. 81–95.

References VI

[30] V. Gutiérrez-Basulto, J. C. Jung, and C. Lutz. “Complexity of Branching Temporal Description Logics”. In: *Proc. of the 20th Eur. Conf. on Artificial Intelligence (ECAI)*. 2012, pp. 390–395.

[31] W. Jamroga. “Concepts, Agents, and Coalitions in Alternating Time”. In: *Proc. of the 20th Eur. Conf. on Artificial Intelligence (ECAI)*. 2012, pp. 438–443.

[32] F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. “Integrating Description Logics and Action Formalisms: First Results”. In: *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI)*. 2005, pp. 572–577.

[33] Y. Gu and M. Soutchanski. “Decidable Reasoning in a Modified Situation Calculus”. In: *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. 2007, pp. 1891–1897.

[34] Y. Gu and M. Soutchanski. “A Description Logic Based Situation Calculus”. In: *Ann. of Mathematics and Artificial Intelligence* 58.1-2 (2010), pp. 3–83.

[35] M. Milicic. “Action, Time and Space in Description Logics”. PhD thesis. Dresden University of Technology, Germany, 2008.

[36] A. Artale, C. Lutz, and D. Toman. “A Description Logic of Change”. In: *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. 2007, pp. 218–223.

References VII

[37] A. Artale, R. Kontchakov, C. Lutz, F. Wolter, and M. Zakharyaschev. "Temporalising Tractable Description Logics". In: *Proc. of the 14th Int. Symp. on Temporal Representation and Reasoning (TIME)*. 2007, pp. 11–22.

[38] F. Baader, S. Ghilardi, and C. Lutz. "LTL over description logic axioms". In: *ACM Trans. on Computational Logic* 13.3 (2012).

[39] F. Baader and M. Lippmann. "Runtime Verification Using the Temporal Description Logic ALC-LTL Revisited". In: *J. of Applied Logic* 12.4 (2014).

[40] F. Baader, S. Borgwardt, and M. Lippmann. "Temporalizing Ontology-Based Data Access". In: *Proc. of the 24th Int. Conf. on Automated Deduction (CADE)*. 2013.

[41] F. Baader, S. Borgwardt, and M. Lippmann. "Temporal query entailment in the Description Logic SHQ". In: *J. of Web Semantics* 33 (2015).

[42] H. J. Levesque. "Foundations of a Functional Approach to Knowledge Representation". In: *Artificial Intelligence* 23 (1984), pp. 155–212.

References VIII

[43] B. Bagheri Hariri, D. C., G. De Giacomo, A. Deutsch, and M. Montali. "Verification of Relational Data-Centric Dynamic Systems with External Services". In: *Proc. of the 32nd ACM Symp. on Principles of Database Systems (PODS)*. 2013, pp. 163–174.

[44] B. Bagheri Hariri, D. C., M. Montali, G. De Giacomo, R. De Masellis, and P. Felli. "Description Logic Knowledge and Action Bases". In: *J. of Artificial Intelligence Research* 46 (2013), pp. 651–686.

[45] D. C., G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. "EQL-Lite: Effective First-Order Query Processing in Description Logics". In: *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. 2007, pp. 274–279.

[46] C. Beeri and M. Y. Vardi. "The Implication Problem for Data Dependencies". In: *Proc. of the 8th Coll. on Automata, Languages and Programming (ICALP)*. Vol. 115. Lecture Notes in Computer Science. Springer, 1981, pp. 73–85.

[47] D. C., G. De Giacomo, D. Lembo, M. Montali, and A. Santoso. "Ontology-Based Governance of Data-Aware Processes". In: *Proc. of the 6th Int. Conf. on Web Reasoning and Rule Systems (RR)*. Vol. 7497. Lecture Notes in Computer Science. Springer, 2012, pp. 25–41.

References IX

[48] S. Brandt, E. G. Kalayci, R. Kontchakov, V. Ryzhikov, G. Xiao, and M. Zakharyaschev. "Ontology-Based Data Access with a Horn Fragment of Metric Temporal Logic". In: *Proc. of the 31st AAAI Conf. on Artificial Intelligence (AAAI)*. AAAI Press, 2017, pp. 1070–1076.

[49] S. Brandt, E. Güzel Kalayci, V. Ryzhikov, G. Xiao, and M. Zakharyaschev. "Querying Log Data with Metric Temporal Logic". In: *J. of Artificial Intelligence Research* 62 (2018), pp. 829–877.

[50] E. Güzel Kalayci, G. Xiao, V. Ryzhikov, T. E. Kalayci, and D. C. "Ontop-temporal: A Tool for Ontology-based Query Answering over Temporal Data". In: *Proc. of the 27th ACM Int. Conf. on Information and Knowledge Management (CIKM)*. 2018, pp. 1927–1930.