End-User Access to Big Data Using Ontologies

Part 1: Modeling Information Through Ontologies

Diego Calvanese

Free University of Bozen-Bolzano

unibz
Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

International Winter School on Big Data

Tarragona, Spain, January 26–30 , 2015

Part 1

Modeling Information Through Ontologies

unibz

Outline of Part 1

unibz

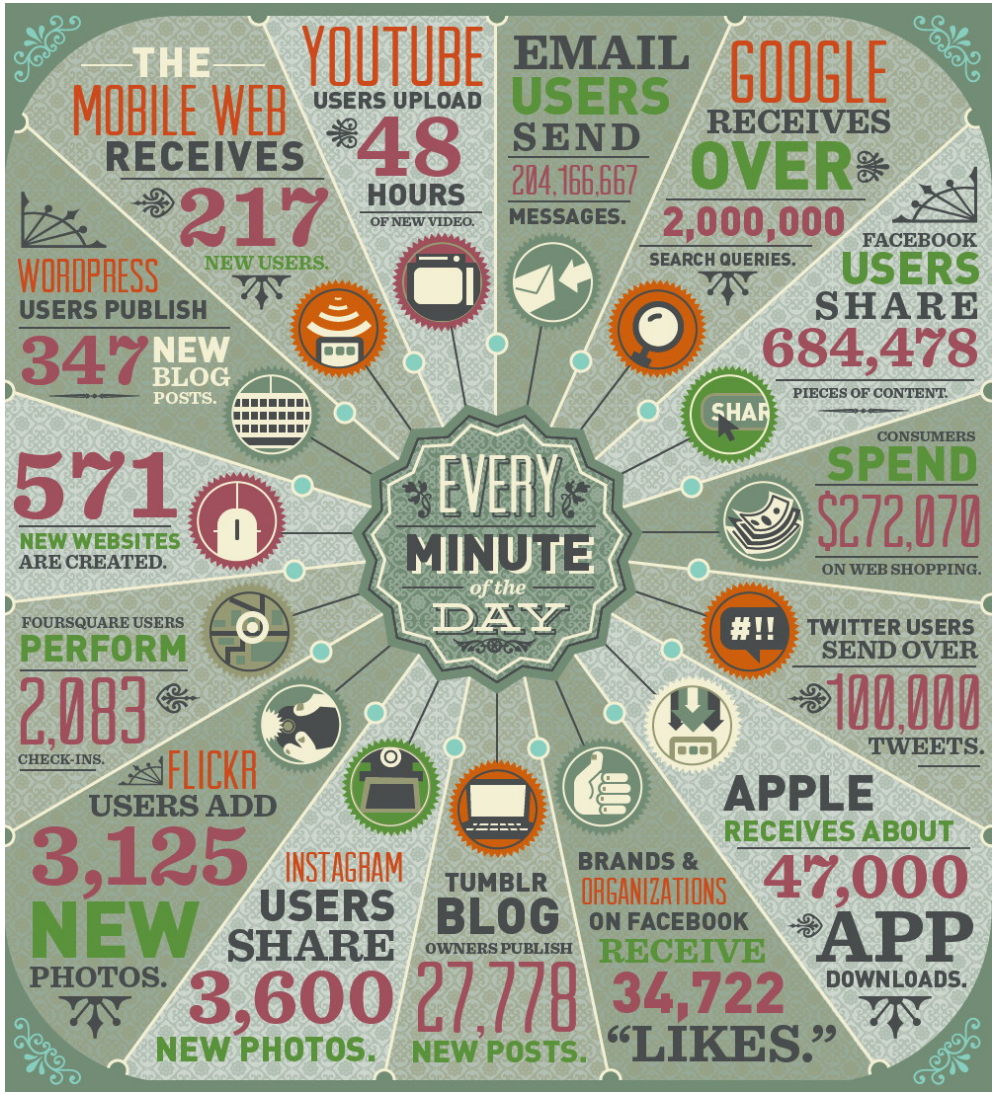
Outline of Part 1

unibz

Outline of Part 1

unibz

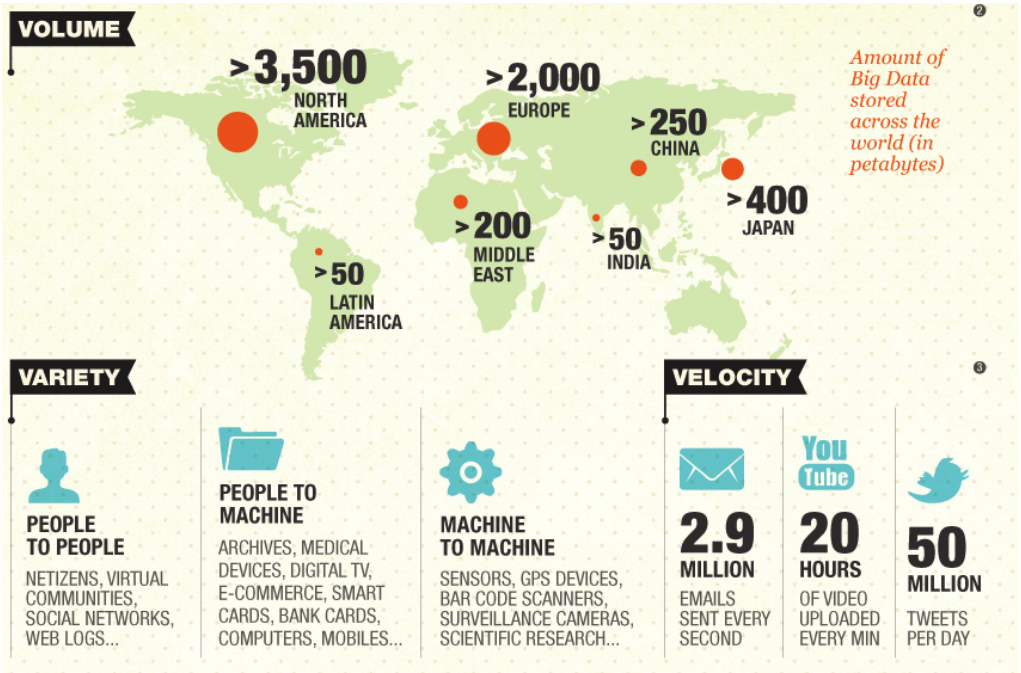
Data never sleeps



unibz

Big data

As firms move from siloed, transaction-oriented systems to more integrated, socially aware ones, they will face challenges related to customer data.
Big data is characterized by increases in data **volume**, **velocity**, **variety**, and **variability**. To improve customer engagement, companies must invest in solutions to effectively manage big data.
[Forrester Research, Inc. June 1, 2012]



unibz

New challenges in information management

Information management is a key challenge in complex systems today:

- The **volume** of information to manage is enormous.
- Data increases with incredible **velocity**.
- The **variety** of information has increased:
  - structured vs. semi-structured vs. unstructured
  - data is distributed and heterogeneous
  - human-processable vs. machine processable
- The meaning of data is **variable**, and depends on the context.
- The **veracity** of the data needs to be questioned and assessed – incompleteness, inconsistency, lack of precision.
- To understand complex data it needs to be **visualized**.
- Data increasingly represents an important **value** for an organization.

There is an increased need to access data in a uniform and integrated way, extract information, and perform various forms of analysis on it.

Traditional data management systems are not sufficient anymore to fulfill today's information management requirements.

Example 1: Statoil Exploration

Experts in geology and geophysics develop stratigraphic models of unexplored areas on the basis of data acquired from previous operations at nearby geographical locations.

Facts:

- 1,000 TB of relational data
- using diverse schemata
- spread over 2,000 tables, over multiple individual data bases

Data Access for Exploration:

- 900 experts in Statoil Exploration.
- up to 4 days for new data access queries, requiring assistance from IT-experts.
- 30–70% of time spent on data gathering.

unibz

Example 2: Siemens Energy Services

Runs service centers for power plants, each responsible for remote monitoring and diagnostics of many thousands of gas/steam turbines and associated components. When informed about potential problems, diagnosis engineers access a variety of raw and processed data.

Facts:

- several TB of time-stamped sensor data
- several GB of event data ("alarm triggered at time T")
- data grows at 30GB per day (sensor data rate 1Hz–1kHz)

Service Requests:

- over 50 service centers worldwide
- 1,000 service requests per center per year
- 80% of time per request used on data gathering

unibz

Addressing information management challenges

Several efforts come from the area of **databases**:

- New kinds of databases are studied:
  - XML databases, graph databases
  - column stores
  - probabilistic databases
  - ...
- Information integration
  - Represents one of the major challenges for the future or IT.
  - E.g., the market for information integration software has been growing at a steady rate of +9% per year since 2007.
  - The overall market value of such software was \$ 4 billion in 2012.

unibz

The role of Knowledge Representation in AI

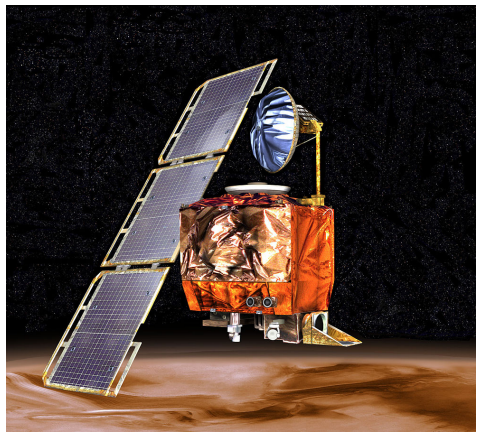
Management of complex kinds of information has traditionally been the concern of **Knowledge Representation** (KR) in AI:

- Research in AI and KR can bring important insights, solutions, techniques, and technologies: concern on **variability** and **veracity**
- However, the other v's have not received the proper attention so far: **volume**, **velocity**, **variety**.

The techniques and tools developed in KR need to be **adapted** and **extended** to address the new challenges in Big Data.

Emphasis is on the **semantics** of data!

- Fundamental for understanding, sharing, and reasoning.
- Example: Mars climate orbiter case in 1999: 327.6M $ lost because of a metric mixup: same data, different interpretations!



Mars Climate Orbiter 2 by NASA, JPL, Corby Waste.

Ontologies

Ontologies in Computer Science

An **ontology** is a representation scheme describing a **formal conceptualization** of a domain of interest.

The specification of an ontology usually comprises two distinct levels:

- **Intensional level**: specifies a set of conceptual elements and of constraints/axioms describing the conceptual structures of the domain.
- **Extensional level**: specifies a set of instances of the conceptual elements described at the intensional level.

Note: an ontology may contain also a **meta-level**, which specifies a set of modeling categories of which the conceptual elements are instances.

unibz

Description logic ontologies

The formal foundations for ontology languages are in logic, and specifically in description logics.

Description logics [Baader *et al.*, 2003] are fragments of first-order logic specifically tailored towards the representation of structured knowledge.

- By grounding the used formalisms in logic, the information is provided with a **formal semantics** (i.e., a meaning).
- The logic-based formalization allows one to provide **automated support** for tasks related to data management, by means of **logic-based inference**.
- **Computational aspects** are of concern, so that **tools** can provide **effective support** for automated reasoning.

In this course:

We will consider specific description logics suitable for data management.

Outline of Part 1

unibz

Conceptual schemas in information systems

Intensional information has traditionally played an important role in information systems.

Design phase of the information system:

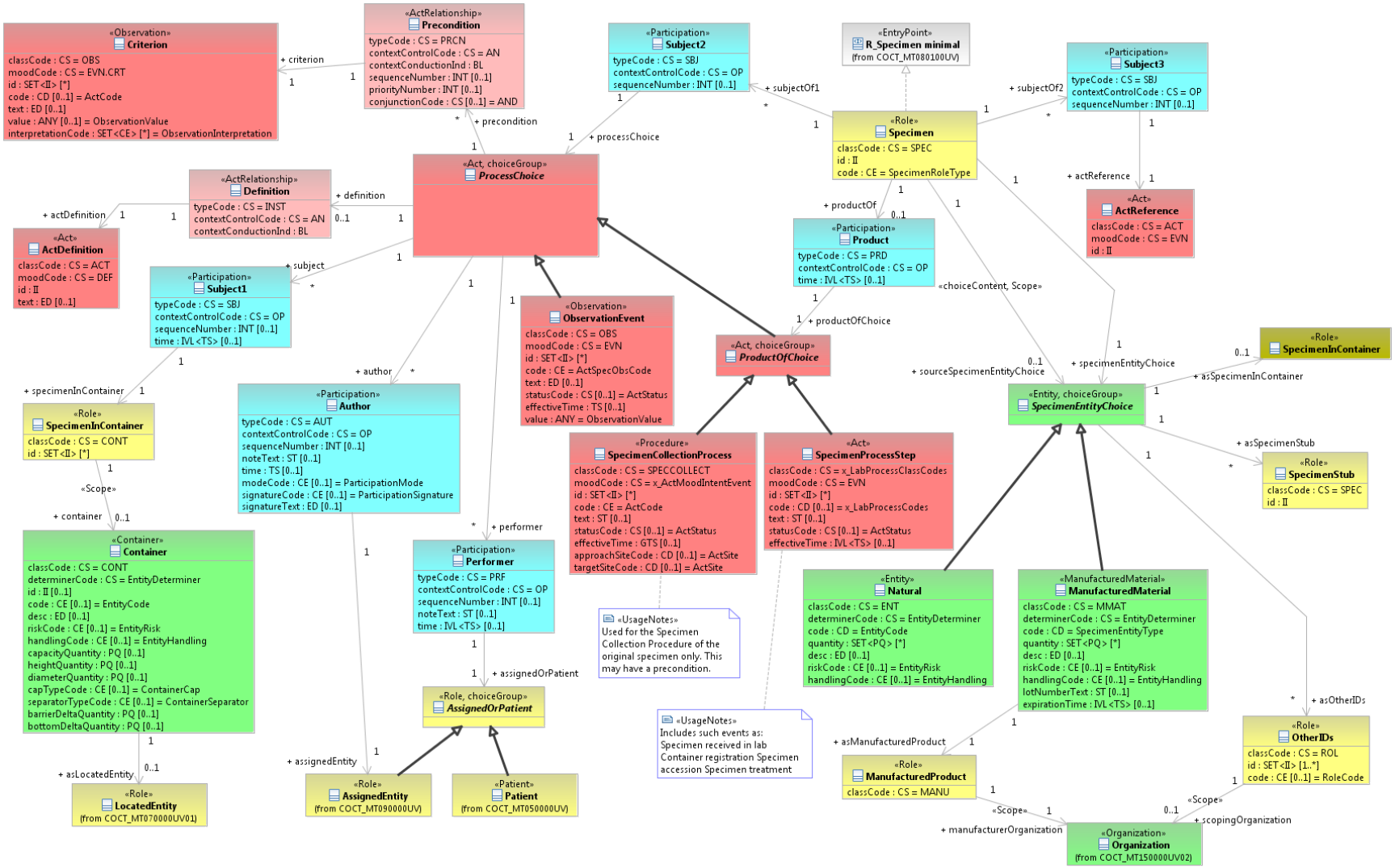
1. **Conceptual modeling**: from the requirements, a **conceptual schema** of the domain of interest is produced.
2. The conceptual schema is used to produce the logical data schema.
3. The data are stored according to the logical schema, and queried through it.

Conceptual Modeling

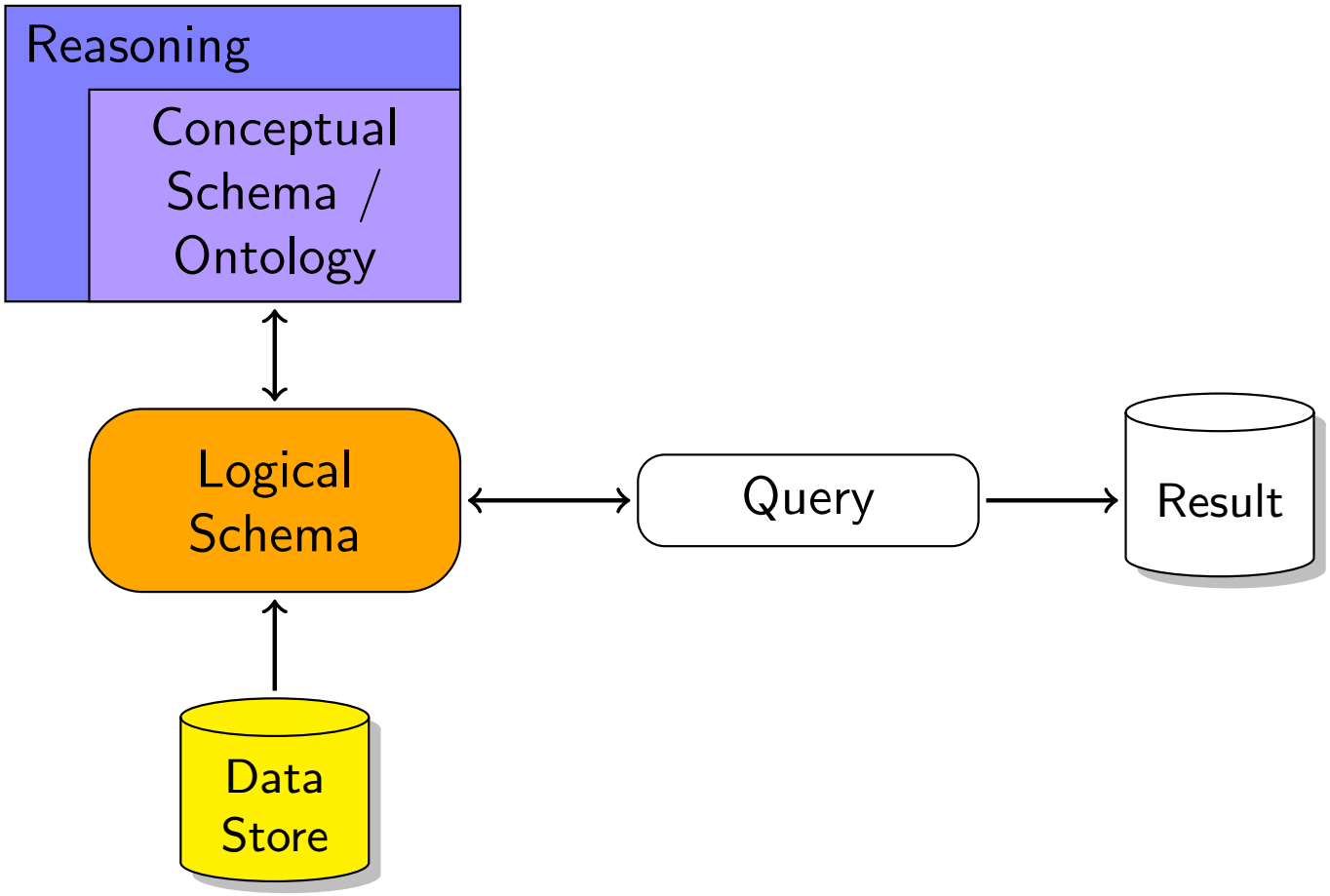
The activity of formally describing some aspects of the physical and social world around us for the purposes of understanding and communication.
[John Mylopoulos]

unibz

Managing complexity ...



Conceptual schemas used at design-time



Ontologies in information systems

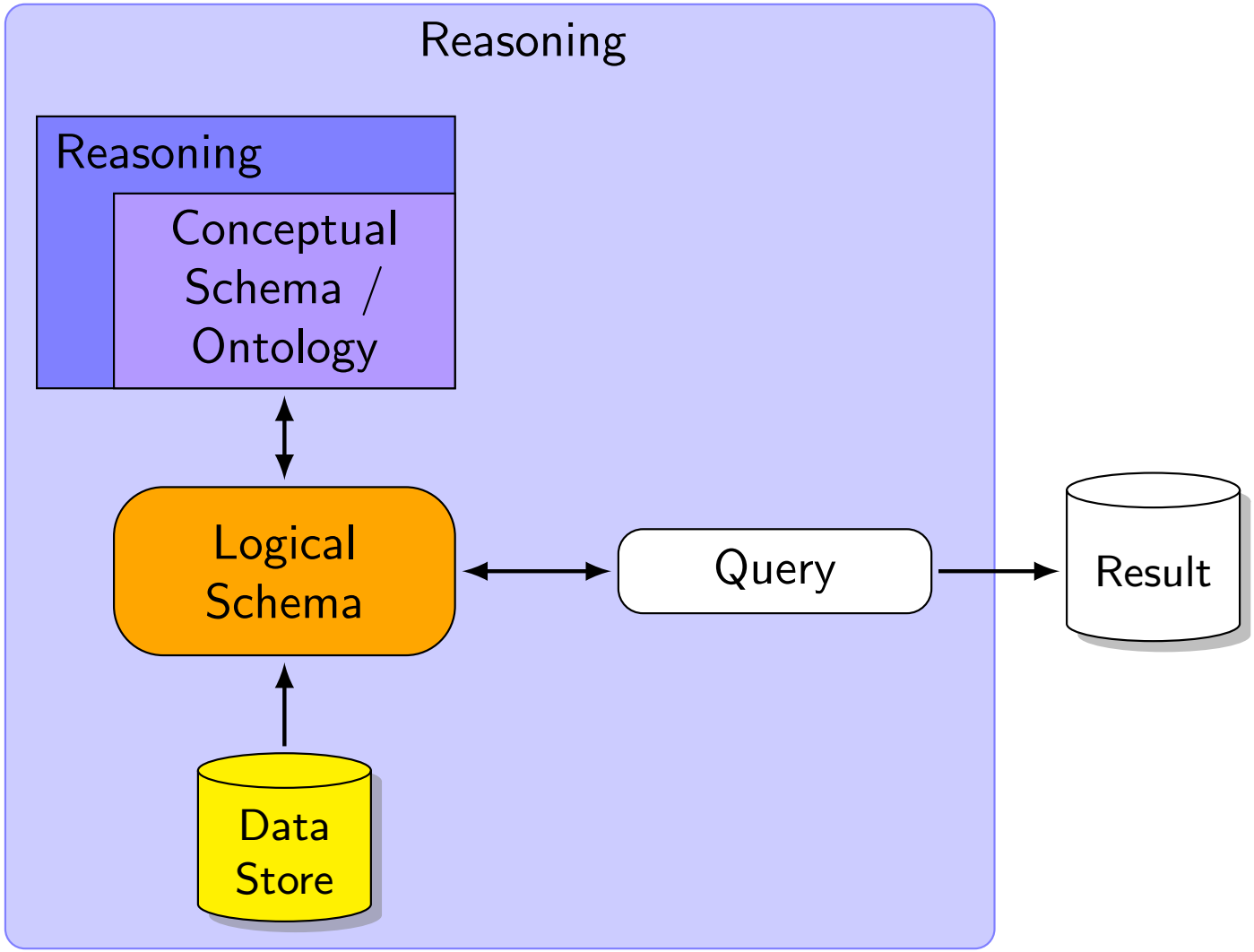
The role of ontologies in information systems goes beyond that of conceptual schemas.

Ontologies affect the whole life-cycle of the information system:

- Ontologies, with the associated reasoning capabilities and inference tools, can provide **support at design time**.
- The use of ontologies can significantly simplify **maintenance** of the information system's data assets.
- The ontology is used also to **support the interaction** with the information system, i.e., **at run-time**.
  $\rightsquigarrow$ **Reasoning** to take into account the constraints coming from the ontology has to be **done at run-time**.

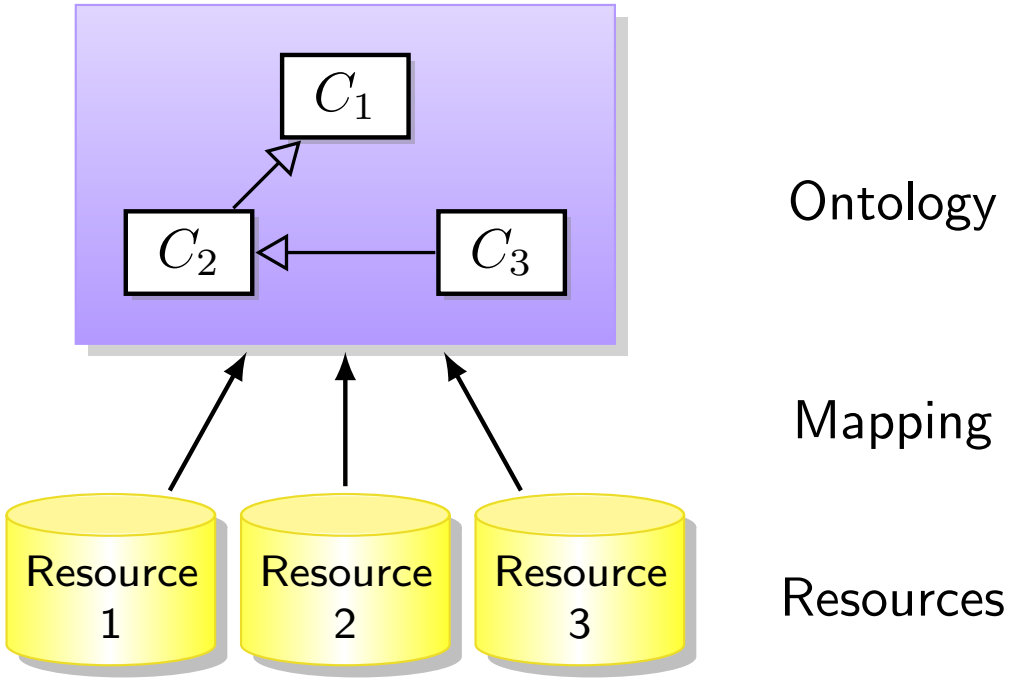
unibz

Ontologies used at run-time



unibz

Ontologies at the core of information systems



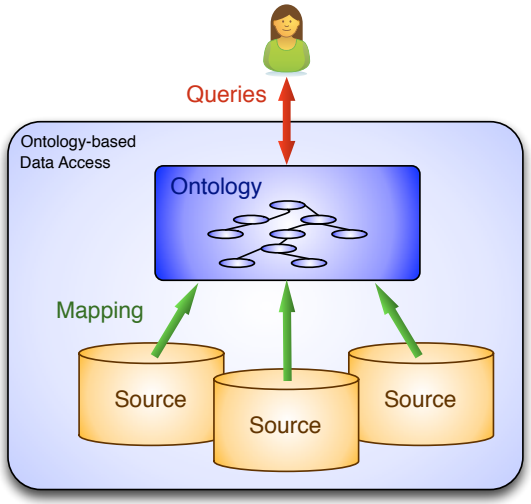
The usage of all system resources (data and services) is done through the domain conceptualization.

unibz

Ontology based data access

Manage data by adopting principles and techniques studied in **Knowledge Representation**.

- Provide a conceptual, high level representation of the domain of interest in terms of an **ontology**.
- Do **not migrate the data** but leave it in the sources.
- **Map** the ontology to the data sources.
- Specify all information requests to the data in terms of the ontology.
- Use the inference services of the OBDA system to **translate the requests** into queries to the data sources.



The OBDA approach is based on **formalisms grounded in logic**, with well understood semantics and computational properties.

Outline of Part 1

unibz

Issues in ontology-based information management

1. Choice of the formalisms to adopt
2. Efficiency and scalability
3. Tool support

unibz

Issue 1: Formalisms to adopt

1. Which is the right ontology language?
   - many proposals have been made
   - differ in expressive power and in complexity of inference
2. Which languages should we use for querying?
   - requirements for querying are different from those for modeling
3. How do we connect the ontology to available data sources?
   - mismatch between information in an ontology and data in a data source

In this course:

- We present and discuss variants of ontology languages suited for **ontology based data management**, and study their logical and computational properties.
- We study the problem of querying data through ontologies.
- We discuss problems and solutions related to the impedance mismatch between ontologies and data sources.

Issue 2: Efficiency and scalability

- How can we handle large ontologies?
  - We have to take into account the **tradeoff** between **expressive power** and **complexity** of inference.
- How can we cope with large amounts of data?
  - What may be good for large ontologies, may not be good enough for large amounts of data.
- Can we handle multiple data sources and/or multiple ontologies?

In this course:

- We discuss in depth the above mentioned **tradeoff**.
- We pay attention to the aspects related to **data management**.
- We do not deal with the problem of integrating multiple information sources. This is typically addressed in *Information Integration*, to which many of the considerations we make also apply.

Issue 3: Tools

- According to the principle that "there is no meaning without a language with a formal semantics", the formal semantics becomes the solid basis for dealing with ontologies.
- Hence every kind of access to an ontology (to extract information, to modify it, etc.), requires to **fully** take into account its semantics.
- We need tools that perform reasoning over the ontology that is **sound and complete** wrt the semantics.
- The tools have to be as "efficient" as possible.

In this course:

- We discuss the requirements, the principles, and the theoretical foundations for ontology inference tools.
- We also briefly present the *ontop* tool for querying data sources through ontologies, which has been built according to those principles.

Outline of Part 1

unibz

Outline of Part 1

unibz

Elements of an ontology language

- **Syntax**
  - Alphabet
  - Languages constructs
  - Sentences to assert knowledge
- **Semantics**
  - Formal meaning
- **Pragmatics**
  - Intended meaning
  - Usage

unibz

Static vs. dynamic aspects

Two broad categories of what we want to model for a given domain of interest:

- **Static aspects**
  - Are related to the structuring of the domain of interest.
  - Supported by virtually all languages.
- Dynamic aspects
  - Are related to how the elements of the domain of interest evolve over time.
  - Supported only by some languages, and only partially (cf. services).

Before delving into the dynamic aspects, we need a good understanding of the static ones.

In this course:

We concentrate on the static aspects only.

unibz

Intensional level of an ontology language

Most important elements of an ontology language at the **intensional level**:

- Concepts (aka, classes, entity types): denote sets of instances/individuals
- Relationships (aka, associations, roles, object properties): express a relation between concept instances
- Properties (aka, attributes, features, data properties): qualify a concept or relationship
- Axioms (aka, assertions): express at the intensional level conditions that need to be satisfied at the extensional level

Ontologies are typically **rendered as diagrams** (e.g., Semantic Networks, Entity-Relationship schemas, UML Class Diagrams).

unibz

Extensional level of an ontology language

At the **extensional level** we have individuals (or objects), and we represent the concepts to which they belong, and the relationships in which they participate:

- An **instance** represents an individual in the extension of a concept.
  E.g., Researcher(martin) represents that martin is an instance of Researcher.
- A **fact** represents a relationship holding between instances.
  E.g., worksFor(martin, optique) represents that martin works for optique.

unibz

Outline of Part 1

unibz

Representing knowledge in Description Logics

- **Description Logics (DLs)** stem from early days (1970s) KR formalisms, and assumed their current form in the late 1980s & 1990s.
- Are logics designed to represent and reason on **structured knowledge**.
- Most DLs can be considered as computationally well-behaved **fragments of first-order logic**.
- Semantics given in terms of **first-order interpretations**.
- Tightly connected to **modal logics**: many DLs are syntactic variants of modal logics.
- Come in hundreds of variations, with different semantic and computational properties.
- Have strongly influenced the W3C standard Web Ontology Language **OWL**.

unibz

Description Logic ontology (or knowledge base)

In DLs, the domain of interest is represented by means of:

- concepts: unary predicates Ex.: Actor, Director, Movie, ...
- roles: binary predicates Ex.: playsIn, directs, ...

Complex concept and role expressions obtained using DL specific constructors.

A **DL ontology** is a pair $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$:

The **TBox** $\mathcal{T}$ represents intensional level information via a set of axioms:

- Concept inclusions: $C_1 \sqsubseteq C_2$, interpreted as $\forall x.C_1(x) \rightarrow C_2(x)$.
- Role inclusions: $R_1 \sqsubseteq R_2$, interpreted as $\forall x, y.R_1(x, y) \rightarrow R_2(x, y)$.
- Role properties: (**transitive** P), (**symmetric** P), ...

The **ABox** $\mathcal{A}$ represents information about individuals via a set of facts:

- Concept membership assertions: $A(d)$
- Role membership assertions: $P(d_1, d_2)$

Note: an ABox is conveniently represented as an edge and node labeled graph.

Traditional DL reasoning services

- Ontology satisfiability: Does $\mathcal{O}$ admit a model?
- Concept satisfiability w.r.t. an ontology: Is there a model $\mathcal{I}$ of $\mathcal{O}$ such that $C^{\mathcal{I}}$ is not empty?
- Concept subsumption w.r.t. an ontology: Does $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ hold, for every model $\mathcal{I}$ of $\mathcal{O}$?

Subsumption is at the basis of classification, i.e., determining the hierarchy of concepts of an ontology.

We are concerned here with a more complicated form of reasoning, namely **query answering**.

Note: A simple form of query answering is instance checking, where the query is simply a concept (instance query): Does $d^{\mathcal{I}} \in C^{\mathcal{I}}$ hold in every model $\mathcal{I}$ of $\mathcal{O}$?

unibz

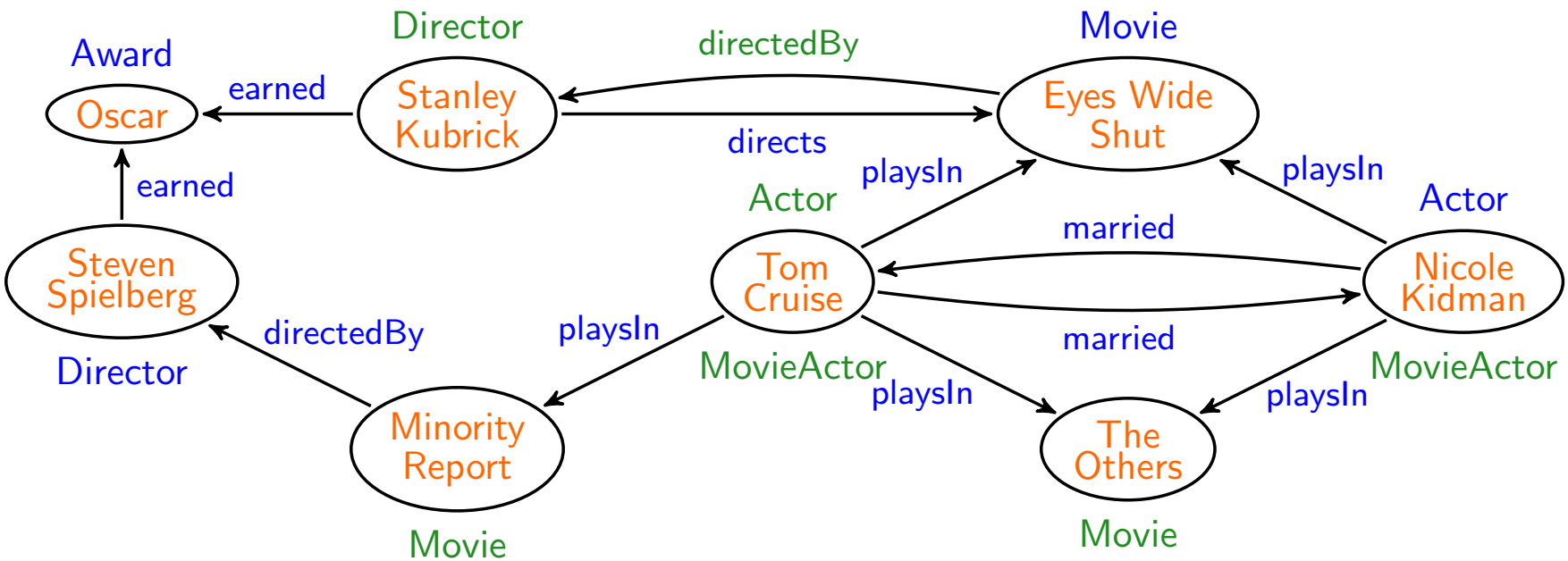
Example TBox and ABox — IMDB

TBox $\mathcal{T}_m$:

$$\exists \text{playsIn}.\text{Movie} \sqsubseteq \text{MovieActor} \qquad \exists \text{directs}.\top \sqsubseteq \text{Director}$$
$$\exists \text{playsIn}.\text{Series} \sqsubseteq \text{SeriesActor} \qquad \text{Movie} \sqsubseteq \exists \text{directedBy}.\top$$
$$\text{Actor} \equiv \text{SeriesActor} \sqcup \text{MovieActor} \qquad \text{directs} \equiv \text{directedBy}^{-}$$
$$\text{MovieActor} \sqsubseteq \forall \text{playsIn}.\text{Movie} \qquad \ldots$$

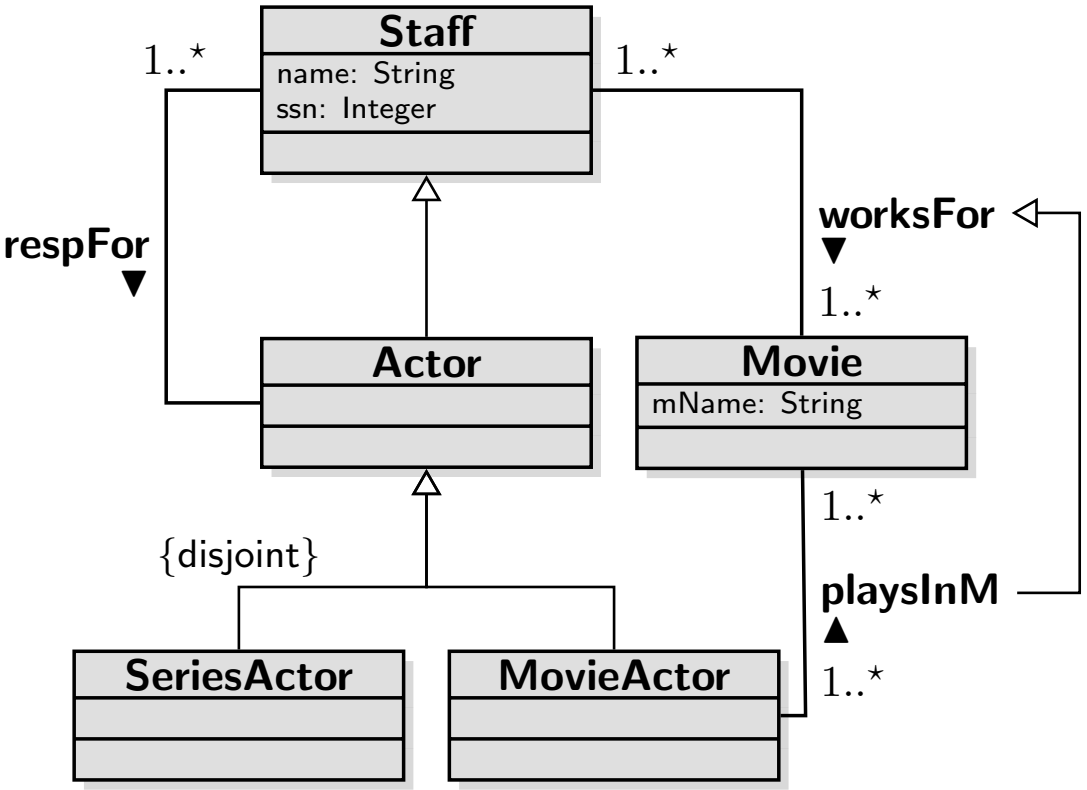
Note: we use $C_1 \equiv C_2$ as an abbreviation for $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$.

ABox $\mathcal{A}_m$:



Ontologies vs. conceptual models

We want to leverage on an extensive amount of work on the tight relationship between conceptual modeling formalisms and ontology languages [Lenzerini and Nobili, 1990; Bergamaschi and Sartori, 1992; Borgida, 1995; Calvanese *et al.*, 1999; Borgida and Brachman, 2003; Berardi *et al.*, 2005; Queralt *et al.*, 2012].



$$
\begin{aligned}
\mathsf{Actor} &\sqsubseteq \mathsf{Staff}\\
\mathsf{SeriesActor} &\sqsubseteq \mathsf{Actor}\\
\mathsf{MovieActor} &\sqsubseteq \mathsf{Actor}\\
\mathsf{SeriesActor} &\sqsubseteq \neg\mathsf{MovieActor}\\
\\
\mathsf{Staff} &\sqsubseteq \exists \mathit{ssn}\\
\exists \mathit{ssn}^- &\sqsubseteq \mathtt{xsd{:}int}\\
&(\textbf{funct}\ \mathit{ssn})\\
\\
\exists \mathsf{playsInM} &\sqsubseteq \mathsf{MovieActor}\\
\exists \mathsf{playsInM}^- &\sqsubseteq \mathsf{Movie}\\
\mathsf{MovieActor} &\sqsubseteq \exists \mathsf{playsInM}\\
\mathsf{Movie} &\sqsubseteq \exists \mathsf{playsInM}^-\\
\mathsf{playsInM} &\sqsubseteq \mathsf{worksFor}\\
&\ldots
\end{aligned}
$$

unibz

Outline of Part 1

unibz

Outline of Part 1

unibz

The Unified Modeling Language (UML)

The **Unified Modeling Language (UML)** was developed in 1994 by unifying and integrating the most prominent object-oriented modeling approaches:

- Booch
- Rumbaugh: Object Modeling Technique (OMT)
- Jacobson: Object-Oriented Software Engineering (OOSE)

History:

- 1995, version 0.8, Booch, Rumbaugh; 1996, version 0.9, Booch, Rumbaugh, Jacobson; version 1.0 BRJ + Digital, IBM, HP, ...
- UML 1.4.2 is industrial standard ISO/IEC 19501.
- Current version: 2.4.1 (Aug. 2011): `http://www.omg.org/spec/UML/`
- 1999–today: de facto standard object-oriented modeling language.

References:

- Grady Booch, James Rumbaugh, Ivar Jacobson, "The unified modeling language user guide", Addison Wesley, 1999 (2nd ed., 2005)
- `http://www.omg.org/` → UML
- `http://www.uml.org/`

unibz

UML Class Diagrams

Here we deal only with one of the most prominent components of UML: **UML Class Diagrams**.

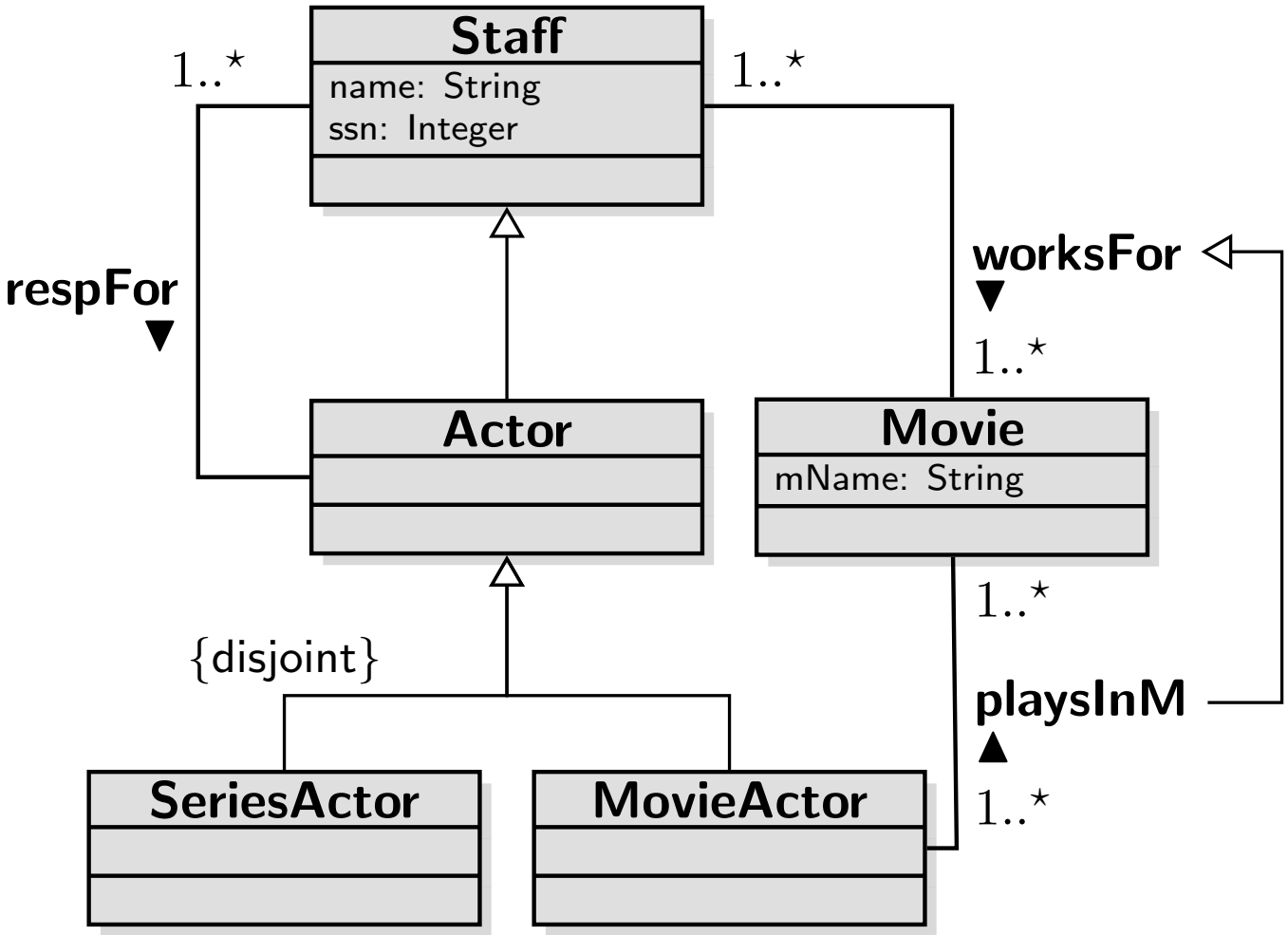
A UML Class Diagram is used to **represent explicitly** the information on **a domain of interest** in terms of:

- objects grouped into classes;
- associations, representing relationships between classes;
- attributes, representing simple properties of the instances of classes;
  Note: here we do not deal with "operations".
- sub-classing, i.e., ISA and generalization relationships.

Note: This is exactly the goal of all conceptual modeling formalism, such as Entity-Relationship diagrams (standard in database design), and of ontologies.

unibz

Example of a UML Class Diagram



Use of UML Class Diagrams

UML CDs are used in various phases of a software design:

1. During the so-called analysis, where an abstract precise view of the domain of interest needs to be developed.
   $\rightsquigarrow$ the so-called "**conceptual perspective**".
2. During software development, to maintain an abstract view of the software to be developed.
   $\rightsquigarrow$ the so-called "implementation perspective".

In this course we focus on 1!

unibz

Outline of Part 1

unibz

Conceptual modeling exercise

We aim at obtaining a description of the data of interest in **semantic terms**.

Let's do a simple exercise!

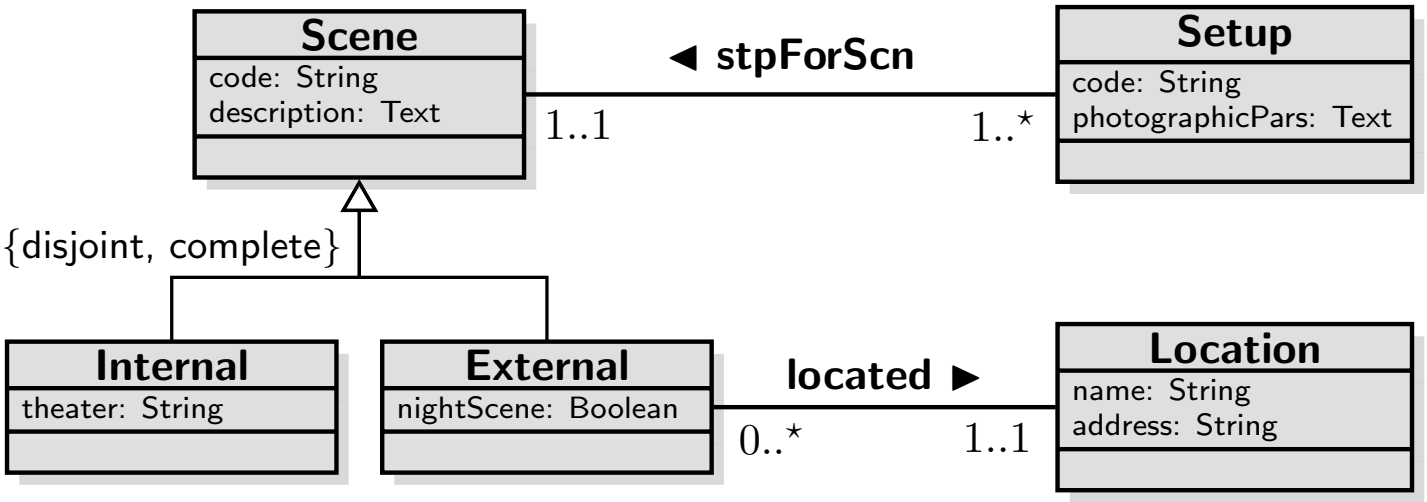
Requirements

We are interested in building a software application to manage filmed scenes for realizing a movie, by following the so-called "Hollywood Approach".

- Every scene is identified by a code (a string) and is described by a English text.
- Scenes are divided into internals, which are filmed in a theater, and externals, which are filmed in a location and can either be "day scene" or "night scene". Locations are characterized by a code (a string) and the address of the location.
- Every scene is filmed from different positions (at least one), each of these is called a setup. Every setup is characterized by a code (a string) and a text in natural language where the photographic parameters are noted (e.g., aperture, exposure, focal length, filters, etc.). Note that a setup is related to a single scene.

Write a precise specification of this domain using any formalism you like!

Solution 1: Use conceptual modeling diagrams (UML)!



unibz

Solution 1: Use conceptual modeling diagrams – Discussion

Good points:

- Easy to generate (it's the standard in software design).
- Easy to understand for humans.
- Well disciplined, well-established methodologies available.

Bad points:

- No precise semantics (people that use it wave hands about it).
- Verification (or better validation) done informally by humans.
- Machine incomprehensible (because of lack of formal semantics).
- Automated reasoning and query answering out of question.
- Limited expressiveness $^{(*)}$.

$^{(*)}$ *Not really a bad point, in fact.*

unibz

Solution 2: Use logic!!

Alphabet: $Scene(x)$, $Setup(x)$, $Take(x)$, $Internal(x)$, $External(x)$, $Location(x)$, $stpForScn(x, y)$, $tkOfStp(x, y)$, $located(x, y)$, ...

Axioms:

$\forall x, y.\, code_{Scene}(x, y) \rightarrow Scene(x) \wedge String(y)$
$\forall x, y.\, description(x, y) \rightarrow Scene(x) \wedge Text(y)$

$\forall x, y.\, code_{Setup}(x, y) \rightarrow Setup(x) \wedge String(y)$
$\forall x, y.\, photographicPars(x, y) \rightarrow Setup(x) \wedge Text(y)$

$\forall x, y.\, theater(x, y) \rightarrow Internal(x) \wedge String(y)$

$\forall x, y.\, nightScene(x, y) \rightarrow External(x) \wedge Boolean(y)$

$\forall x, y.\, name(x, y) \rightarrow Location(x) \wedge String(y)$
$\forall x, y.\, address(x, y) \rightarrow Location(x) \wedge String(y)$

$\forall x.\, Scene(x) \rightarrow (1 \leq \sharp\{y \mid code_{Scene}(x, y)\} \leq 1)$

$\forall x.\, Internal(x) \rightarrow Scene(x)$
$\forall x.\, External(x) \rightarrow Scene(x)$
$\forall x.\, Internal(x) \rightarrow \neg External(x)$
$\forall x.\, Scene(x) \rightarrow Internal(x) \vee External(x)$

$\forall x, y.\, stpForScn(x, y) \rightarrow Setup(x) \wedge Scene(y)$
$\forall x, y.\, located(x, y) \rightarrow External(x) \wedge Location(y)$

$\forall x.\, Setup(x) \rightarrow (1 \leq \sharp\{y \mid stpForScn(x, y)\} \leq 1)$
$\forall y.\, Scene(y) \rightarrow (1 \leq \sharp\{x \mid stpForScn(x, y))\}$
$\forall x.\, External(x) \rightarrow (1 \leq \sharp\{y \mid located(x, y)\} \leq 1)$
...

unibz

Solution 2: Use logic – Discussion

Good points:

- Precise semantics.
- Formal verification.
- Allows for query answering.
- Machine comprehensible.
- Virtually unlimited expressiveness $^{(*)}$.

Bad points:

- Difficult to generate.
- Difficult to understand for humans.
- Too unstructured (making reasoning difficult), no well-established methodologies available.
- Automated reasoning may be impossible.

$^{(*)}$ *Not really a good point, in fact.*

unibz

Solution 3: Use both!!!

Note: these two approaches seem to be orthogonal, but in fact they can be used together cooperatively.

Basic idea:

- Assign formal semantics to constructs of the conceptual design diagrams.
- Use conceptual design diagrams as usual, taking advantage of methodologies developed for them in Software Engineering.
- Read diagrams as logical theories when needed, i.e., for formal understanding, verification, automated reasoning, etc.

Added values:

- Inherited from conceptual modeling diagrams: ease-to-use for humans
- Inherited from logic: formal semantics and reasoning tasks, which are needed for formal verification and machine manipulation.

unibz

Solution 3: Use both!!! (cont'd)

Important:

The logical theories that are obtained from conceptual modeling diagrams are of a specific form.

- Their expressiveness is limited (or better, well-disciplined).
- One can exploit the particular form of the logical theory to simplify reasoning.
- The aim is getting:
  - decidability, and
  - reasoning procedures that match the intrinsic computational complexity of reasoning over the conceptual modeling diagrams.

unibz

Reasoning on UML Class Diagrams via FOL reasoning

There are several properties that are of interest for a UML Class Diagrams, and that can be phrased as forms of inference on the diagram:

- Consistency of the whole diagram.
- Consistency of classes and associations, to avoid maintaining in the diagram useless information.
- Subsumption of classes (or associations), to detect when properties of a more general class are inherited by a more specific class.
- Equivalence of classes (or associations), to detect and eliminate redundancy in the diagram.
- Implication of properties, to make implicit information explicit.

One can formally define the above reasoning tasks on UML Class Diagrams, and show how they can be recast in terms of FOL reasoning.

unibz

Logic based conceptual modeling

Summing up, to use logic as a conceptual modeling tool, we proceed as follows:

1. Represent the domain of interest as a **conceptual schema**, similar to what done during database design.
2. Formalize the conceptual schema as a **logical theory**, namely the **ontology**.
3. Use the resulting logical theory for **reasoning** and **query answering**.

unibz

Questions

Two questions come up.

1. Can we develop sound, complete, and **terminating** procedures for reasoning on UML CDs?
   - We cannot do so by directly relying on FOL!
   - But we can use specialized logics with better computational properties. A form of such specialized logics are **Description Logics**.

2. How hard is it to reason on UML CDs in general?
   - What is the worst-case situation?
   - Can we single out **interesting fragments** on which to reason efficiently?

We will address also **answering queries** over such diagrams, which is in general a more complicated task than satisfiability or subsumption.

unibz

References I

[Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors.
The Description Logic Handbook: Theory, Implementation and Applications.
Cambridge University Press, 2003.

[Berardi *et al.*, 2005] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo.
Reasoning on UML class diagrams.
Artificial Intelligence, 168(1–2):70–118, 2005.

[Bergamaschi and Sartori, 1992] Sonia Bergamaschi and Claudio Sartori.
On taxonomic reasoning in conceptual design.
ACM Trans. on Database Systems, 17(3):385–422, 1992.

[Borgida and Brachman, 2003] Alexander Borgida and Ronald J. Brachman.
Conceptual modeling with description logics.
In Baader et al. [2003], chapter 10, pages 349–372.

[Borgida, 1995] Alexander Borgida.
Description logics in data management.
IEEE Trans. on Knowledge and Data Engineering, 7(5):671–682, 1995.

unibz

References II

[Calvanese *et al.*, 1999] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi.
Unifying class-based representation formalisms.
J. of Artificial Intelligence Research, 11:199–240, 1999.

[Lenzerini and Nobili, 1990] Maurizio Lenzerini and Paolo Nobili.
On the satisfiability of dependency constraints in entity-relationship schemata.
Information Systems, 15(4):453–461, 1990.

[Queralt *et al.*, 2012] Anna Queralt, Alessandro Artale, Diego Calvanese, and Ernest Teniente.
OCL-Lite: Finite reasoning on UML/OCL conceptual schemas.
Data and Knowledge Engineering, 73:1–22, 2012.

unibz