Employing logical reasoning to enhance context-aware applications

Anni-Yasmin Turhan

Institute of Theoretical Computer Science,
TU Dresden

Context-aware applications

What are context-aware systems?

- Systems, that can react to the situation of their user
- They usually work . . .
  - autonomous,
  - pro-active
  - personalised, and
  - flexible
- For instance, mobile devices, intelligent home, and vehicles can be context-aware systems

Information sources of context-aware systems

Context-aware systems have to gather situation information constantly.

Where does the situation information come from?

From various data sources, for instance from

- sensor data,
- data bases,
- user preferences,
- other applications

What is the nature of situation information?

- highly dynamic
- heterogeneous
- incomplete
- given on varying levels of detail
- hardly ever explicit

Tasks of context-aware systems

Recognize the user's current context and invoke the right action!

To be able to do this a context-aware system has to:

- gather information
- detect wrong information
- handle missing or faulty information robustly and gracefully
- represent the user's situation
  (derive facts about the user)

From the represented situation of the user recognise the context.

Overview of the rest of the talk

1. Description Logics primer

2. Intelligent door application

3. Query answering

4. Avatr application

5. Conclusions and outlook

Description Logics

- Description Logics:
  family of different logics with varying expressivity
- formalism for declarative description of facts
- have well-defined formal semantics
- powerful reasoning services based on the semantics:
  make implicit knowledge explicit
- are applied in many practical areas
- basis for the web ontology language OWL

The Description Logic $\mathcal{ALC}$

Atomic types: concept names $A, B, \ldots$ (unary predicates)

role names $R, S, \ldots$ (binary predicates)

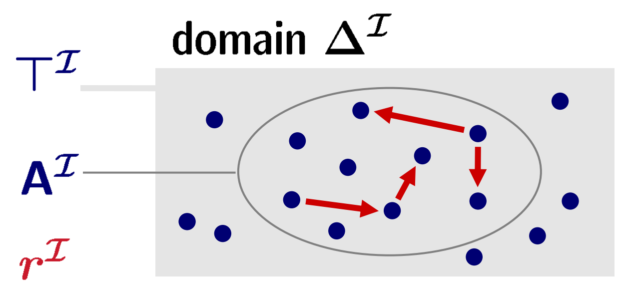
Constructors:

$\neg C$	(negation)		
$C \sqcap D$	(conjunction)	$\exists R.C$	(existential restriction)
$C \sqcup D$	(disjunction)	$\forall R.C$	(value restriction)

Semantics based on interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$r.Grey

Concepts: Subsets of domain $\Delta^{\mathcal{I}}$

Roles: binary relations on domain $\Delta^{\mathcal{I}}$



Complex $\mathcal{ALC}$-Concepts: Semantics

Semantics of complex concepts:

$$(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}$$

$$(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$$

$$(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$$

$$(\exists R.C)^\mathcal{I} = \{d \mid \text{there is an } e \in \Delta^\mathcal{I} \text{ with } (d,e) \in R^\mathcal{I} \text{ and } e \in C^\mathcal{I}\}$$

$$(\forall R.C)^\mathcal{I} = \{d \mid \text{for all } e \in \Delta^\mathcal{I}, (d,e) \in R^\mathcal{I} \text{ implies } e \in C^\mathcal{I}\}$$

TBox

TBoxes capture:

- categories from the application domain
- relations from the application domain in terms of roles
- relate concepts: state super- / sub-concept relationships

TBoxes statements:

- Concept disjointness axiom

  $C \sqcap D \sqsubseteq \bot$
- General concept equivalence

  $C \equiv D$
- General concept (inclusion) axiom (GCI)

  $C \sqsubseteq D$

$$\mathcal{T} = \{ \text{WildAnimal} \sqsubseteq \text{Animal} \sqcap \neg\exists\text{owner}.\top, \quad \text{Mammal} \sqcap \exists\text{bodypart}.\text{Hunch} \equiv \text{Camel} \sqcup \text{Dromedary} \}$$

TU Dresden

TBox reasoning

TBox consistency:

:= Given a TBox $\mathcal{T}$ and a concept description C:
Is there a contradiction in C w.r.t. $\mathcal{T}$ in all cases?

Subsumption:

:= Given a TBox $\mathcal{T}$ and two concept descriptions C and D:
Is C always a sub-concept of D w.r.t. $\mathcal{T}$?

TBox classification:

:= Given a TBox $\mathcal{T}$.
Compute the subsumption hierarchy for all named concepts.

ABox

ABoxes capture:

- facts from the application domain
- knowledge about individuals
- knowledge about the relations between individuals

ABox assertions in DL systems are:

- Concept assertions: $C(a)$
- Role assertions: $(a, b)R$

$\mathcal{A} = \{$ Mammal(dumbo),
Darkgrey(g23),
(dumbo, g23)has-color,
$\forall$ has-color.LightGrey(dumbo) $\}$

ABox is a partial description of the world.

ABox reasoning

Instance checking:

:= Given a KB=$(\mathcal{T}, \mathcal{A})$, a concept description C and an individual a:
Is a an instance of C regarding $\mathcal{T}$ und $\mathcal{A}$?

Instance retrieval:

:= Given a KB=$(\mathcal{T}, \mathcal{A})$ and concept description C:
Which individuals from $\mathcal{A}$ are instances of C?

➽ allows only tree-shaped queries (since concept description)

ABox realization:

:= Given a KB=$(\mathcal{T}, \mathcal{A})$. Compute for all individuals a:
Of which concepts from $\mathcal{T}$ is a an instance?

Description Logics

DLs research:

- devise sound and complete reasoning algorithms (guaranteed quality)
- investigate trade-off between expressivity of DL and complexity of reasoning

DLs today:

- optimized reasoning systems available (RacerPro, Fact++, Pellet, Hermit, jCEL, QuOnto, . . .)
- Two directions: expressive DLs "vs." lightweight DLs
  - lightweight DLs tailored to specific reasoning service
  - Reflected in OWL 2.0 standard: profiles

Context-aware system I: intelligent door system

Application: intelligent door system

Intelligent door system:

- Determine next action depending on:
  person ringing or situation of the resident
- System is equipped with:
  video camera, sensors, connections to other devices in the house
- Decide which action is to be taken:
  - open door
  - leave door closed or
  - contact the resident

Context knowledge base

TBox: Concepts that describe categories of contexts

e.g.: kinds of acting persons, locations
technical details of devices, etc.

ABox: Individuals that describe the current situation

e.g.: hasLocation(Alice, AliceHome), Employee(Alice),
uses(Alice, AliceMobile)

DL systems can

- model contexts and situations on different levels of detail
- handle incomplete situation descriptions gracefully

➽ Do context recognition by DL reasoning!

DL reasoning for building the context TBox

At design time:

- Build context TBox
- Model context categories
- Test: are the context concepts . . .
  - contradictory? — Test for TBox consistency.
  - modelled on a suffcient level of detail?

    Are subsumption relations missing? } Classify TBox.

    Are there unintended subsumptions? } Classify TBox.

    Are there unintended synonyms? — Test for equivalent concepts.

DL reasoning for the situation ABox

At run time:

- Context application provides situation information as ABox assertions using concepts from the context TBox.

- Test:
  - Does the situation description contain a contradiction? — Test for ABox consistency.
  - To which context concept does the current situation belong? — Do ABox realization.

Example: person concepts from the context TBox

$\mathsf{VacationResident} :=$
$\mathsf{Resident} \sqcap \exists \mathsf{hasActivity}.\mathsf{Vacation}$

$\mathsf{AuthorisedPerson} :=$
$\mathsf{Resident} \sqcup (\mathsf{Person} \sqcap \exists \mathsf{AuthorisedBy}.\mathsf{Resident})$

$\mathsf{AuthorisedNeighbour} :=$
$\mathsf{Neighbour} \sqcap \exists \mathsf{AuthorisedBy}.\mathsf{Resident}$

Example: context concepts from the context TBox

$$\mathsf{DoorContext} := \mathsf{Context} \sqcap \exists \mathsf{hasContextAgent}.(\mathsf{Person} \sqcap \exists \mathsf{isRinging}.\mathsf{DoorBell}) \sqcap \exists \mathsf{hasContextResident}.\mathsf{Resident}$$

$$\mathsf{AuthorisedPersonRingingContext} := \mathsf{DoorContext} \sqcap \exists \mathsf{hasContextAgent}.\mathsf{AuthorisedPerson}$$

$$\mathsf{ResidentOutOfHomeContext} := \mathsf{DoorContext} \sqcap \exists \mathsf{hasContextResident}.\mathsf{ResidentOutOfHome}$$

$$\mathsf{AuthorisedNeighbourRingingContext} := \mathsf{DoorContext} \sqcap \exists \mathsf{hasContextAgent}.\mathsf{AuthorisedNeighbor}$$

Context recognition

Procedure:

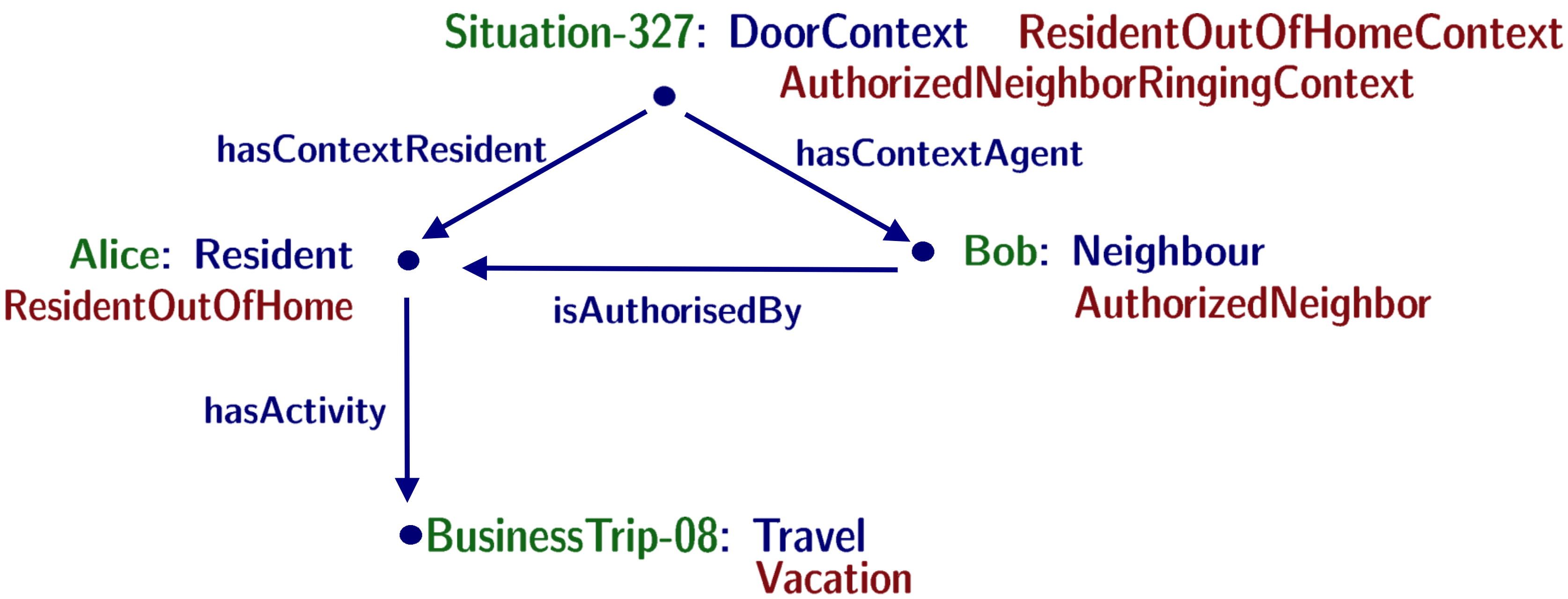
1.) From data sources the context application generates a situation ABox describing situation-327.

2.) Realization of the ABox individual situation-327.
I.e. compute the set of most specific named concepts from the context TBox that situation-327 is an instance of.

3.) Return the set of concepts to the context application.

4.) Context application performs action associated with this set of obtained concepts.

Realization in the situation ABox



Realization of the ABox individual Situation-327:

- AuthorisedNeighborRingingContext, ResidentOutOfHomeContext

Conjunctive queries

Definition:

A conjunctive query $q(X)$ over a knowledge base has the form:

$$q(X) \longleftarrow \exists Y.conj(X, Y).$$

where

X : is a tupel of "distinguished" variables (answer set variables)

Y : is a tuple of "undistinguished" variables (existentially quantified variables)

$conj$: a set of expressions of the form: $C(z_i)$ or $r(z_i, z_j)$,

A union of conjunctive queries is a disjunction of conjunctive queries.

Conjunctive query: example

$q_1(x_1) \longleftarrow$
$\exists y_1, y_2.\{\text{Person}(x_1),\ \text{hasNeighbour}(x_1, y_1),\ \text{isAuthorizedBy}(x_1, y_2)\}.$

$x_1 = \text{Bob}$

$q_2(x_1) \longleftarrow$
$\{\text{Person}(x_1),\ \text{hasNeighbour}(x_1, \text{Alice}),\ \text{isAuthorizedBy}(x_1, \text{Alice})\}.$

$x_1 = \text{Bob}$

$q_3(x_1, x_2) \longleftarrow$
$\{\text{Person}(x_1),\ \text{hasNeighbour}(x_1, x_2),\ \text{isAuthorizedBy}(x_1, x_2)\}.$

$x_1 = \text{Bob}$
$x_2 = \text{Alice}$

Query answering

Query answering:

:= Returns all (bindings of the variables in X of) individuals from the ABox that fullfill the query.

- flexible way to access data in the ABox
- can be done very efficiently for the OWL 2.0 QL profile (underlying DL: DL Lite)

Context-aware application II: Avatr application

Application: Avatr web service recommendations

Avatr application:

- virtual assistant for for recommending various on-line services
- system makes context-dependant recommendations
- recommendations depend on the information available from / about the user

On-line services:

- fixed set of Avatr services
- described by service constructors
  service constructor: set of parameters for the service

Avatr example

Avatr user Fred

- plans holiday travel to New York to see a basket ball event
- he uses the hotel service within Avatr for booking
- is a folk rock fan;
  has many folk rock musicalbums saved on his computer

Avatr application:

- "knows" his travel destination and dates
- "knows" his taste regarding music
- proposes related events in New York at for the week of travel
  e.g. a folk festival

Building the knowledge base

Information sources:

- Java code of the Avatr application
  description of the services
- User profile: data from earlier sessions
- User data from current sessions

Generating the TBox

Generating the TBox from the Java sources:

- Close connection between UML class diagramms and DLs
- Correspondance: classes $\approx$ concepts,
  attributes $\approx$ roles,
  inheritance $\approx$ subsumption
- Differences:
  - Attributes are local to a class
    (can be achieved by renaming)

Obtained knowledge base

Obtained TBox:

- simple structure
- 50 concepts; 180 roles
- augmented by hand
- DL Lite_{core}

Examples:

$\text{AvatrEvent} \sqsubseteq$
$\exists\ \text{has-city}.\top \ \sqcap\ \exists\ \text{has-date}.\top$
$\text{Festival} \sqsubseteq \text{AvatrEvent}$

⋮

for

- basis for OWL 2 QL

Obtained ABox:

- generated from DB Pedia
- only names in concept assertions: DL Lite!
- augmented by hand

Avatr example

Now find the folk festival!

Formulate query that uses the available constructor information.

$$q(x) \longleftarrow$$
$$\exists\, y_c, y_d.\ \text{User(Fred)} \wedge \text{AvatrEvent}(x) \wedge \text{City}(y_c) \wedge \text{uses(Fred}, y_c) \wedge \text{Date}(y_d) \wedge \text{uses(Fred}, y_d) \wedge \text{located}(x, y_c) \wedge \text{starts}(x, y_d)$$

No results?

Generalize the query: search for events in the same state.

$$q(x) \longleftarrow$$
$$\exists\, y_c, y_d, y_e, y_f.\ \text{User(Fred)} \wedge \text{AvatrEvent}(x) \wedge \text{City}(y_c) \wedge \text{uses(Fred}, y_c) \wedge \text{Region}(y_e) \wedge \text{located-in}(y_c, y_e) \wedge \text{City}(y_f) \wedge \text{located-in}(y_f, y_e) \wedge \text{Date}(y_d) \wedge \text{uses(Fred}, y_d) \wedge \text{located}(x, y_f)\ y_c) \wedge \text{starts}(x, y_d)$$

TU Dresden

Summary and conclusions

Description Logics:

- offer powerful reasoning services that are based on the formal semantics of DLs
- allow to model contexts and situations
- DL reasoning services can be employed for context recognition
- DL reasoning services are a versatile methods for the OWL community to extract implicit information

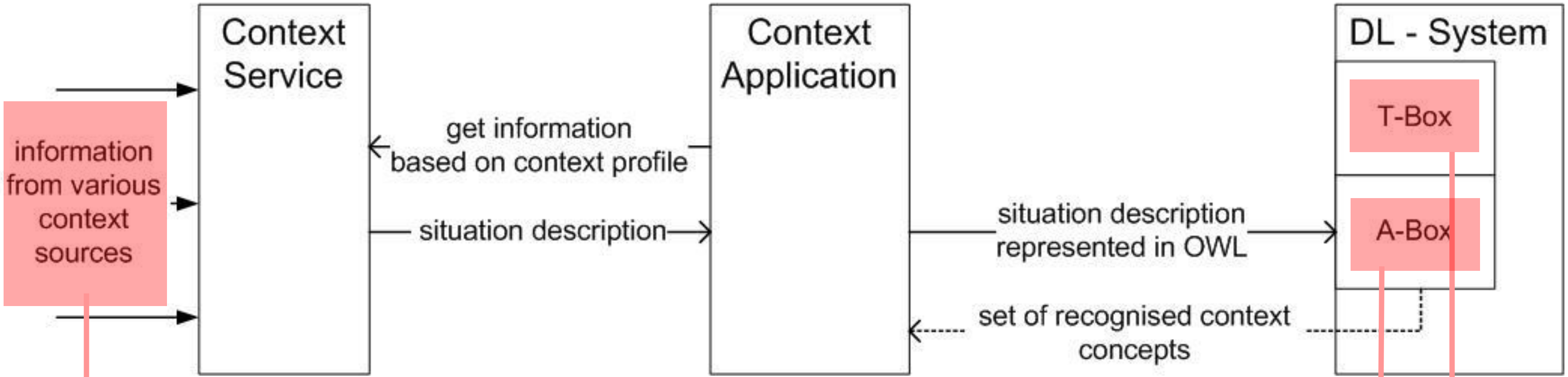
Open issues

- treatment of numerical values (e.g. user preferences, measurements)
- conversion: sub-symbolic to symbolic
- representing uncertainty
- often times tools are not “there” (yet)
- DL community needs exchange with users!
  - user knowledge bases $\rightsquigarrow$ benchmarks
  - user problems $\rightsquigarrow$ new reasoning services

Really!

Thank you!

Context framework



- low-level data
- e.g. data from sensors

Contains descriptions of cc ... nt to the application

— divided into task contexts

Contains description of individual situation

DL Lite

DL Lite:

- family of DLs: optimized for trade-off between expressivity and complexity of query answering
- "maximal" DLs with this property
- can express basic constructs of UML class and ER diagramms

DL Lite syntax:

concept axioms: $Cl \sqsubseteq Cr$:

$$Cl \longrightarrow A \mid \exists Q$$

$$Cr \longrightarrow A \mid \exists Q \mid \neg A \mid \neg \exists Q$$

$$Q \longrightarrow P \mid P^{-}$$

roles:

- functional roles
- role inclusions $Q \sqsubseteq R$
- domain & range restrictions

Komplexität von Query answering für conjunctive Queries

Expressive DLs:

- $\mathcal{SHIQ}$
  combined complexity: 2ExpTime-complete
- $\mathcal{ALCI}$
  combined complexity: 2ExpTime-complete

light-weight DLs:

- $\mathcal{EL}$:
  NP-hard für combined complexity
- $\mathcal{DL}$-Lite:
  PTime in size of TBox
  LogSpace for data complexity