Ontologies and Reasoning for Ambient Assisted Living

by

Immanuel Normann, Frank Dylla, Joana Hois, Oliver Kutz, Mehul Bhatt

Wolfgang Putz, Mario Schmitt, Sebastian Weber
When somebody rings at my house door while I am outdoors, the system should send me an email instead...

```
public class dispatcher {
    private String defaultMessage;
    private Person resident;
    ....
    public void sendMessage (String msg, Location loc) {
        if (loc.atHome) {homzone.tell(msg)}
        else {
            try {email.send(msg, resident.emailAdress)}
            catch {.....
```
Monitoring visitor access via audio/video communication contributes a lot to this sense of security. It allows the resident to identify who is at the front door and to freely decide who is allowed to enter. Unknown visitors can be refused, and even if the resident is not at home the access system could leave a message who wanted to visit. Actual domotic principally supports this functionality, but it could be more comfortable, flexible, and user-friendly, if intelligent technology were used. Up to now, usually in the house or flat a proprietary system with a central monitor is installed that allows you to get in touch with the visitor via audio/video communication. If the resident has reduced mobility, reaching the central interaction device could be cumbersome or even impossible.
Gain of Ontological Approach

1. Ontology languages are comprehensible
Gain of Ontological Approach

1. Ontology languages are comprehensible

2. Ontologies represent models as explicitly as needed
Gain of Ontological Approach

1. Ontology languages are comprehensible
2. Ontologies represent models as explicit as needed
3. Ontologies can be used as a substantial part of the executable code!
Overview

- Software interaction
- Basic evaluation loop in an ontological program
- Case study for Ambient Assisted Living
- Reasoning support for main stream programming languages
- Outlook
Interaction of Software
imperative perspective

Process

Input  Output

Internal State
Interaction of Software
ontological perspective

Event

Reasoning

knowledge base

Effect

observes

produces
Interaction of Software
ontological perspective

Observation

Reasoning

knowledge base

Reaction

Event \quad \text{observes} \quad \text{produces} \quad \text{Effect}
Overview

- Software interaction
- **Basic evaluation loop in an ontological program**
- Case study for Ambient Assisted Living
- Reasoning support for main stream programming languages
- Outlook
Office

Ontology

rule
If a person is in the office then the light is switched on
rule
If a person is in the office then the light is switched on
If a person is in the office then the light is switched on
rule
If a person is in the office then the light is switched on

new fact:
A person is in the office
rule:
If a person is in the office then the light is switched on

new fact:
A person is in the office
new fact: A person is in the office

rule:
If a person is in the office then the light is switched on

derived fact: light is switched on
new fact: A person is in the office

rule:
If a person is in the office then the light is switched on

derived fact: light is switched on

actuator

sensor
rule
If a person is in the office then the light is switched on

new fact:
A person is in the office

derived fact:
light is switched on
new fact: A person is in the office

rule: If a person is in the office then the light is switched on

derived fact: light is switched on
Overview

- Software interaction
- Basic evaluation loop in an ontological program
- **Case study for Ambient Assisted Living**
- Reasoning support for main stream programming languages
- Outlook
The *Entrance Hall* scenario in one sentence:

when a visitor rings the door bell, the resident should be notified on a communication channel adequate for his/her current location.
A Case Study Ontology

- The resident is in the living room
A Case Study Ontology

- The resident is in the living room
- inside(living_room).
- note: we do not represent the resident in the fact, since ...
A Case Study Ontology

- The resident is in the living room
- inside(living_room).
- note: we do not represent the resident in the fact, since ...
- The TV can be seen from the sofa and the chair and speaker can be heard in the bath.
A Case Study Ontology

- The resident is in the living room
- inside(living_room).
- note: we do not represent the resident in the fact, since ...
- The TV can be seen from the sofa and the chair and speaker can be heard in the bath.
- covers(tv,sofa), covers(tv,chair), covers(speaker,bath).
A Case Study Ontology

- The resident is in the living room
- inside(living_room).
- note: we do not represent the resident in the fact, since ...
- The TV can be seen from the sofa and the chair and speaker can be heard in the bath.
- covers(tv,sofa), covers(tv,chair), covers(speaker,bath).
- A device D covers an area A if A is a subpart of an area B which is also covered by D.
A Case Study Ontology

- The resident is in the living room
- inside(living_room).
- note: we do not represent the resident in the fact, since ...
- The TV can be seen from the sofa and the chair and speaker can be heard in the bath.
- covers(tv,sofa), covers(tv,chair), covers(speaker,bath).
- A device D covers an area A if A is a subpart of an area B which is also covered by D.
- covers(D,A) if partOf(A,B) and covers(D,B).
A Case Study Ontology

- Video connection is undesirable in intimate spaces.
- A video connection $V$ is admissible if it covers the area $A$ where the resident is inside, but only if this area is not intimate.
A Case Study Ontology

• Video connection is undesirable in intimate spaces.

• A video connection $V$ is admissible if it covers the area $A$ where the resident is inside, but only if this area is not intimate.

• admissible($V$) if videoDevice($V$) and
  • covers($V,A$) and inside($A$) and not(intimate($A$)).
A Case Study Ontology

- Video connection is undesirable in intimate spaces.
- A video connection $V$ is **admissible** if it covers the area $A$ where the resident is inside, but only if this area is not intimate.
  - $\text{admissible}(V)$ if $\text{videoDevice}(V)$ and
    - $\text{covers}(V,A)$ and $\text{inside}(A)$ and not($\text{intimate}(A)$).
- facts:
  - $\text{intimate}(A)$ if $\text{sleeping\_room}(A)$ or $\text{bath}(A)$ or ...
  - $\text{videoDevice}(V)$ if $\text{television}(A)$ or $\text{computer}(A)$ or ...
  - $\text{partOf}(\text{chair, living\_room})$, ...
  - $\text{partOf}(\text{bed, sleeping\_room})$, ...
  - $\text{partOf}(\text{sleeping\_room, apartment})$, ...
Apartment

Ontology & Reasoning
rules:

When the bell rings open communication channel.
In intimate areas prefer audio over video connection.
...

dynamic facts:

Apartment

static facts:

The sofa is part of the sitting area in the living room.
The living room is part of the apartment...

dynamic facts:
Apartment

admissible(D) if
  videoDevice(D) and
  covers(D,A) and
  inside(A) and
  not(intimate(A)).

admissible(D) if
  audioDevice(D) and
  covers(D,A) and
  inside(A).

... partOf(sofa1,sitting_area).
    partOf(sitting_area,living_room)
    partOf(living_room,apartment)
...
dynamic facts:

admissible(D) if
  videoDevice(D) and
  covers(D,A) and
  inside(A) and
  not(intimate(A)).

admissible(D) if
  audioDevice(D) and
  covers(D,A) and
  inside(A).

... partOf(sofa1,sitting_area).
partOf(sitting_area,living_room)
partOf(living_room,apartment)
...
admissible(D) if videoDevice(D) and covers(D,A) and inside(A) and not(intimate(A)).

admissible(D) if audioDevice(D) and covers(D,A) and inside(A).

... partOf(sofa1,sitting_area).
partOf(sitting_area,living_room)
partOf(living_room,apartment)
...

dynamic facts:
admissible(D) if
  videoDevice(D) and
covers(D,A) and
inside(A) and
not(intimate(A)).

admissible(D) if
  audioDevice(D) and
covers(D,A) and
inside(A).

... partOf(sofa1,sitting_area).
partOf(sitting_area,living_room)
partOf(living_room,apartment)
...

dynamic facts:

inside(bed)
admissible(D) if
  videoDevice(D) and
  covers(D,A) and
  inside(A) and
  not(intimate(A)).

admissible(D) if
  audioDevice(D) and
  covers(D,A) and
  inside(A).

...partOf(sofa1,sitting_area).
partOf(sitting_area,living_room)
partOf(living_room,apartment)
...

dynamic facts:
inside(bed)
admissible(D) if videoDevice(D) and covers(D,A) and inside(A) and not(intimate(A)).

admissible(D) if audioDevice(D) and covers(D,A) and inside(A).

... partOf(sofa1,sitting_area).
partOf(sitting_area,living_room)
partOf(living_room,apartment)
...

dynamic facts:
inside(bed)
admissible(V) if videoDevice(V) and covers(V,A) and inside(A) and not(intimate(A)).

dynamic facts:

inside(bed)
admissible(V) if
videoDevice(V) and
covers(V,A) and
inside(A) and
not(intimate(A)).

dynamic facts:

inside(bed)
admissible(V) if
videoDevice(V) and
covers(V,A) and
inside(A) and
not(intimate(A)).

inside(A1) if
partOf(A2,A1) and
inside(A2).

partOf(bed, tv1_area).

dynamic facts:
inside(bed)
admissible(V) if
  videoDevice(V) and
  covers(V,A) and
  inside(A) and
  not(intimate(A)).

inside(bed) if
  partOf(bed,tv1_area) and
  inside(tv1_area).

partOf(bed,tv1_area).

dynamic facts:

inside(bed)
admissible(tv1) if
videoDevice(tv1) and
covers(tv1, tv1_area) and
inside(tv1_area) and
not(intimate(tv1_area)).

dynamic facts:
inside(bed) if
partOf(bed, tv1_area) and
inside(tv1_area).

partOf(bed, tv1_area).
partOf(tv1_area, sleeping_room).

intimate(A) if
partOf(A, sleeping_room).
admissible(tv1) if videoDevice(tv1) and covers(tv1,tv1_area) and inside(tv1_area) and not(intimate(tv1_area)).

inside(bed) if
partOf(bed,tv1_area) and inside(tv1_area).

partOf(bed,tv1_area).
partOf(tv1_area,sleeping_room).

intimate(bed) if
partOf(bed,sleeping_room).

dynamic facts:
inside(bed)
admissible(D) if
  videoDevice(D) and
  covers(D,A) and
  inside(A) and
  not(intimate(A)).

admissible(D) if
  audioDevice(D) and
  covers(D,A) and
  inside(A).

... partOf(sofa1,sitting_area).
partOf(sitting_area,living_room)
partOf(living_room,apartment)
...
Typical Objections

- „These rules don't look like OWL!“
- „Why not just implement rules via if-then in Java?“
- „This was Prolog, but we want Java!“
Overview

- Software interaction
- Basic evaluation loop in an ontological program
- Case study for Ambient Assisted Living
- **Reasoning support for main stream programming languages**
- Outlook
Rule Engines

http://www.manageability.org/blog/stuff/rule_engines/view

- List of ca. 30 rule engines and reasoners implemented in Java.

- Most prominent systems:
  - JESS, OpenRules, Jboss Drools, Pellet, Jena, ...

Rule engines have been extensively field-tested in business-logic by global players! e.g. IBM's ILOG business rule management system (http://www.ilog.com)

=> automated reasoning is not just an academic playground, but already facilitates sophisticated applications.

OASIS can take advantage from this experience!
Outlook

- Explore further OASIS domains with the ontological approach.
- Adoption of other OASIS ontologies.
- Transform the Prolog prototype to a Java implementation with integrated reasoner & rule engine.
- Practical experience with reasoning on modularized ontologies.
- Apply advanced modularisation techniques as well as more sophisticated (qualitative) spatial reasoning.