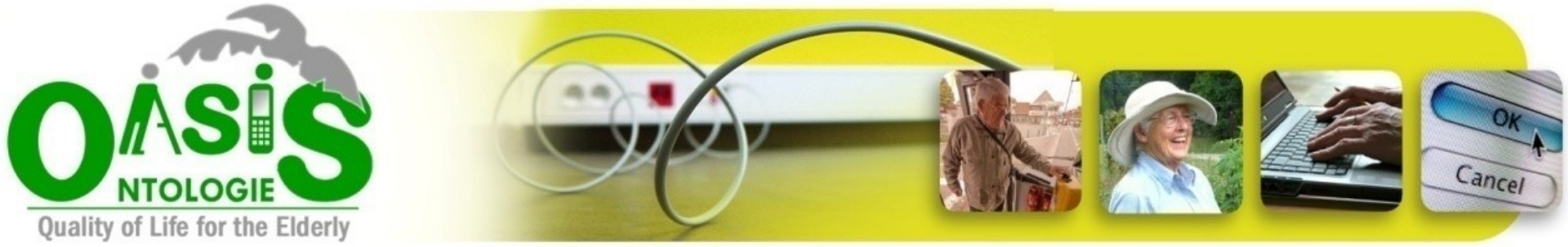


OASIS 1st International Conference

Florence, Italy – 4-5 November, 2009

Ontologies and Reasoning for Ambient Assisted Living

by

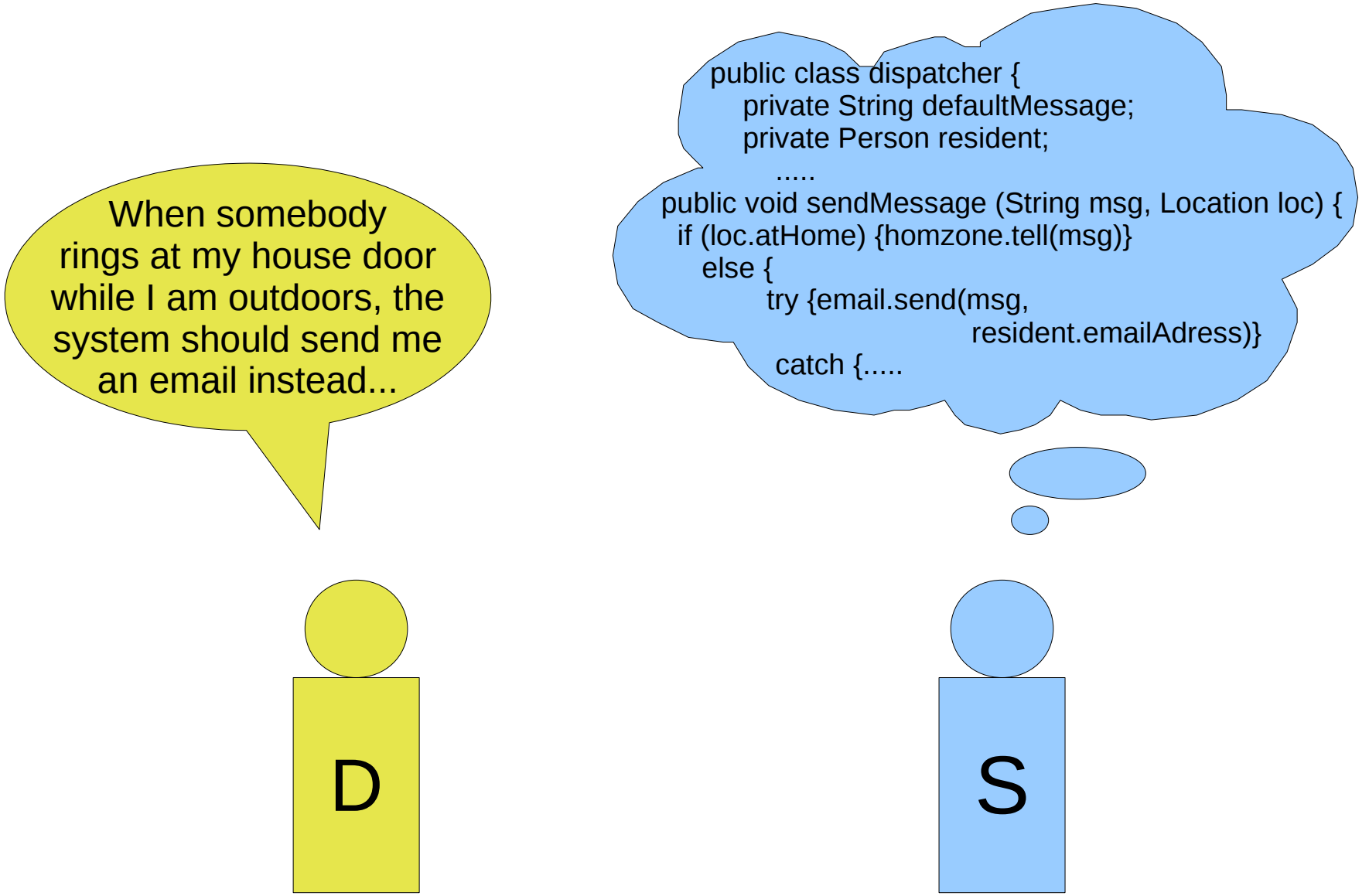
Immanuel Normann,
Frank Dylla, Joana Hois, Oliver Kutz, Mehul Bhatt

Universität Bremen

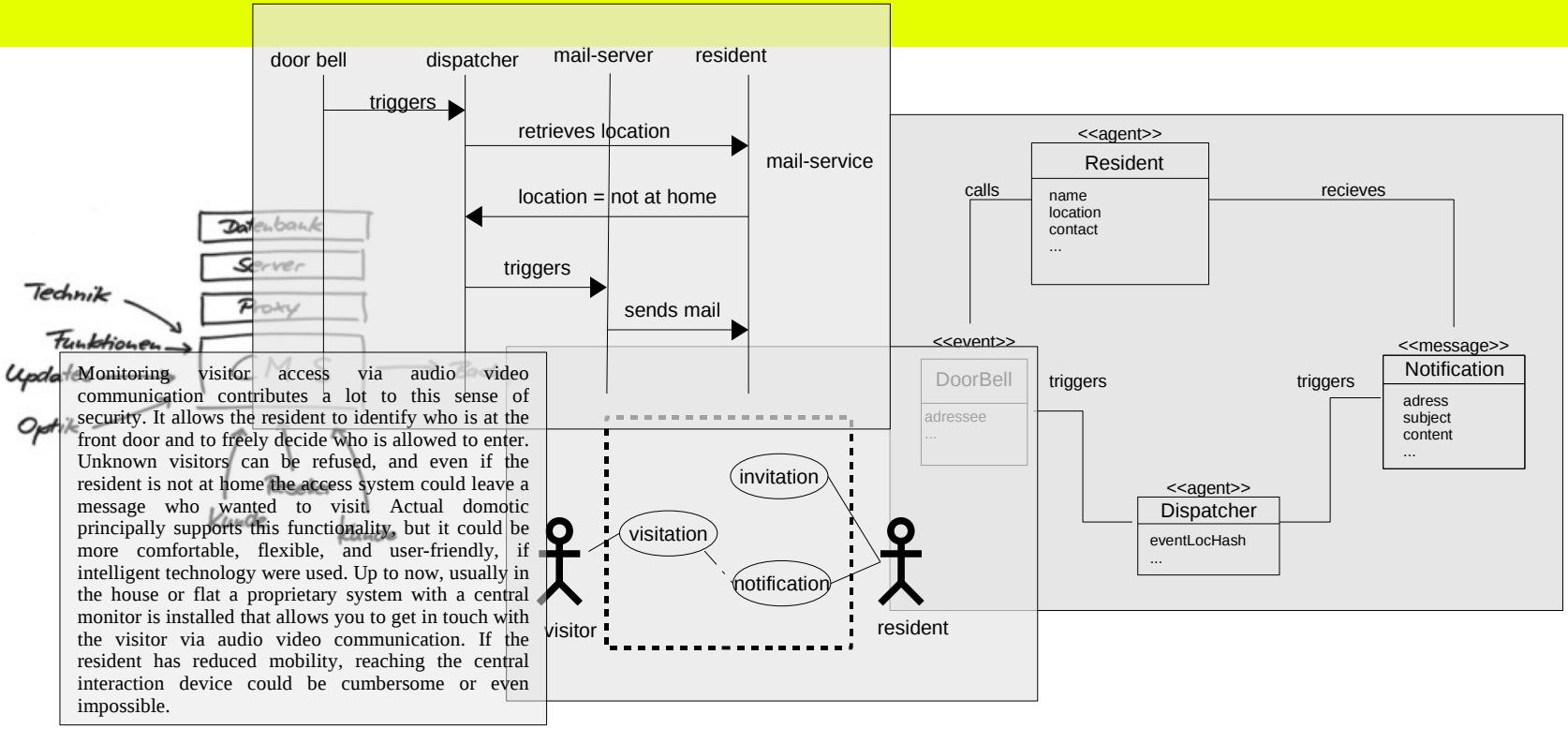
Wolfgang Putz,
Mario Schmitt,
Sebastian Weber

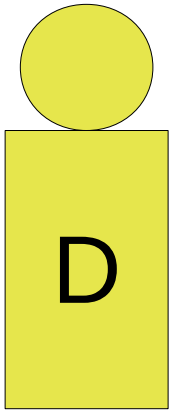
Fraunhofer IESE

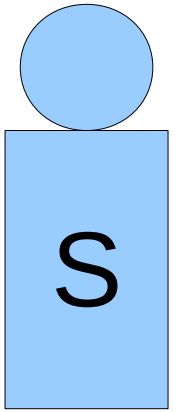
Software Development as Communication Challenge



Software Development as Communication Challenge







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

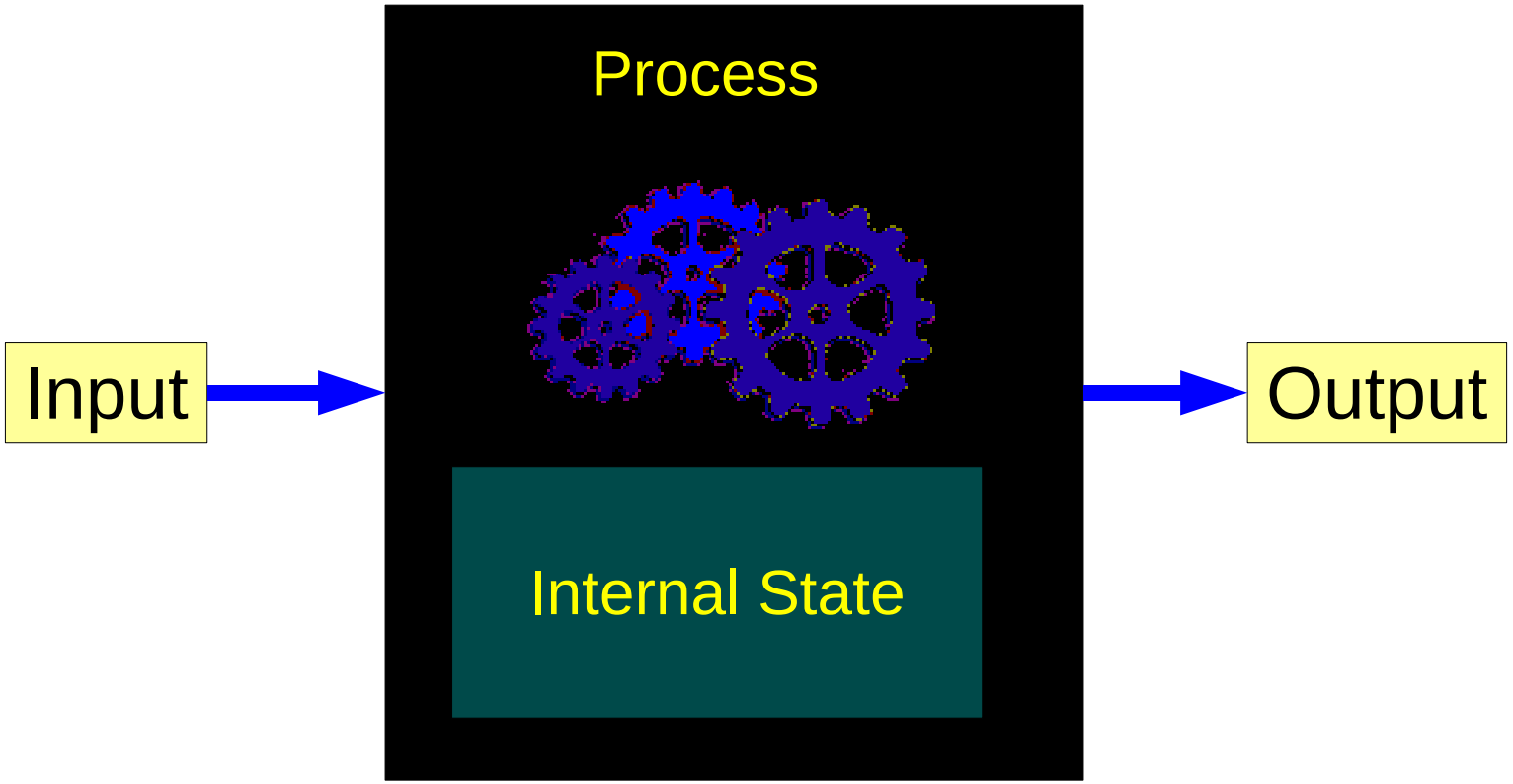
1. Ontology languages are comprehensible
2. Ontologies represent models as explicit as needed
3. Ontologies can be used as a substantial part of the executable code!

Overview

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

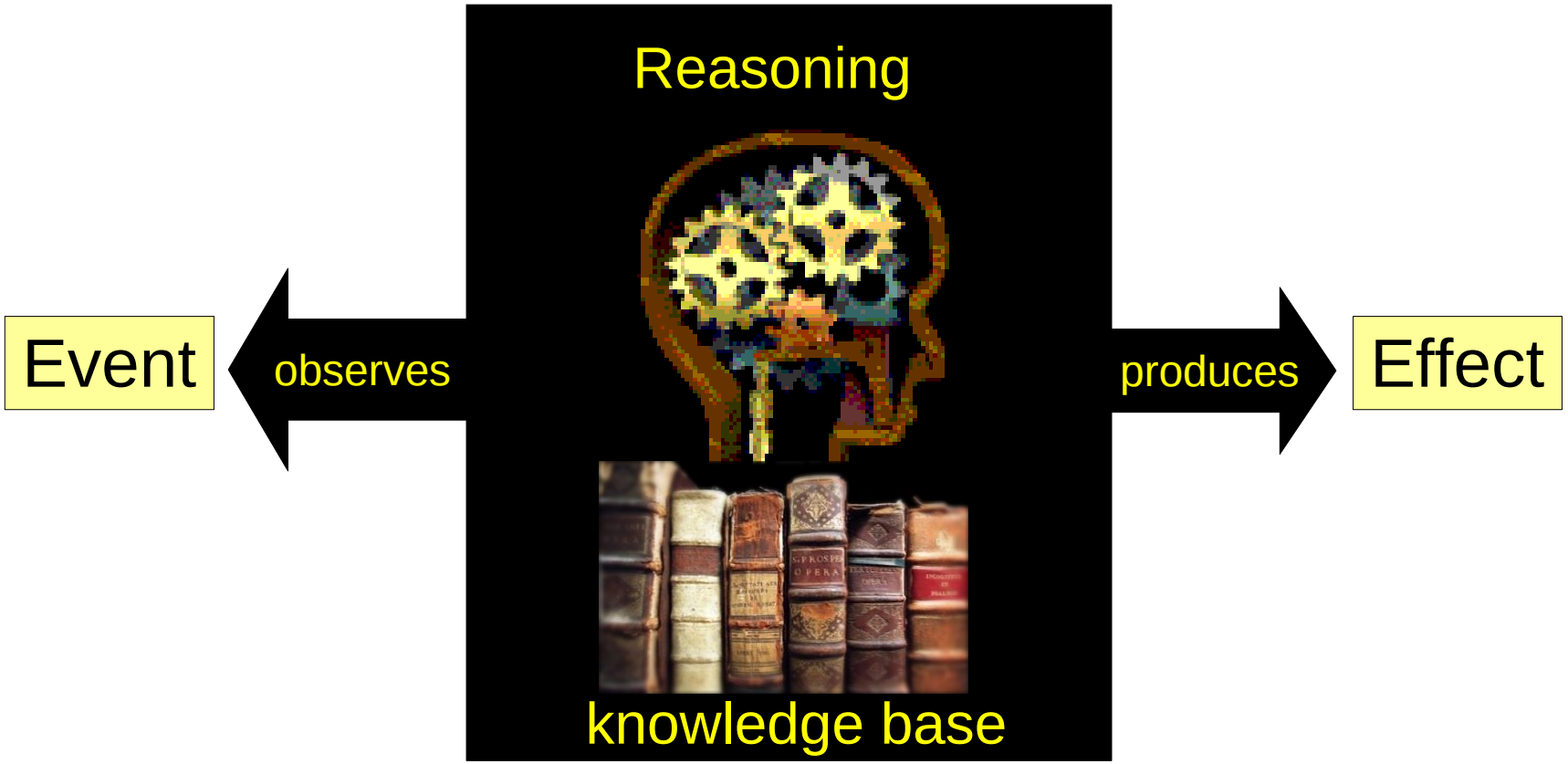
Interaction of Software

imperative perspective



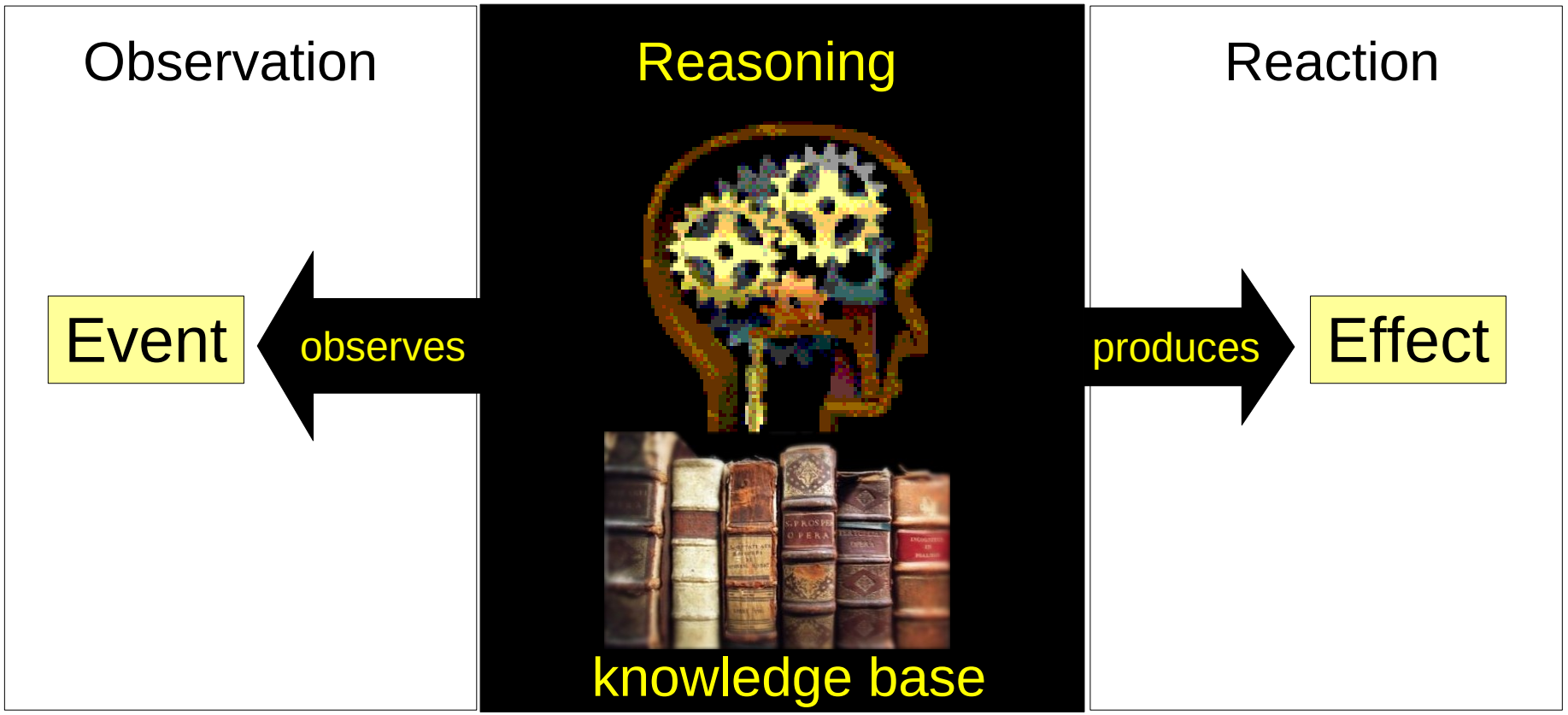
Interaction of Software

ontological perspective



Interaction of Software

ontological perspective



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

rule:

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

sensor

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

new fact:
A person is in the office

sensor

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

new fact:
A person is in the office

sensor

derived fact:
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

actuator

derived fact:
light is switched on

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

sensor

new fact:
A person is in the office

actuator

derived fact:
light is switched on

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

sensor

new fact:
A person is in the office

actuator

derived fact:
light is switched on

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

sensor

new fact:
A person is in the office

Overview

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

A Case Study Ontology

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

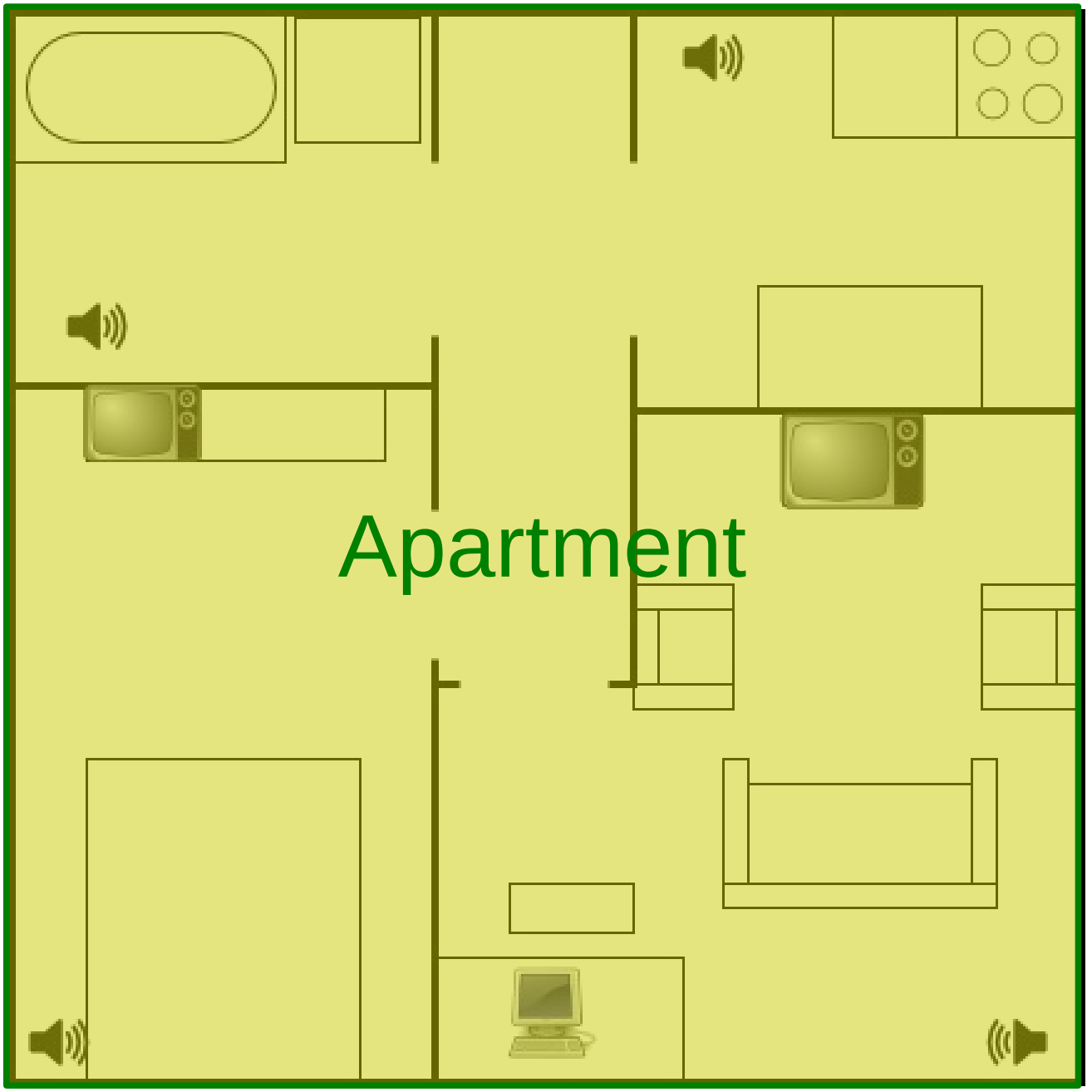
- Video connection is undesirable in intimate spaces.
- A video connection V is admissible if it covers the area A where the resident is inside, but only if this area is not intimate.

A Case Study Ontology

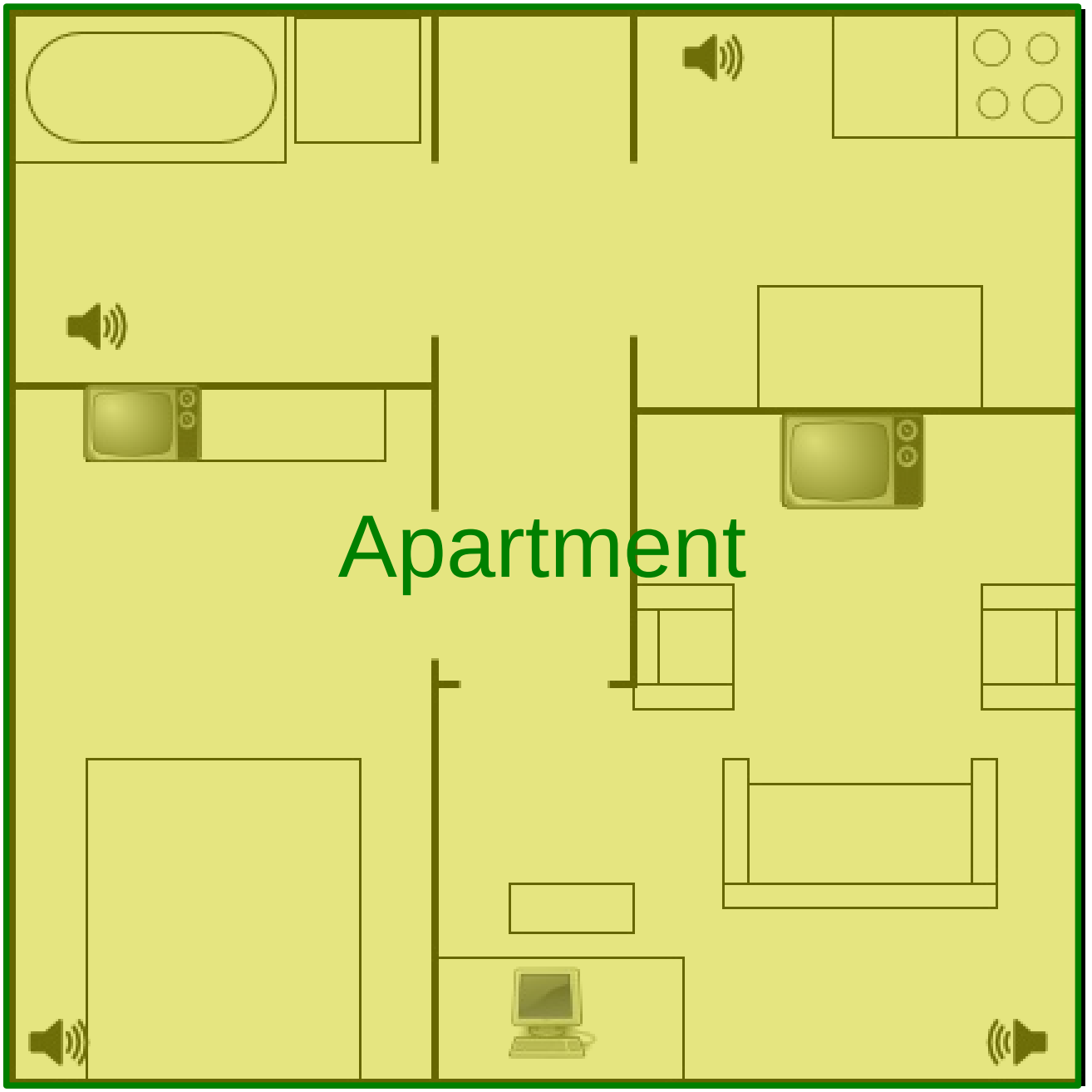
- Video connection is undesirable in intimate spaces.
- A video connection V is admissible if it covers the area A where the resident is inside, but only if this area is not intimate.
- admissible(V) if videoDevice(V) and
  - covers(V,A) and inside(A) and not(intimate(A)).

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.
- admissible(V) if videoDevice(V) and
  - covers(V,A) and inside(A) and not(intimate(A)).
- facts:
- intimate(A) if sleeping_room(A) or bath(A) or ...
- videoDevice(V) if television(A) or computer(A) or ...
- partOf(chair,livinging_room), ...
- partOf(bed,sleeping_room), ...
- partOf(sleeping_room,apartment),...



Ontology
&
Reasoning



Knowledge base

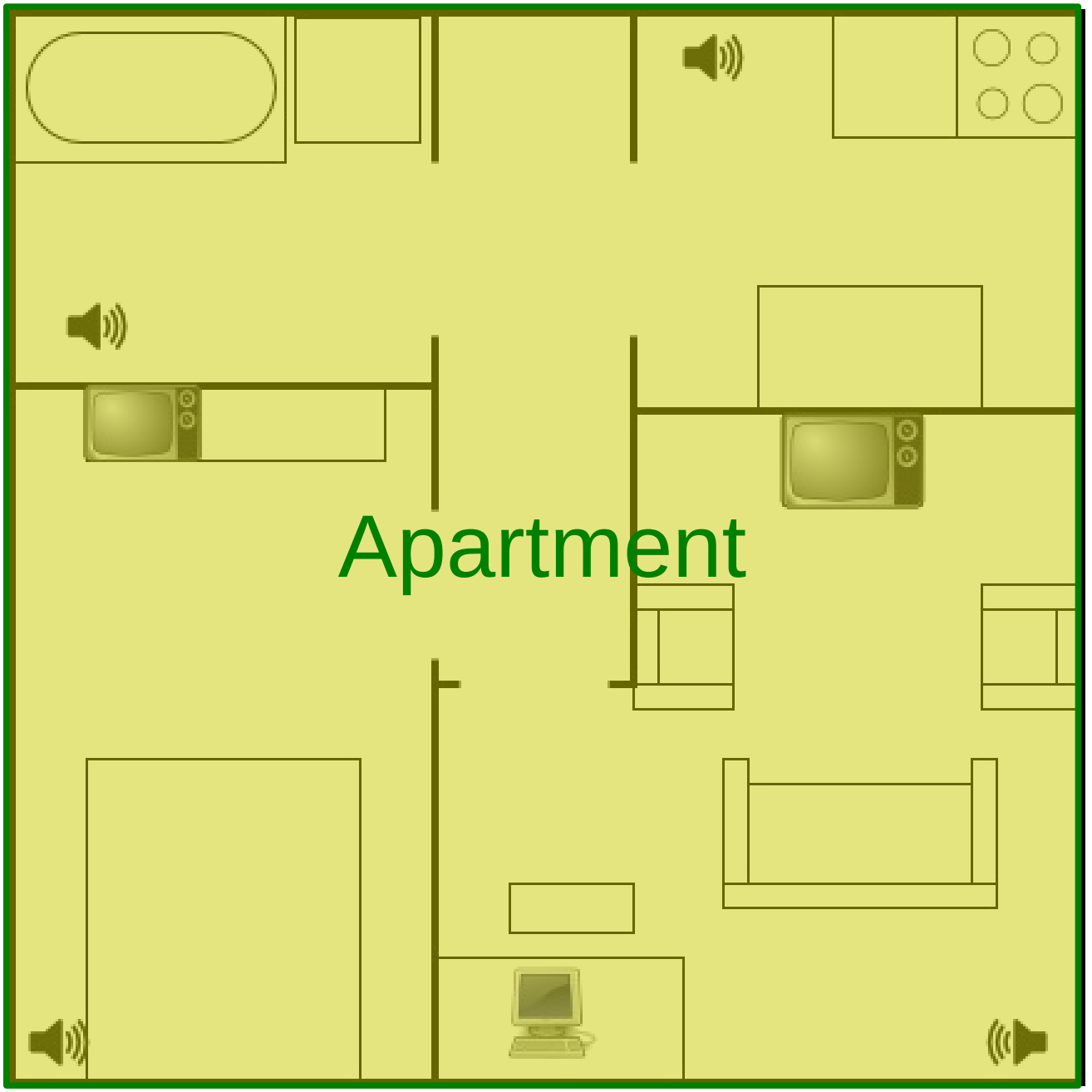
rules:

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

static facts:

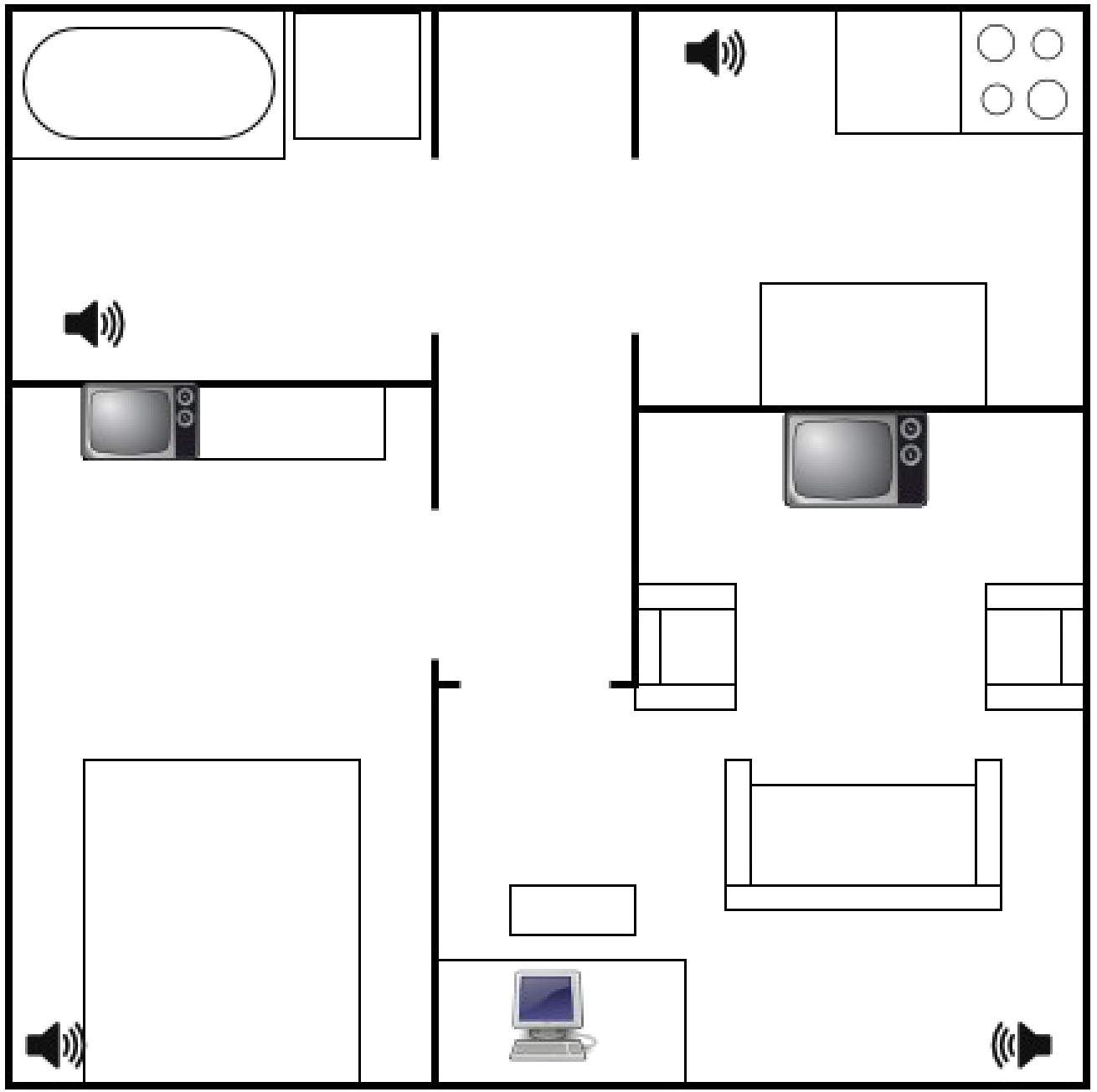
The sofa is part of the sitting area in the living room
The living room is part of the apartment ...

dynamic facts:



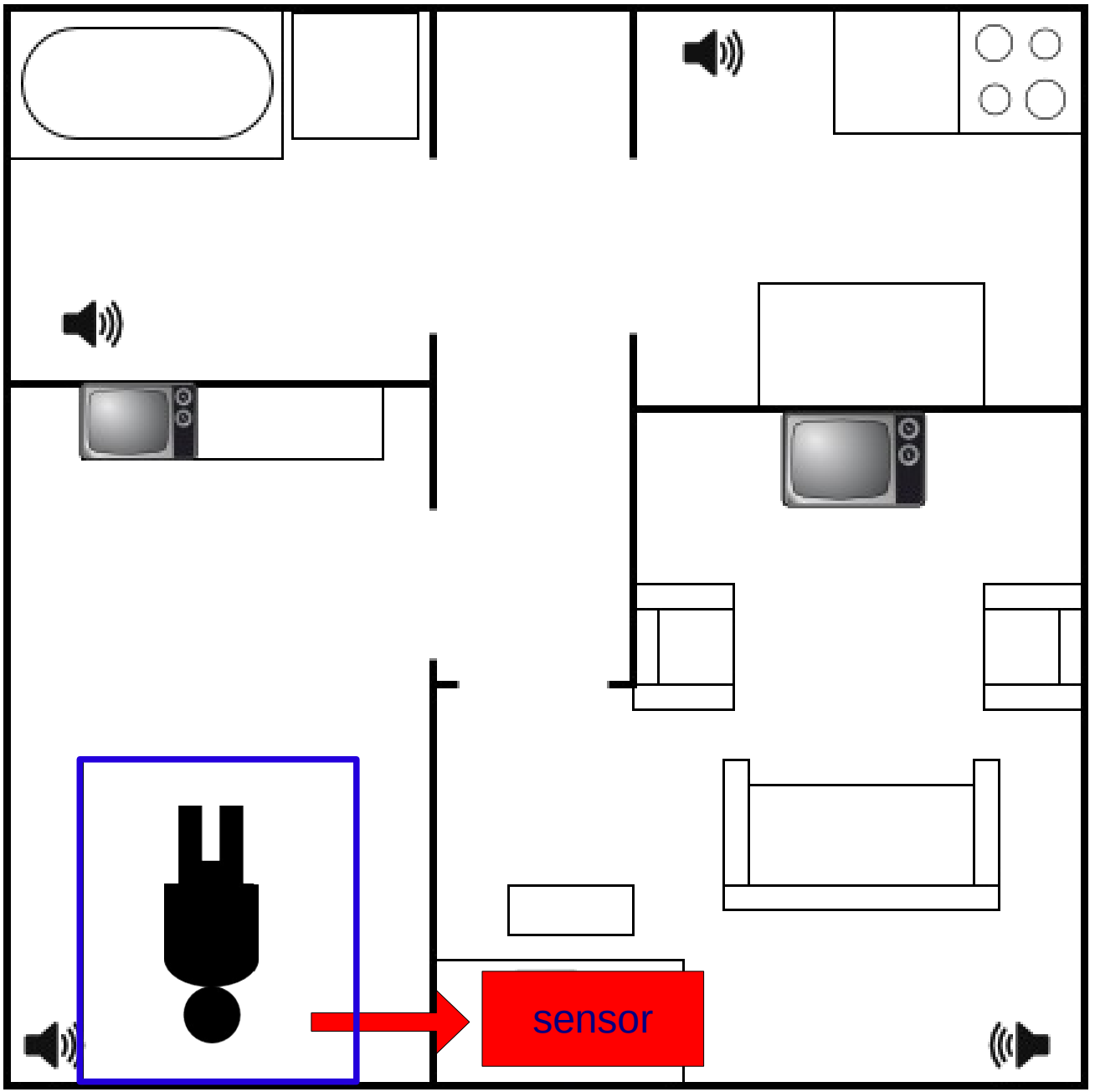

```
admissible(D)  if
      videoDevice(D) and
      covers(D,A) and
      inside(A) and
      not(intimate(A)).

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:



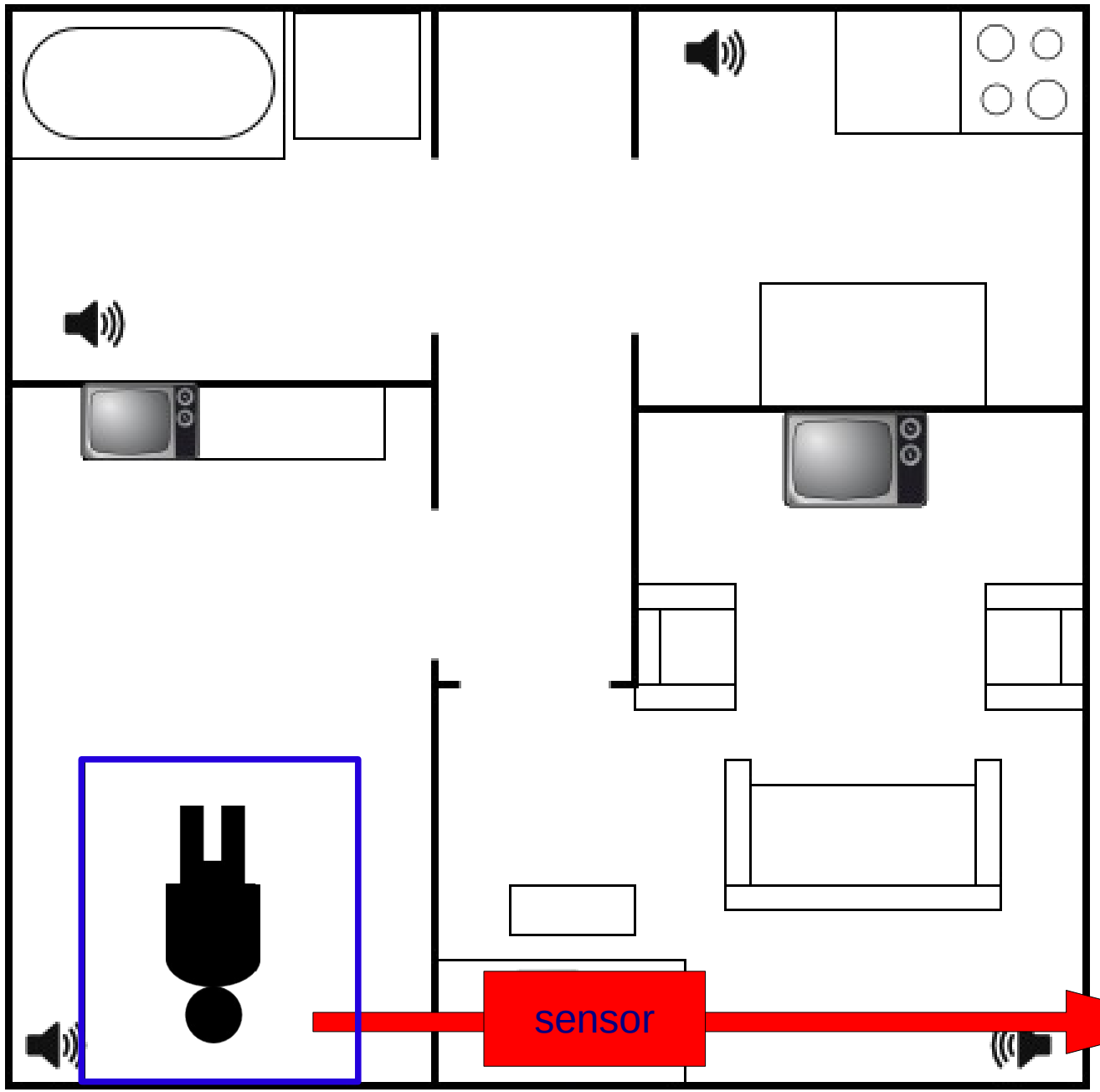
```
admissible(D)  if
      videoDevice(D) and
      covers(D,A) and
      inside(A) and
      not(intimate(A)).

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)
...
```

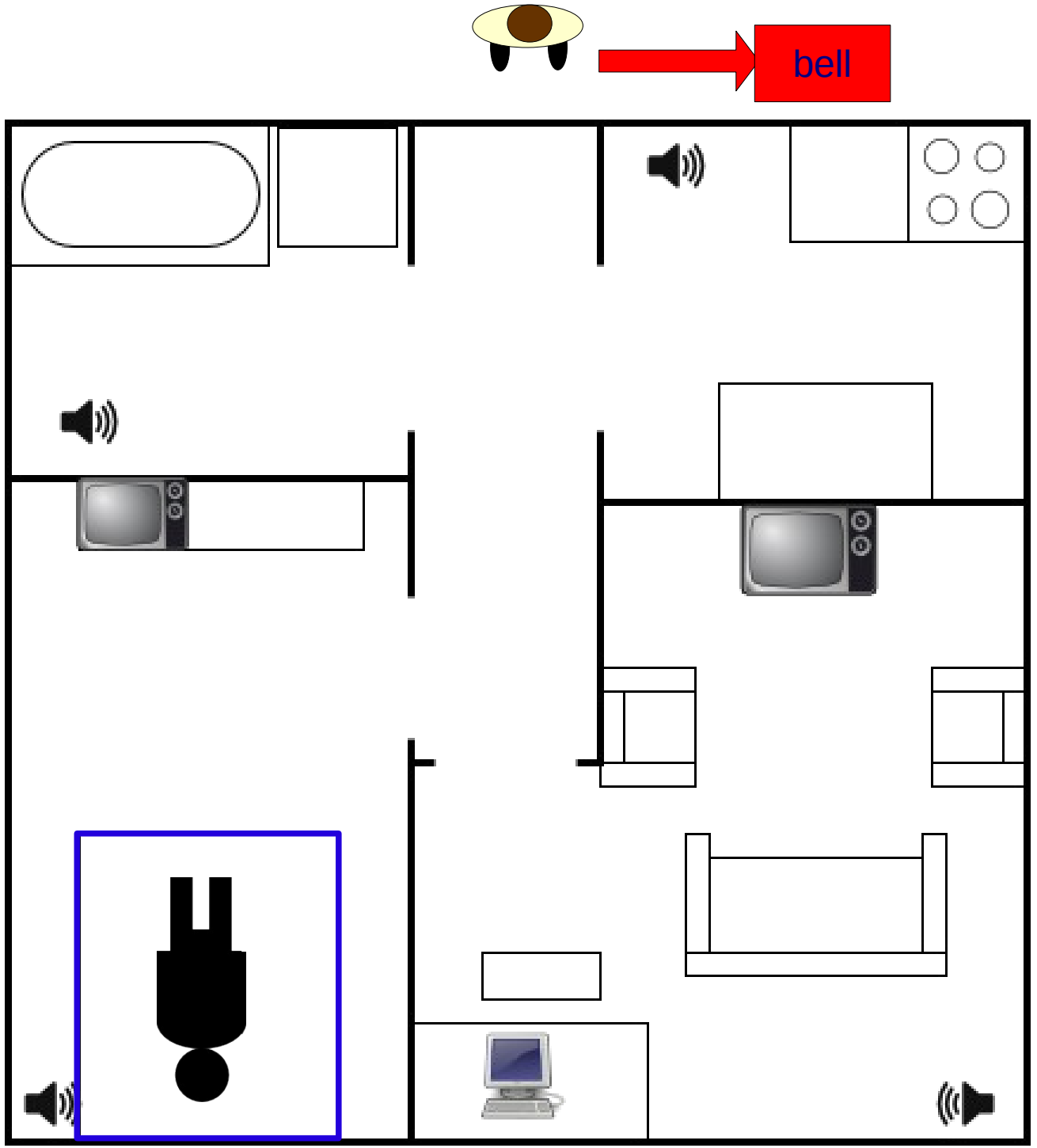
dynamic facts:




```
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:

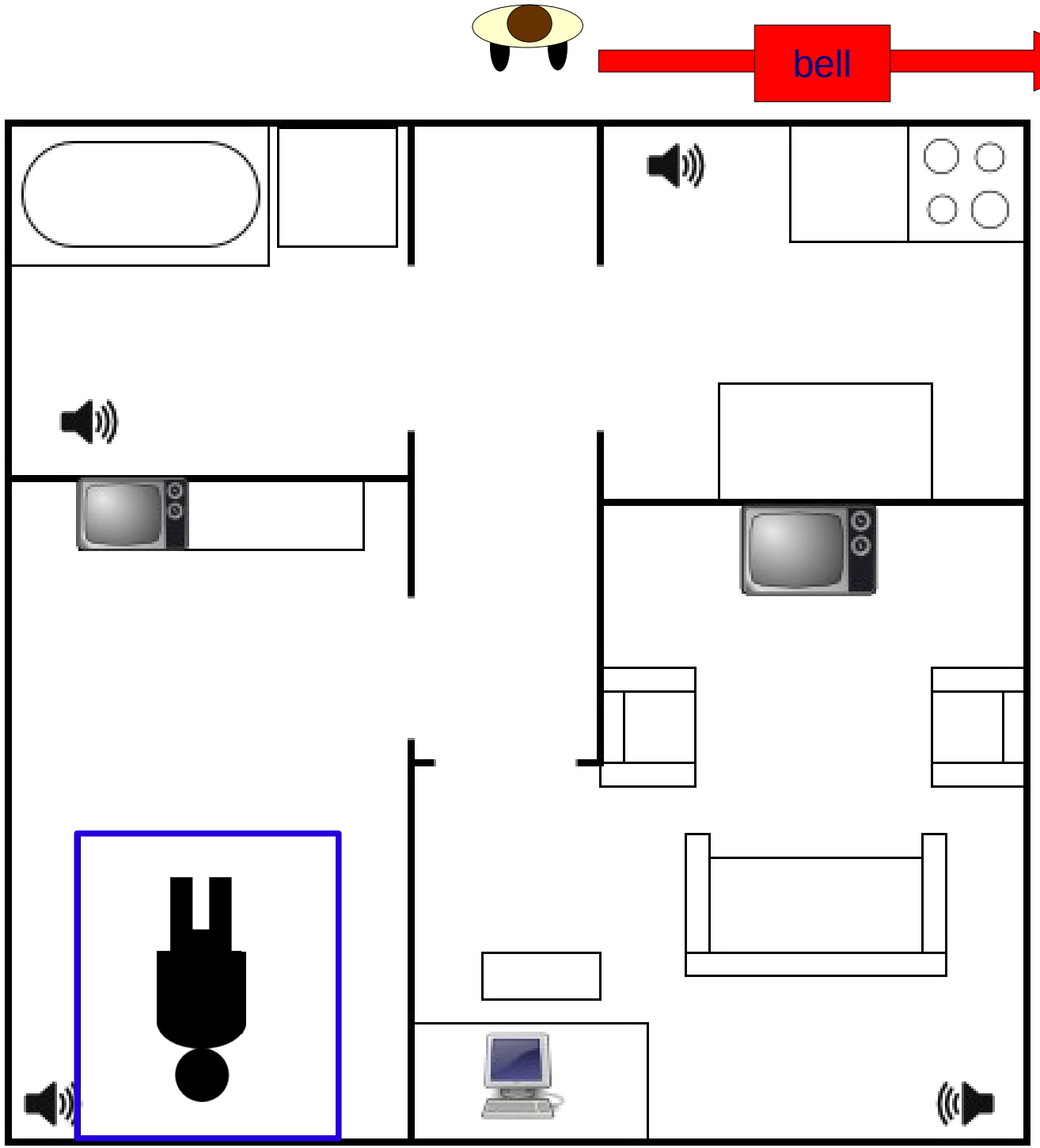
inside(bed)




```
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:

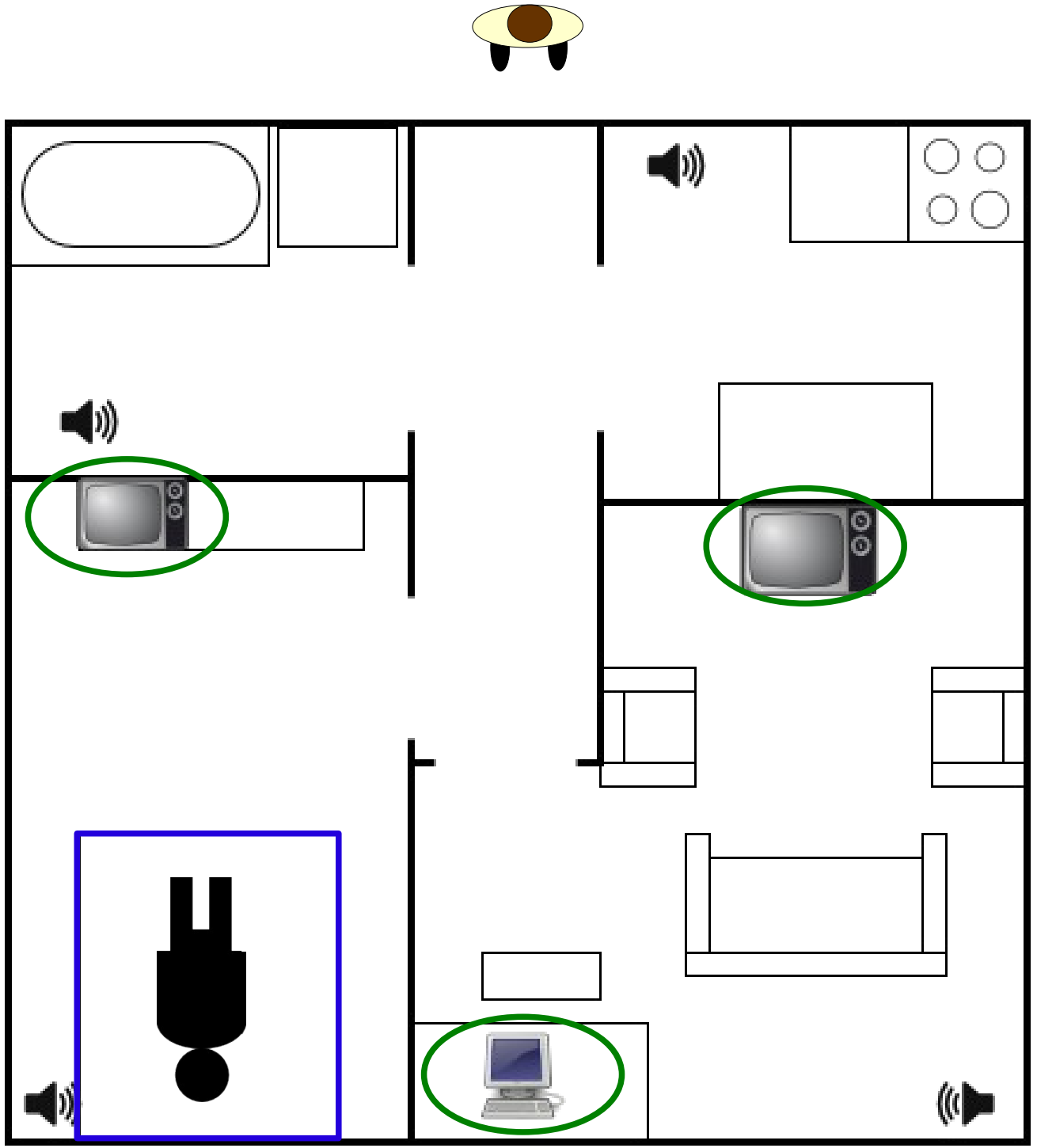
inside(bed)




```
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)

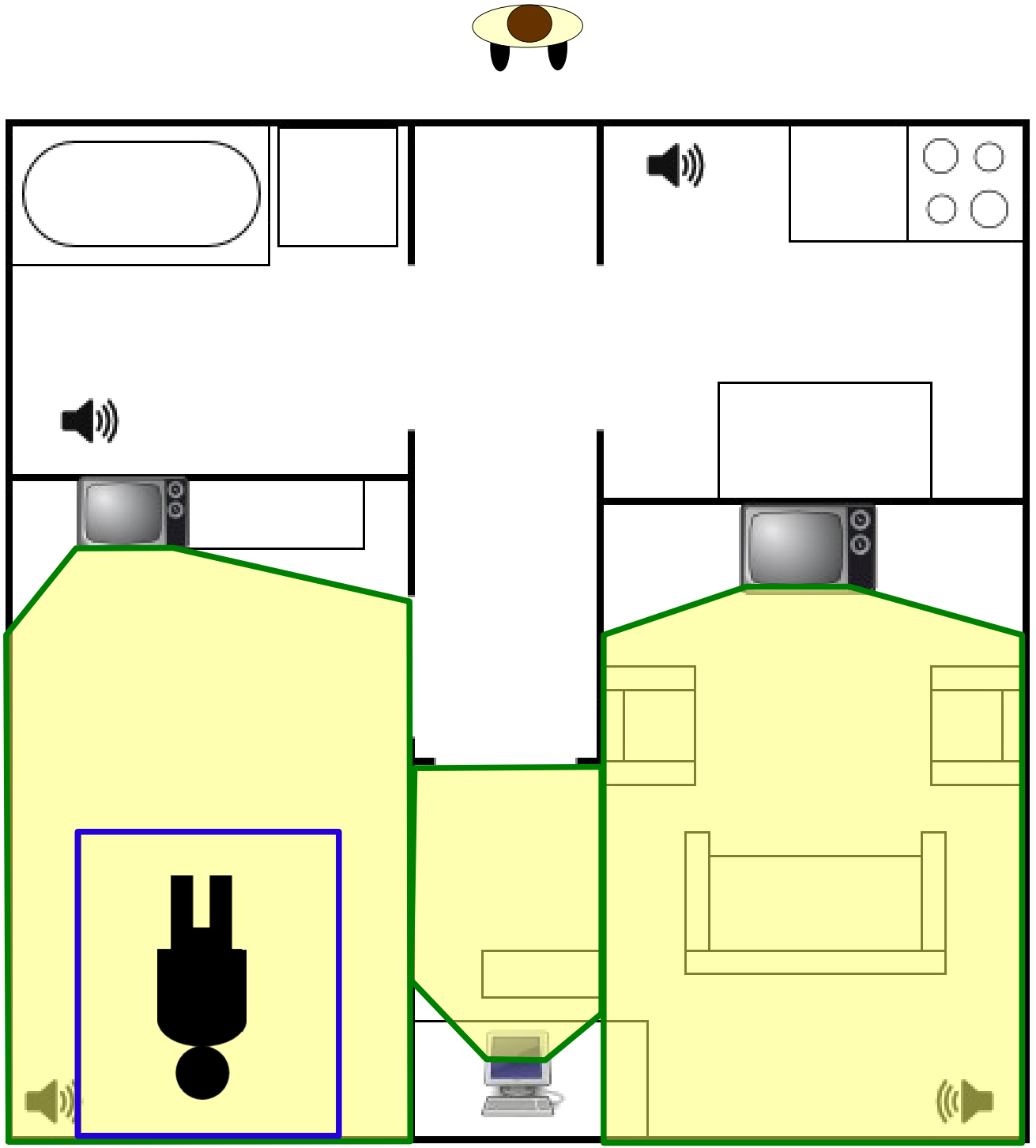


admissible(V)?

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

dynamic facts:

inside(bed)

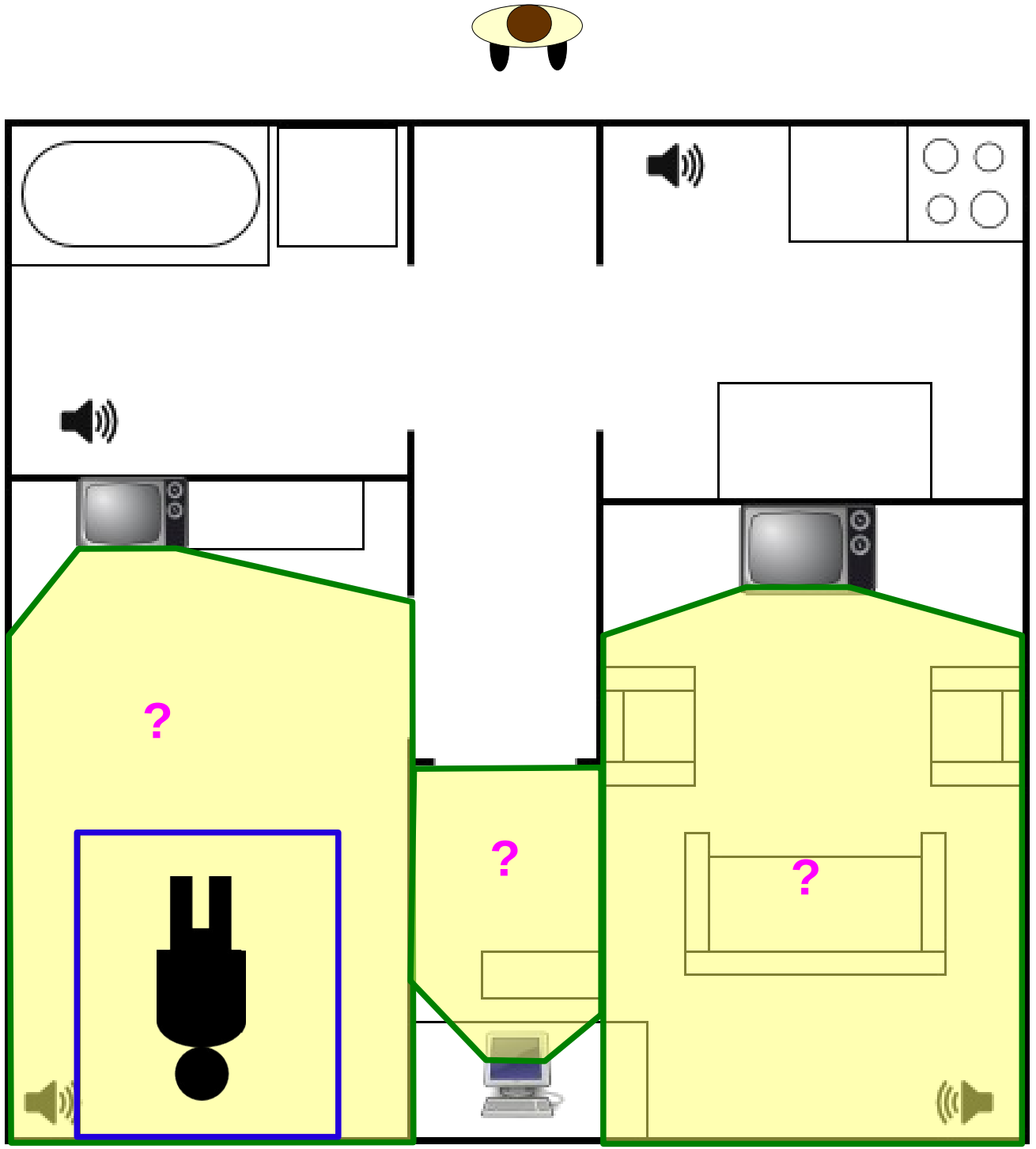


admissible(V)?

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

dynamic facts:

inside(bed)



admissible(V)?

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

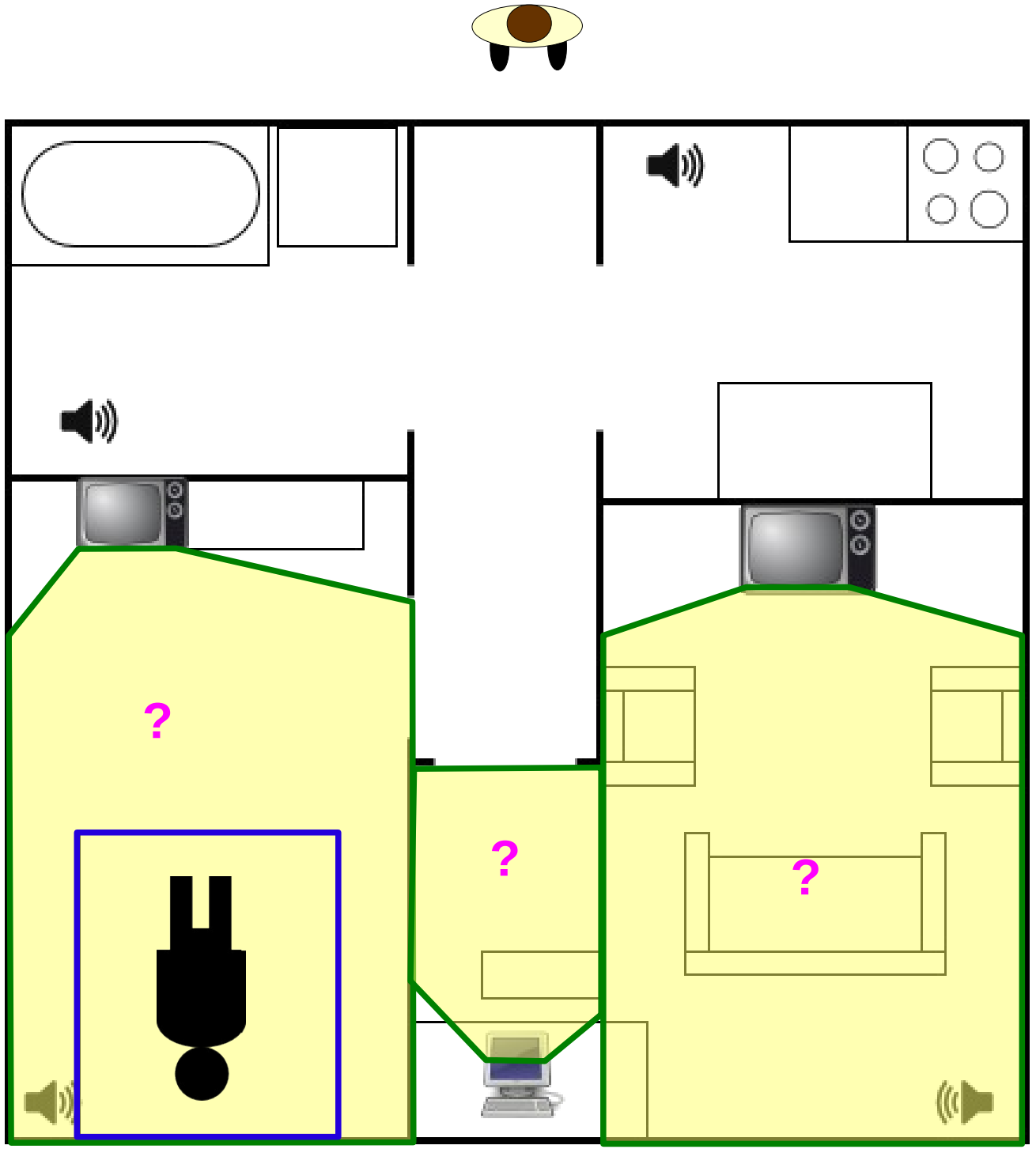
?

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

partOf(bed,tv1_area).

dynamic facts:

inside(bed)



admissible(V)?

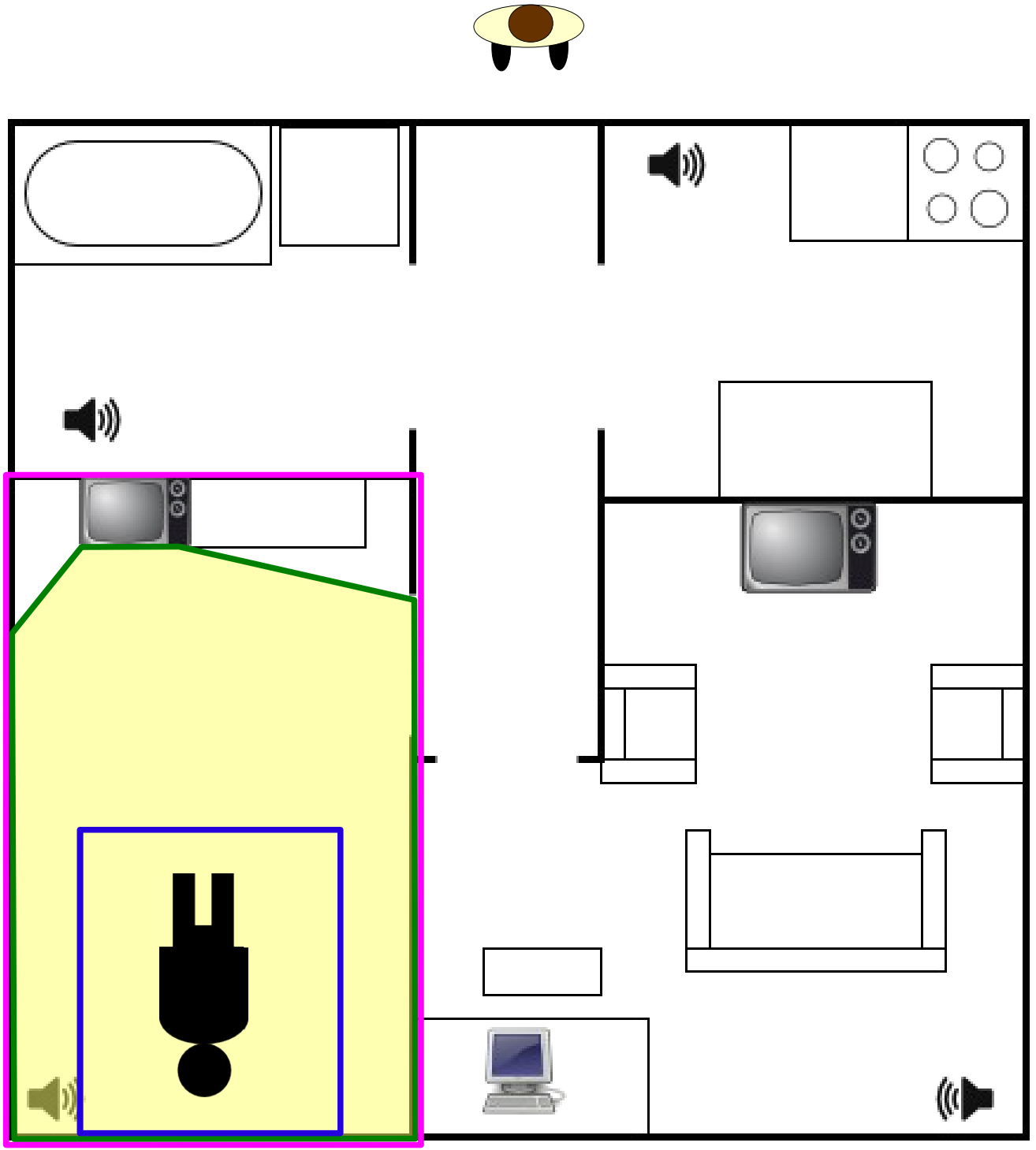
admissible(V) if
videoDevice(V) and
covers(V,A) and
inside(A) and
not(intimate(A)).

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

partOf(bed,tv1_area).

dynamic facts:

inside(bed)



admissible(V)?

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

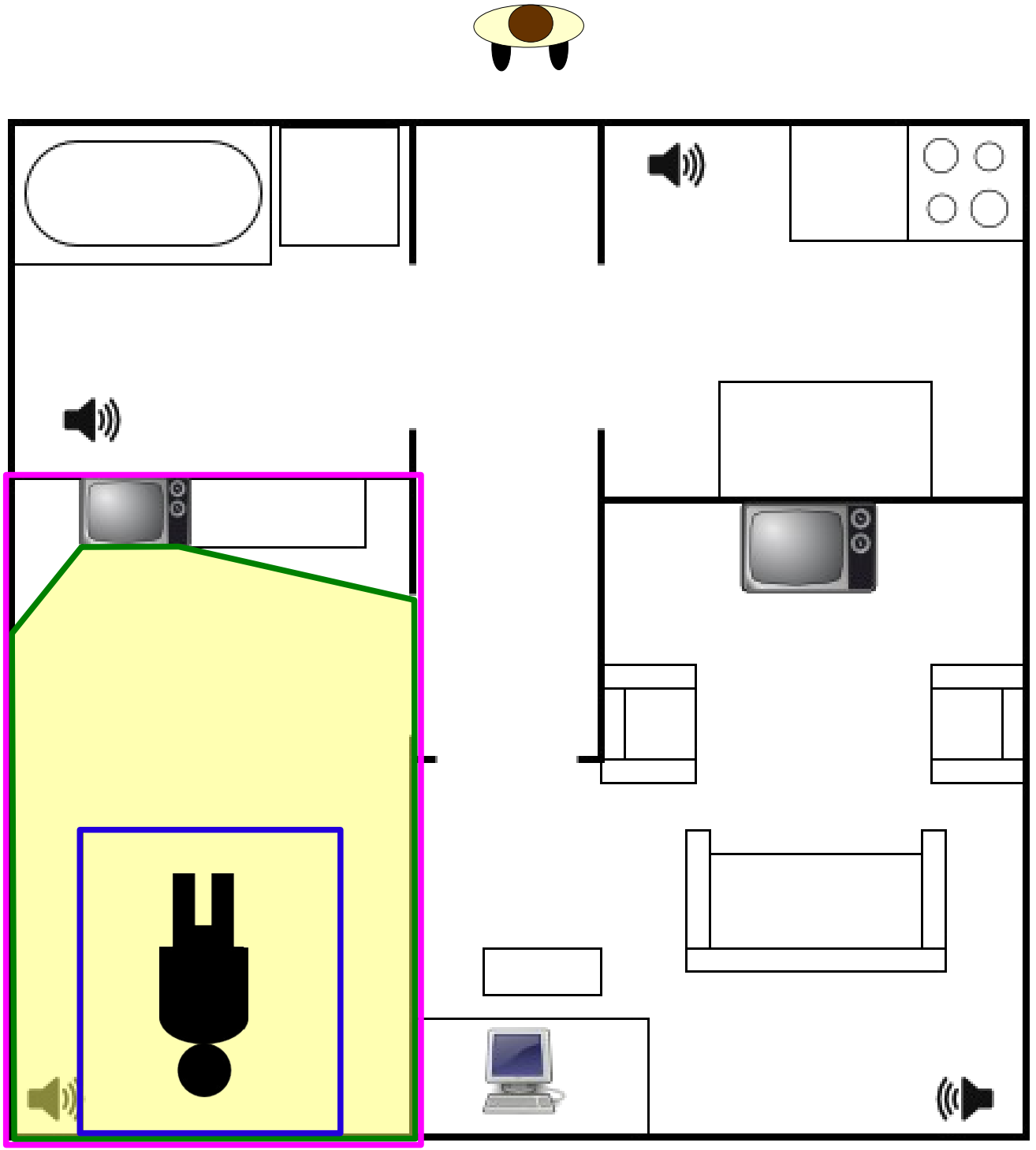
inside(bed) if
partOf(bed,tv1_area) and
inside(tv1_area).

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

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

dynamic facts:

inside(bed)

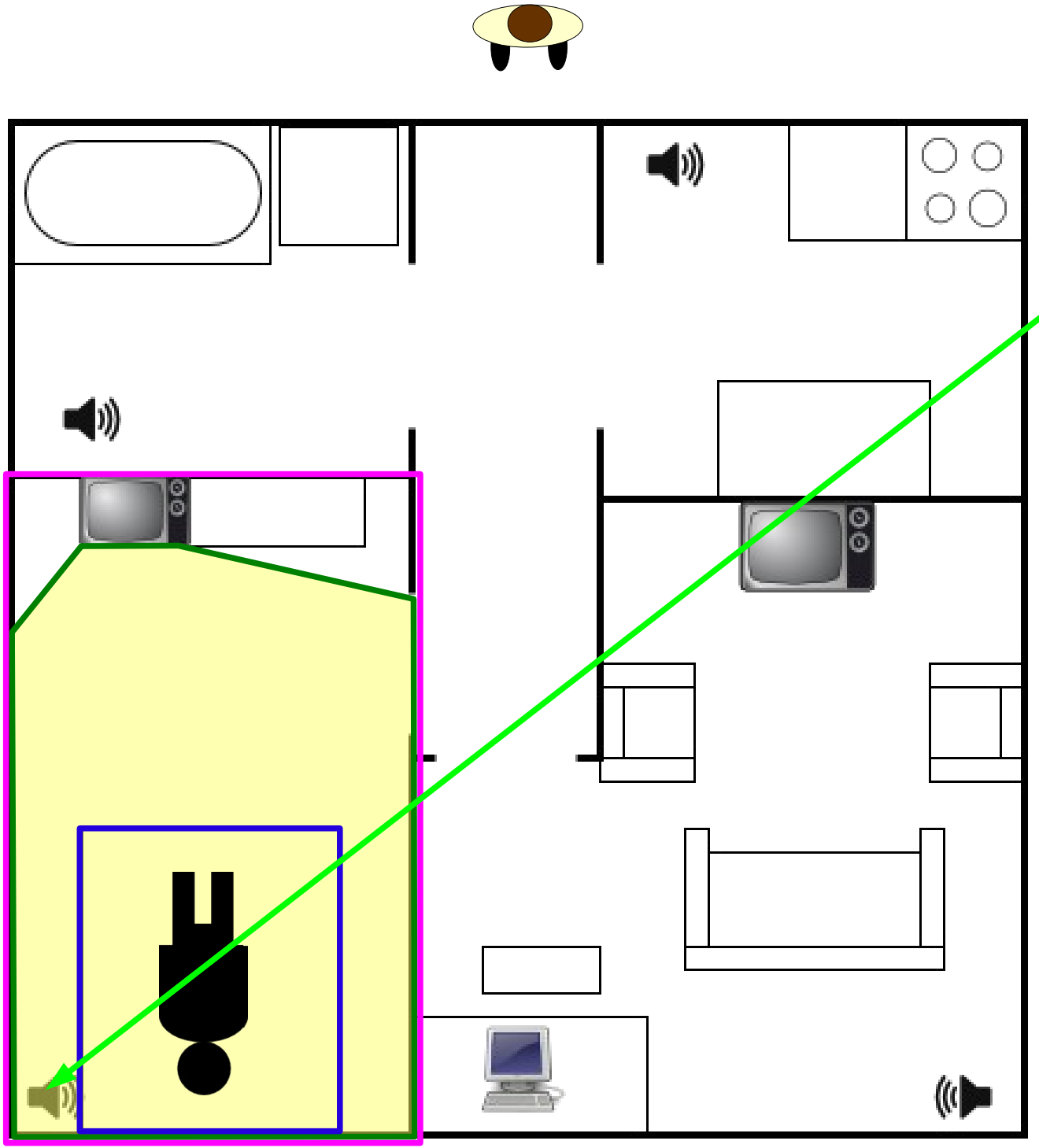


admissible(V)?

```
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(sp1)!

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)

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.