Chapter 5: Representing Knowledge

- Lecture 1 Knowledge representation issues. Defining a solution. Choosing a representation. Mapping from a problem to a representation.
- Lecture 2 Choosing objects and relations. Semantic networks, frames, primitive and derived relations.
- Lecture 3 Knowledge sharing, ontologies.

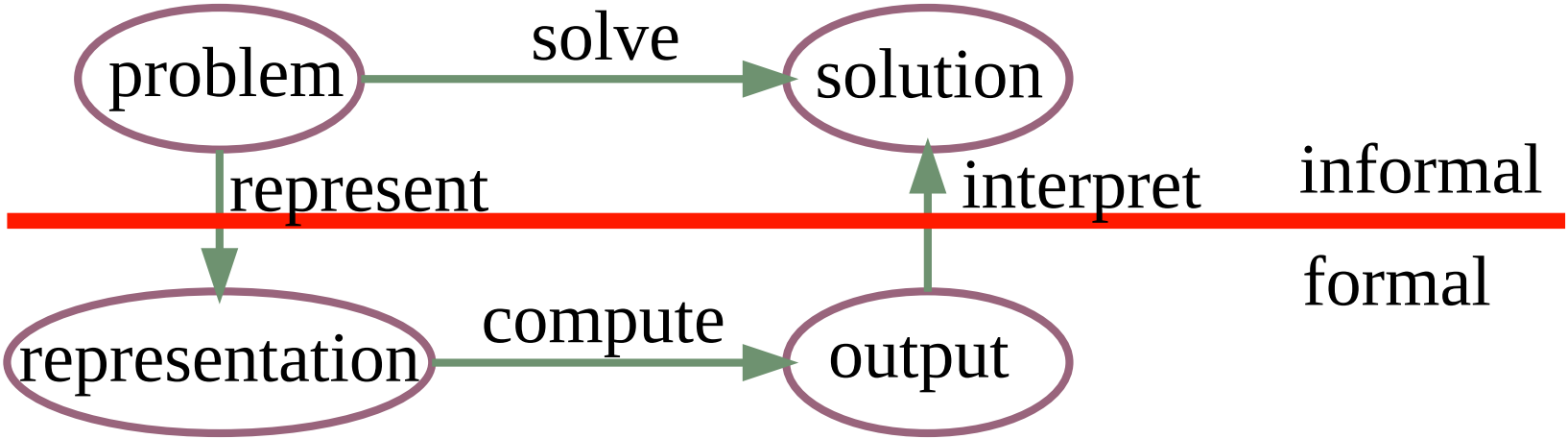
Representing Knowledge

Given a problem to solve, how do you solve it?

- What is a solution to the problem?
- What do you need in the language to represent the problem?
- How can you map from the informal problem description to a representation of the problem?
- What distinctions in the world are important to solve the problem?
- What knowledge is required?
- What level of detail is required?

- What reasoning strategies are appropriate?
- Is worst-case performance or average-case performance the critical time to minimize?
- Is it important for a human to understand how the answer was derived?
- How can you acquire the knowledge from experts or from experience?
- How can the knowledge be debugged, maintained, and improved?

Knowledge representation framework



Defining a Solution

- Given an informal description of a problem, you need to determine what would constitute a solution.
- Typically much is left unspecified, but the unspecified parts can't be filled in arbitrarily.
- Much work in AI is motivated by common-sense reasoning. You want the computer to be able to make common-sense conclusions about the unstated assumptions.

Quality of Solutions

Does it matter if the answer is wrong or answers are missing?

Classes of solution:

Optimal solution the best solution according some measure of solution quality.

Satisficing solution one that is good enough, according to some description of which solutions are adequate.

Approximately optimal solution one whose measure of quality is close to the best theoretically possible.

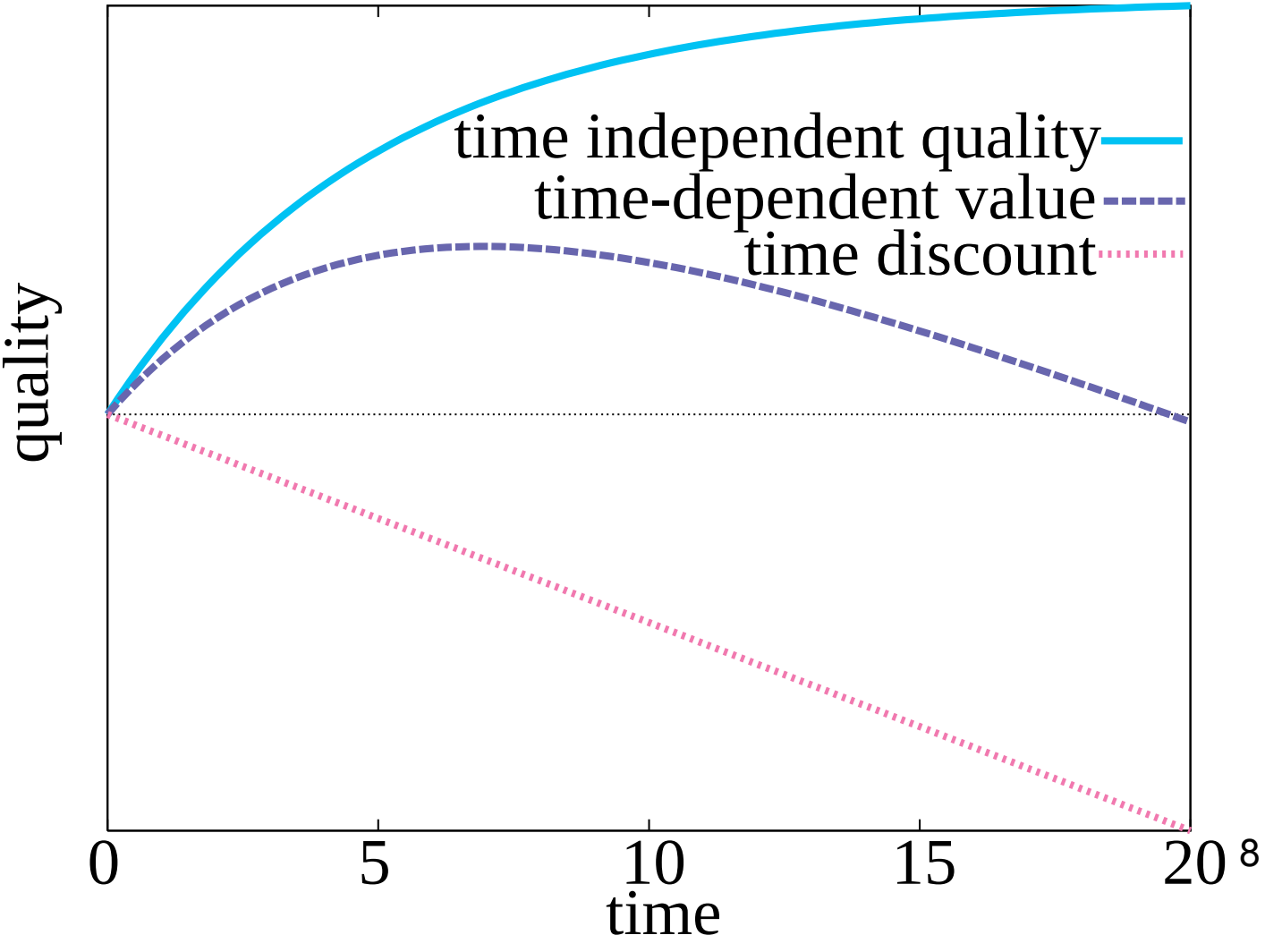
Probable solution one that is likely to be a solution.

Decisions and Outcomes

- Good decisions can have bad outcomes. Bad decisions can have good outcomes.
- Information can be valuable because it leads to better decisions: value of information.
- You have to trade off computation time and solution quality: an anytime algorithm can provide a solution at any time; given more time it can produce better solutions.

You don't only need to be concerned about finding the right answer, but about acquiring the appropriate information, and computing it in a timely manner.

Solution quality and computation time



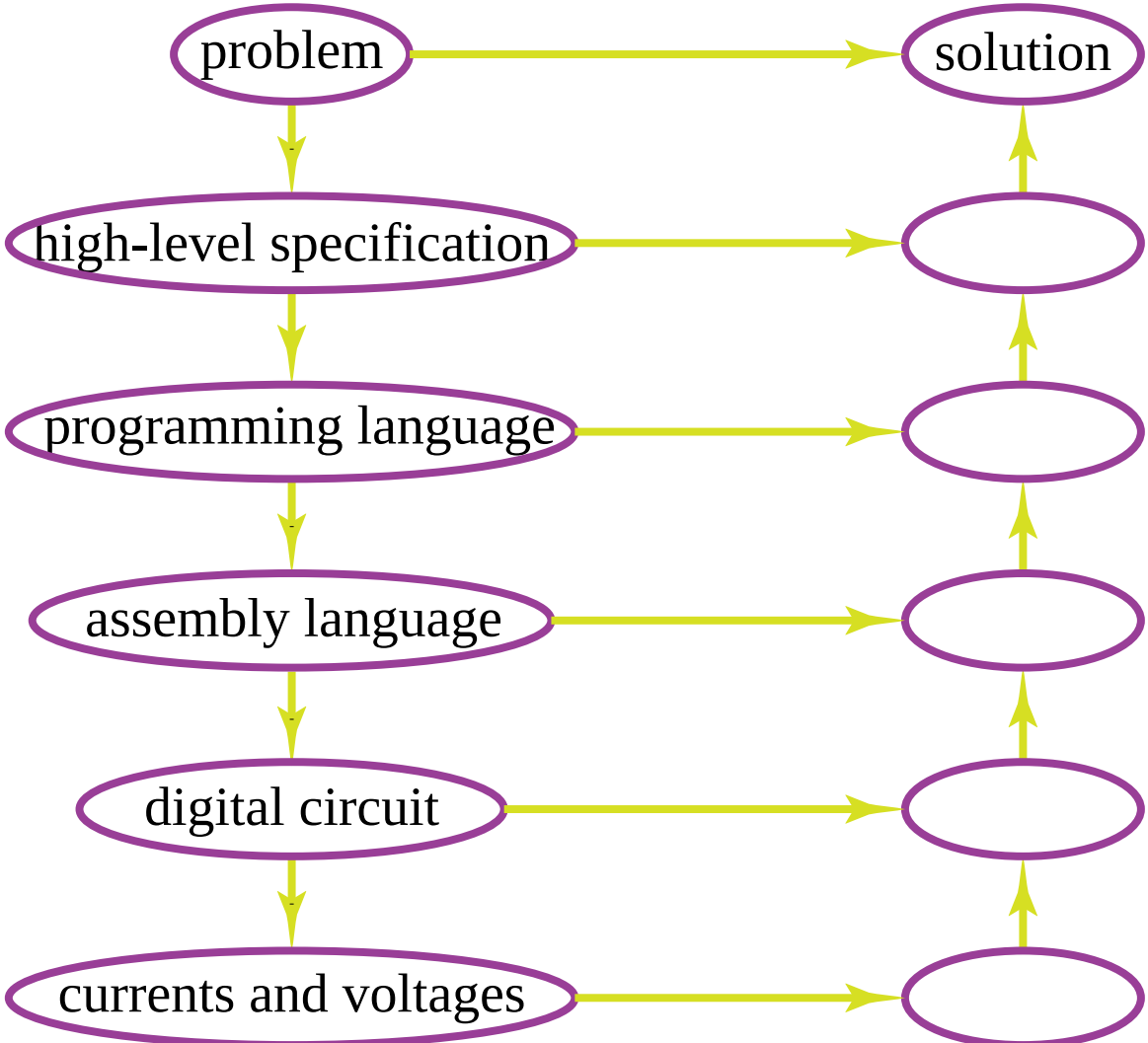
Choosing a Representation Language

You need to represent a problem to solve it on a computer.

Example representations: C++, CILog/Prolog, English

A logic is a language + specification of what follows from input in that language.

Hierarchy of representations



Knowledge & Symbol Levels

Two levels of abstraction seem to be common among biological and computational entities:

Knowledge level in terms of an agent's knowledge and goals

Symbol level in terms of what symbols the agent is manipulating.

The knowledge level is about the external world to the agent.

The symbol level is about what symbols an agent uses to implement the knowledge level.

Mapping from Problem to Representation

- What level of abstraction of the problem do you want to have to represent?
- What objects and relations in the world do you want to represent?
- How can you represent the knowledge to ensure that the representation is natural, modular, and maintainable?
- How can you acquire the information from data, sensing, experience, or other agents?

Choosing a level of abstraction

- A high-level description is easier for a human to specify and understand.
- A low-level description can be more accurate and more predictive. High-level descriptions abstract away details that may be important for actually solving the problem.
- The lower the level, the more difficult it is to reason with.
- You may not know the information needed for a low-level description.

It is sometime possible to use multiple levels of abstraction.

Choosing Objects and Relations

How to represent: “Pen #7 is red.”

Choosing Objects and Relations

How to represent: "Pen #7 is red."

$red(pen_7)$. It's easy to ask "What's red?"

Can't ask "what is the color of pen_7?"

Choosing Objects and Relations

How to represent: "Pen #7 is red."

$red(pen_7)$. It's easy to ask "What's red?"

Can't ask "what is the color of pen_7?"

$color(pen_7, red)$. It's easy to ask "What's red?"

It's easy to ask "What is the color of pen_7?"

Can't ask "What property of pen_7 has value *red*?"

Choosing Objects and Relations

How to represent: "Pen #7 is red."

$red(pen_7)$. It's easy to ask "What's red?"

Can't ask "what is the color of pen_7?"

$color(pen_7, red)$. It's easy to ask "What's red?"

It's easy to ask "What is the color of pen_7?"

Can't ask "What property of pen_7 has value *red*?"

$prop(pen_7, color, red)$. It's easy to ask all these questions.

Choosing Objects and Relations

How to represent: "Pen #7 is red."

$red(pen_7)$. It's easy to ask "What's red?"

Can't ask "what is the color of pen_7?"

$color(pen_7, red)$. It's easy to ask "What's red?"

It's easy to ask "What is the color of pen_7?"

Can't ask "What property of pen_7 has value *red*?"

$prop(pen_7, color, red)$. It's easy to ask all these questions.

prop(*Object*, *Attribute*, *Value*) is the only relation needed:

object-attribute-value representation

Universality of *prop*

To represent “a is a parcel”

- *prop*(*a*, *is_a*, *parcel*), where *is_a* is a special attribute
- *prop*(*a*, *parcel*, *true*), where *parcel* is a Boolean attribute

Reification

- To represent *scheduled*(*cs*422, 2, 1030, *cc*208). “section 2 of course *cs*422 is scheduled at 10:30 in room *cc*208.”
- Let *b*123 name the booking:

  prop(*b*123, *course*, *cs*422).

  prop(*b*123, *section*, 2).

  prop(*b*123, *time*, 1030).

  prop(*b*123, *room*, *cc*208).
- We have reified the booking.
- Reify means: to make into an object.

Semantics Networks

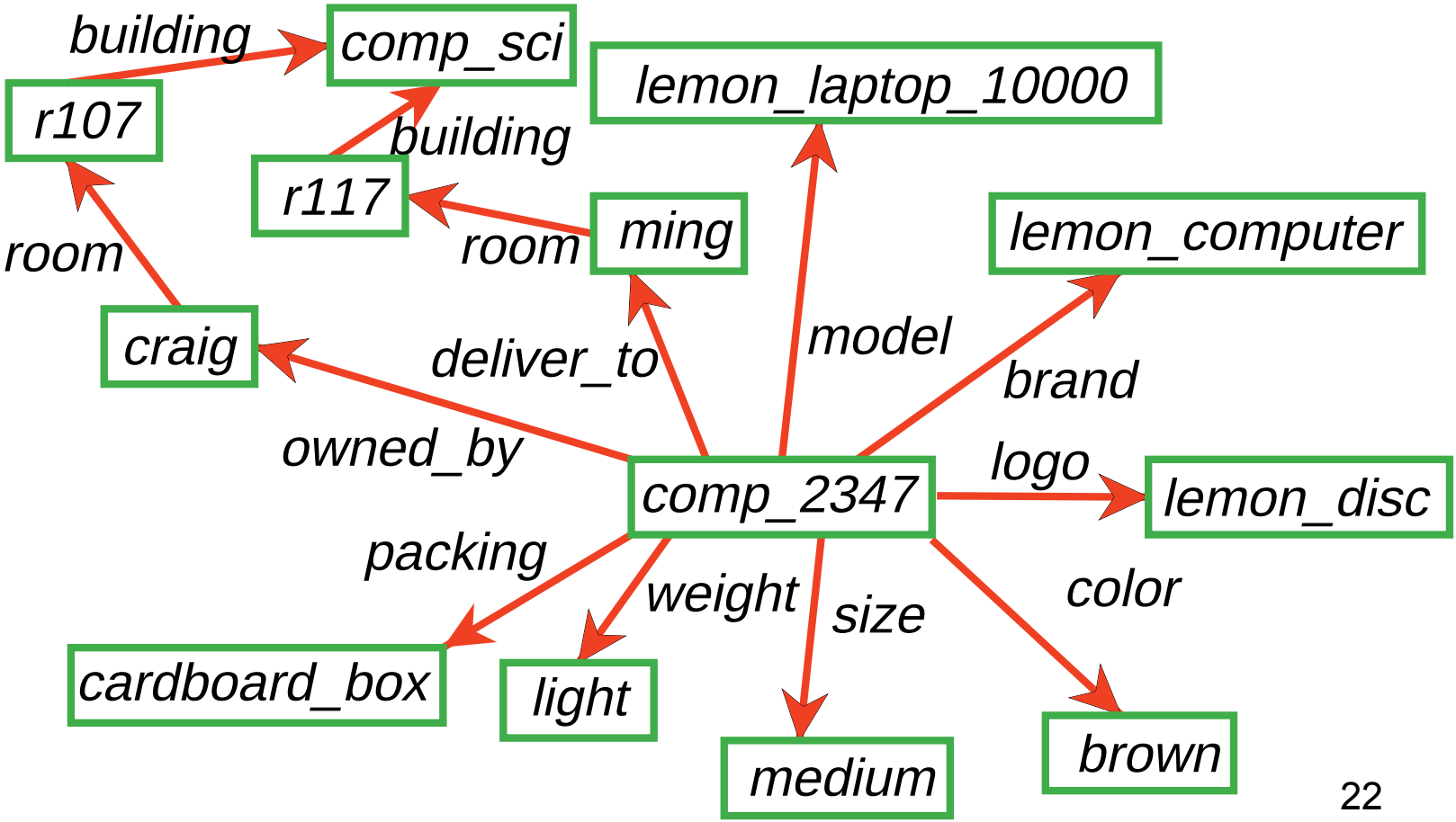
When you only have one relation, *prop*, it can be omitted without loss of information.

Write

$$prop(Obj, Att, Value)$$

as

An Example Semantic Network



Equivalent Logic Program

prop(*comp*_2347, *owned*_*by*, *craig*).

prop(*comp*_2347, *deliver*_*to*, *ming*).

prop(*comp*_2347, *model*, *lemon*_*laptop*_10000).

prop(*comp*_2347, *brand*, *lemon*_*computer*).

prop(*comp*_2347, *logo*, *lemon*_*disc*).

prop(*comp*_2347, *color*, *brown*).

prop(*craig*, *room*, *r*107).

prop(*r*107, *building*, *comp*_*sci*).

⋮

Frames

The properties and values for a single object can be grouped together into a frame.

We can write this as a list of *attribute* = *value* or *slot* = *filler*.

$$
\begin{aligned}
&[owned\_by = craig,\\
&deliver\_to = ming,\\
&model = lemon\_laptop\_10000,\\
&brand = lemon\_computer,\\
&logo = lemon\_disc,\\
&color = brown,\\
&\ldots]
\end{aligned}
$$

Primitive versus Derived Relations

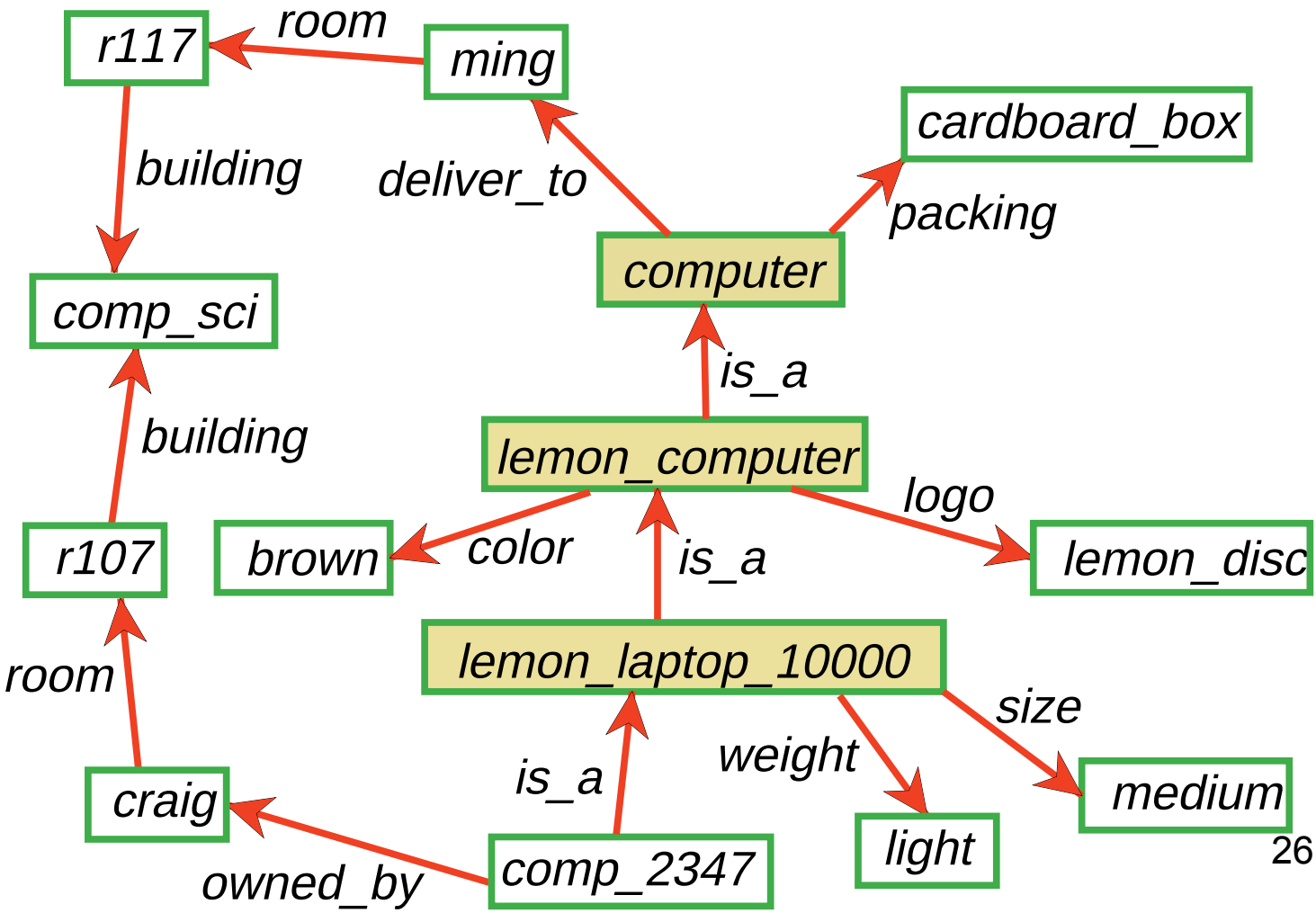
Primitive knowledge is that which is defined explicitly by facts.

Derived knowledge is knowledge defined by rules.

Example: All lemon laptops may have have *size* = *medium*. Associate this property with the class, not the individual.

Allow a special attribute *is_a* between an individual and a class or between two classes that allows for property inheritance.

A Structured Semantic Network



Logic of Property Inheritance

An arc $\xrightarrow{p} n$ from a class c means every individual in the class has value n of attribute p:

$$prop(Obj, p, n) \leftarrow$$
$$\quad prop(Obj, is\_a, c).$$

Example:

$$prop(X, weight, light) \leftarrow$$
$$\quad prop(X, is\_a, lemon\_laptop\_10000).$$
$$prop(X, is\_a, lemon\_computer) \leftarrow$$
$$\quad prop(X, is\_a, lemon\_laptop\_10000).$$

Multiple Inheritance

- An individual is usually a member of more than one class. For example, the same person may be a mother, a teacher, a football coach,….
- The individual can inherit the properties of all of the classes it is a member of: multiple inheritance.
- If there are default values, we can have a problem when an individual inherits conflicting defaults from the different classes: multiple inheritance problem.

Choosing Primitive and Derived Relations

- Associate an attribute value with the most general class with that attribute value.
- Don’t associate contingent properties of a class with the class. For example, if all of current computers just happen to be brown.
- Axiomatize in the causal direction. You want knowledge that is stable as the world changes.

Knowledge Sharing

- If more than one person is building a knowledge base, they must be able to share the conceptualization.
- A conceptualization is a map from the problem domain into the representation. A conceptualization specifies:
  - What sorts of objects are being modelled
  - The vocabulary for specifying objects, relations and attributes
  - The meaning or intention of the relations or attributes
- An ontology is a specification of a conceptualization.

Semantic Web

- Ontologies are published on the web in machine readable form and are publically readable.
- Builders of knowledge bases or web sites adhere to and refer to a published ontology:
  - the same symbol means the same thing across the various web sites that obey the ontology.
  - if someone wants to refer to some other object or relation, the ontology is expanded. The community needs to agree to the new terminology.

Semantic Information Processing Based on the Semantic Web

- **Web information is linked to a web-based ontology by means of a XML annotation**
- **An ontology defines semantic relations between identifiers:**
  - synonyms, subclasses, superclasses
  - classes vs. individuals
  - properties of classes and relations between classes
  - possible values for properties
- **An inference system processes web information based on the ontology and derives implicit knowledge**
- **For content-based information processing applications make use of the inference system**

Ontology Definitions with DAML+OIL (1)

(http://www.daml.org/2001/03/daml+oil)

```
<daml:Class rdf:ID="Animal">
    <rdfs:label>Animal</rdfs:label>
    <rdfs:comment>
        This class of animals is illustrative of a number of ontological idioms.
    </rdfs:comment>
</daml:Class>

<daml:Class rdf:ID="Male">
    <rdfs:subClassOf rdf:resource="#Animal"/>
</daml:Class>

<daml:Class rdf:ID="Female">
    <rdfs:subClassOf rdf:resource="#Animal"/>
    <daml:disjointWith rdf:resource="#Male"/>
</daml:Class>

<daml:Class rdf:ID="Man">
    <rdfs:subClassOf rdf:resource="#Person"/>
    <rdfs:subClassOf rdf:resource="#Male"/>
</daml:Class>
```

Ontology Definitions with DAML+OIL (2)

```
<daml:Class rdf:ID="Person">
    <rdfs:subClassOf rdf:resource="#Animal"/>
    <rdfs:subClassOf>
        <daml:Restriction>
            <daml:onProperty rdf:resource="#hasParent"/>
            <daml:toClass rdf:resource="#Person"/>
        </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction daml:cardinality="1">
            <daml:onProperty rdf:resource="#hasFather"/>
        </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction>
            <daml:onProperty rdf:resource="#shoesize"/>
            <daml:minCardinality>1</daml:minCardinality>
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>
```

Ontology Definitions with DAML+OIL (3)

```
<daml:ObjectProperty rdf:ID="hasParent">
    <rdfs:domain rdf:resource="#Animal"/>
    <rdfs:range rdf:resource="#Animal"/>
</daml:ObjectProperty>

<daml:DatatypeProperty rdf:ID="age">
    <rdfs:comment>
        age is a DatatypeProperty whose range is xsd:decimal.
        age is also a UniqueProperty (can only have one age)
    </rdfs:comment>
    <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#nonNegativeInteger"/>
</daml:DatatypeProperty>
```

Ontology Definitions with DAML+OIL (4)

```
<daml:Class rdf:about="#Person">
    <rdfs:subClassOf>
        <daml:Restriction daml:maxCardinalityQ="1">
            <daml:onProperty rdf:resource="#hasOccupation"/>
            <daml:hasClassQ rdf:resource="#FullTimeOccupation"/>
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>

<daml:UniqueProperty rdf:ID="hasMother">
    <rdfs:subPropertyOf rdf:resource="#hasParent"/>
    <rdfs:range rdf:resource="#Female"/>
</daml:UniqueProperty>

<daml:ObjectProperty rdf:ID="hasChild">
  <daml:inverseOf rdf:resource="#hasParent"/>
</daml:ObjectProperty>
```

Inferences Based on Ontologies - Example

Consistency check of E-business internet catalogue:

SPECIAL OFFER:

MMZ100, year 2000, à EUR 75,-

Elsewhere in the internet:

MMZ100 is a Multimedia Center

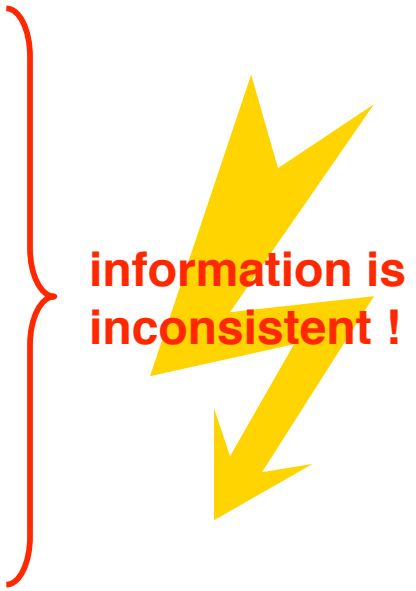
MMZ100 has a list price of DM 150,-

All entertainment systems built before 2002 are sold with 20% rebate on the list price

A Multimedia Center is a special TV set

A TV set is an entertainment system

1 EUR = 1,95583 DM



Challenges of building ontologies

- They can be huge: finding the appropriate terminology for a concept may be difficult.
- How one divides the world can depend on the application. Different ontologies describe the world in different ways.
- People can fundamentally disagree about the appropriate structure.
- Different knowledge bases can use different ontologies.
- To allow KBs based on different ontologies to interoperate, there must be mapping between different ontologies.

- It has to be in user's interests to use an ontology.
- The computer doesn't understand the meaning of the symbols. The formalism can constrain the meaning, but can't define it.

Concept Hierarchy

- The core of an ontology are concept hierachies.
- A concept hierarchy is a tree (or trees) where
  - the nodes correspond to concepts or classes and
  - the parents of a node correspond to a more general concept
  - children of a node are mutually exclusive

Example Concepts in an Ontology

The following are some of the concepts in an ontology for documents.

http://www.cs.umd.edu/projects/plus/DAML/onts/docmnt1.0.daml

homepage	correspondence	publication
letter	periodical	article
book	letter	magazine
journal	document	communication
workshopPaper	journalPaper	discussion
newspaper	PersonalHomepage	Speech