

COMMONSENSE KNOWLEDGE

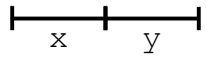
What is commonsense knowledge?

- Throughout, we have talked to “domain knowledge”, knowledge about the situation an agent finds itself in.
- It seems to be relatively easy to program an agent with good detailed knowledge about domains.
- (Dealing with the volume of knowledge is another question.)
- However, it is much harder to give agents the kind of general knowledge about the world that children have.
- (Perhaps because agents don't get to spend as much time learning about the world by interacting with it that children do.)
- Such knowledge is necessary, though, if we want to build general intelligent agents.

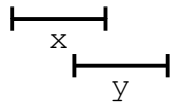
- Typical pieces of commonsense knowledge
 - Two blocks can't be in the same place.
 - Liquid comes out of a cup if you tilt it.
 - Water makes things wet.
 - You can't drive through walls.
 - If you let go of something it falls.
- “What any fool knows”
- AI has looked for compact representations of this.
- Maybe we just need to type it all in.

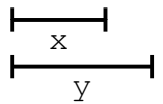
Time

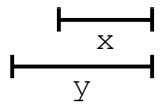
- Lots of ways of representing time:
 - discrete
 - continuous
 - unidirectional
 - bidirectional
 - linear
 - branching
- Formal tools for many of these.
- Consider Allen's interval calculus.

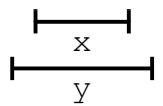

 Meets (x, y) Met_by (y, x)

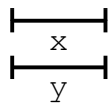

 Before (x, y) After (y, x)


 Overlaps (x, y) Overlapped_by (y, x)


 Starts (x, y) Started_by (y, x)


 Ends (x, y) Ended_by (y, x)


 During (x, y) Contains (y, x)


 Equals (x, y) Equals (y, x)

© 1998 Morgan Kaufman Publishers

- These map into logic

$$\forall x, y \cdot Meets(x, y) \equiv (end(x) = start(y))$$

$$\forall x, y \cdot Before(x, y) \equiv \exists z (Meets(x, z) \wedge Meets(z, y))$$

- And the rest can be defined in a similar manner.
- Then we can describe temporal aspects about the world:

$$\begin{aligned} \forall y \cdot Occurs(Flow, y) \Rightarrow \\ \exists x, z \cdot Occurs(Turn, x) \wedge Occurs(Turn, y) \\ \wedge Overlaps(x, y) \wedge Overlaps(y, z) \end{aligned}$$

- We can then:
 - Use a theorem prover to reason about the world.
 - Use a STRIPS planner to decide what to do.

Taxonomic knowledge

- Often we find that commonsense knowledge comes in the form of taxonomies.
- All *P*s are *Q*s.
- We can represent this as logic.

Laser_printer(snoopy)

$\forall x \cdot \text{Laser_printer}(x) \Rightarrow \text{Printer}(x)$

$\forall X \cdot \text{Printer}(x) \Rightarrow \text{Office_machine}(x)$

- We can infer $\forall x \cdot \text{Laser_printer}(x) \Rightarrow \text{Office_machine}(x)$
- We can inherit properties from superclass to subclass.

Semantic Networks

- Taxonomic reasoning can be more efficient not in logic.
- Developed by Quillian in 1968, for *semantic memory*.
- Models the “associations” between ideas that people maintain.
- Semantic net is a *labelled graph*.
 - nodes in graph represent *objects, concepts, or situations*;
 - arcs in graph represent *relationships between objects*.

Key types of arc:

- $x \xrightarrow{\text{subset}} y$

“ x is a kind of y ” (\subset)

Example: $penguin \xrightarrow{\text{subset}} bird$

- $x \xrightarrow{\text{member}} y$

“ x is a y ”

Example: $opus \xrightarrow{\text{member}} penguin$

- $x \xrightarrow{R} y$

“ x is R -related to y ”

Example: $bill \xrightarrow{\text{friend}} opus$

- Inference is then by traversing arcs.



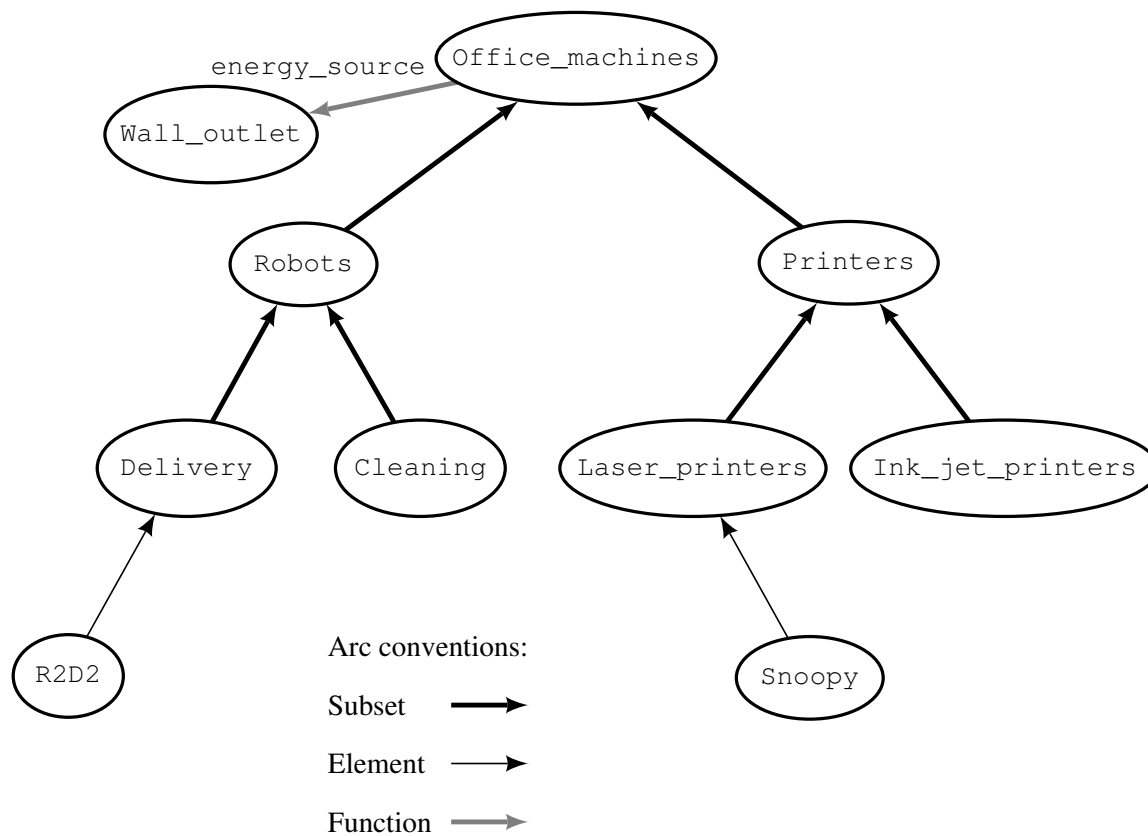
- *Binary* relations are easy and natural to represent.
- Others kinds of relation are harder.
- Unary relations (properties).
Example: “Opus is small”.
- Three place relations.
Example: “Opus brings tequila to the party.”
- Some binary relations are problematic ...
“Opus is larger than Bill.”

- *Quantified* statements are very hard for semantic nets.

Examples:

- “every dog has bitten a postman”
 - “every dog has bitten every postman”
- *Partitioned* semantic nets can represent these.
- Of course, expressions like this are very easy to represent in first order logic.

- Example semantic net:

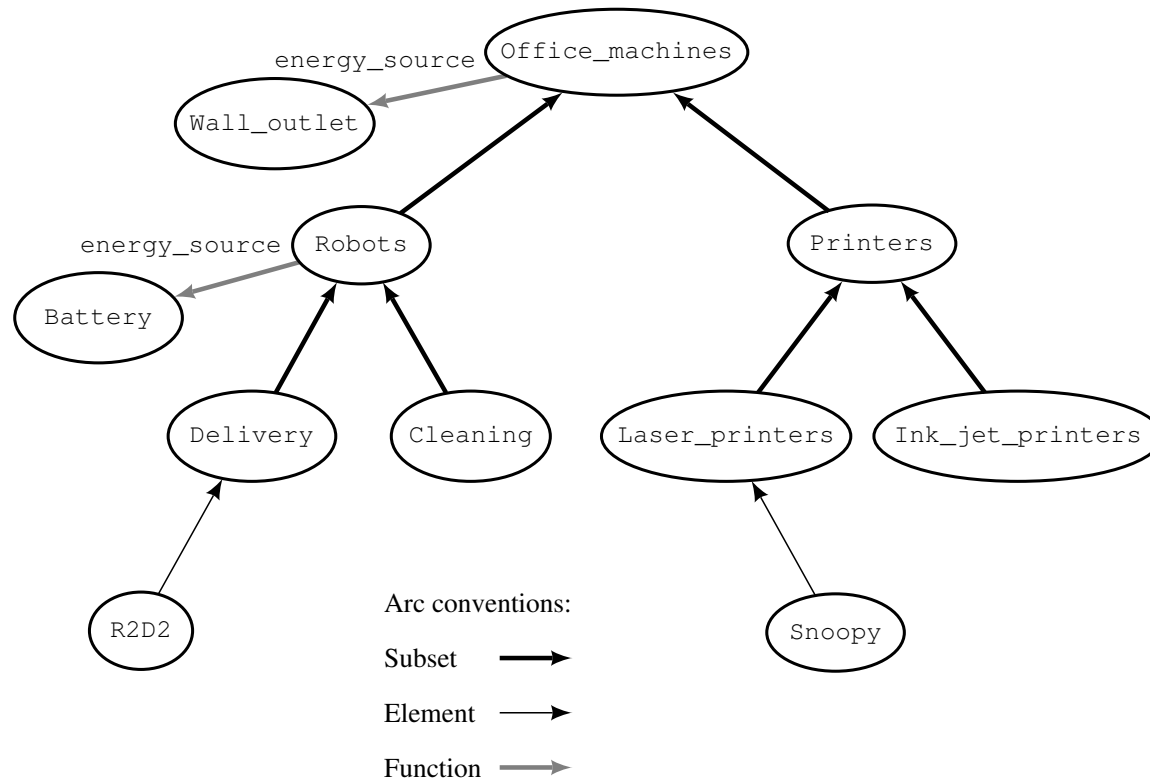


© 1998 Morgan Kaufman Publishers

Default reasoning

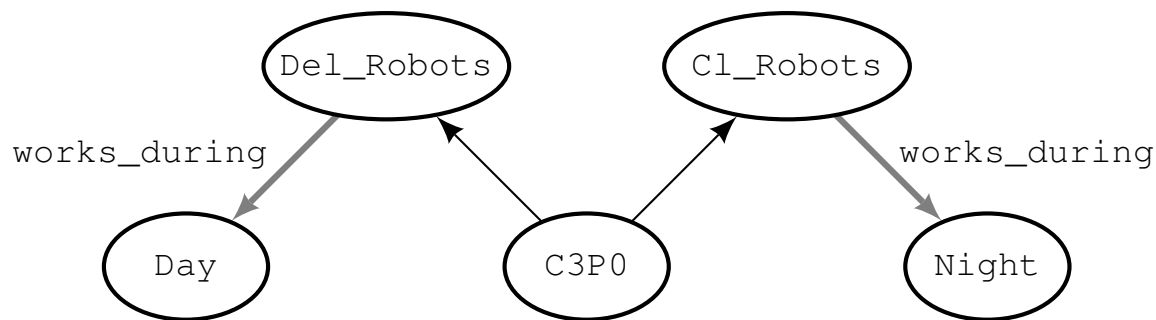
- Reasoning about inheritance is a classic type of *default* reasoning.
- We make an assumption which may later turn out to be wrong.
- This provides us with one way to write commonsense knowledge compactly.
- Can write down the properties of superclasses and let subclasses inherit them.
- Thus, we don't need to represent the fact that Snoopy and R2D2 connect to wall outlets.

- We can also deal with inheritance conflicts:



© 1998 Morgan Kaufman Publishers

- However, coming up with general mechanisms for resolving these conflicts is hard.
- When does C3P0 work?



© 1998 Morgan Kaufman Publishers

- Can add priorities/preferences to deal with this.

Frames

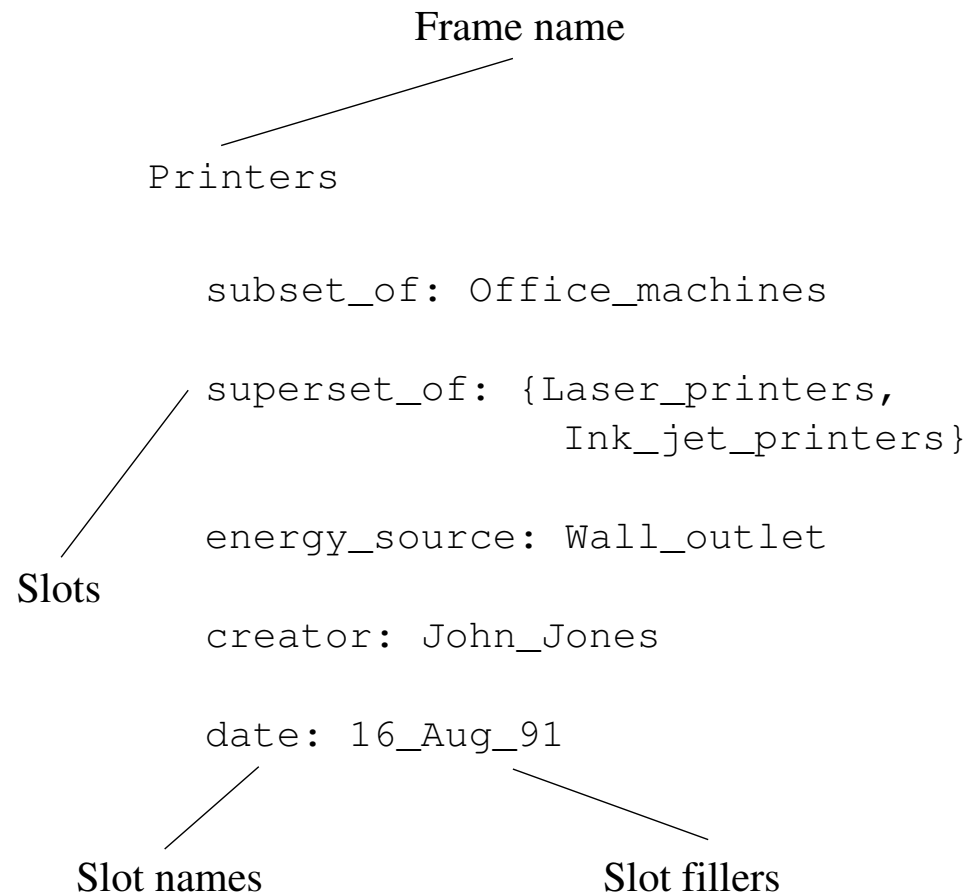
- Frames are a kind of *structured* knowledge representation mechanism.
- All information relevant to a particular concept is stored in *frame* which resembles C struct, PASCAL record, Java object...
- Each frame has a number of *slots*.
- Each slot may be *filled* by:
 - a value;
 - a pointer to another frame;
 - a procedure.
- Slots may have *default values* associated with them.
- Frames = OO!

- Frames are typically used to represent the *properties* of objects, and the relationships between them.
- Frames may represent:
 - *generic concepts* (cf classes) or
 - *specific items* (cf objects).
- Most important kind of link between frames:

is-a

- Facilitates reasoning about object properties.
- Allows *default values* to be *inherited*.

- Example frame system:



© 1998 Morgan Kaufman Publishers

- How to reason with frame systems?
- Easy to answer questions such as
is x a y?
Simply follow the *is-a* links.
- Example: Is snoopy a laser printer.
- (Problem of *multiple inheritance* — Nixon diamond.)
- Also useful for *default* reasoning.
Simply *inherit* all default values that are not explicitly provided.
- Example: Does snoopy the printer have a wall outlet?

- *Scripts* are a variant of frames, for representing *stereotypical sequences of events*.
- A script is thus a frame with a set of prescribed slots, for example:
 - Some initial conditions;
 - Some final conditions;
 - Some state description;
 - Some actions; and
 - Some actors
- The structure of the script is heavily domain dependent.

- Example:

SCRIPT

Name: RESTAURANT

Roles: Customer, Waiter, Cook, Cashier

Entry condition: Customer is hungry

Props: Food, table, money, menu, tip

Events:

- 1/ Customer enters restaurant

- 2/ Customer goes to table

- 3/ Waiter brings menu

- 4/ Customer orders food

- 5/ Waiter brings food

- 6/ Customer eats food

...

...

10/ Customer leaves restaurant

Main concept: 6

Results: Customer not hungry,
Customer has less money,
Restaurant has more money,
Waiter gets tip

- Scripts developed by Roger Schank for *understanding stories*.
- Used to help *understand language*.
- Scripts provide *context* information without which sentences cannot be understood:
 - sentences are not unconstrained sequences of words;
 - stories are not unconstrained sequences of sentences.
- Schank developed SAM (Script Applier Mechanism) that could *fill in gaps* in stories.
- Also able to “explain” elements of stories, e.g., people get upset or angry when story deviates from script.

Problems with Frames & Semantic Nets

- Both frames and semantic nets are essentially *arbitrary*.
- Both are useful for representing certain sorts of knowledge.
- But both are essentially *ad hoc* — lack precise meaning, or *semantics*.
- Inference procedures poorly defined & justified.
- The *syntax* of KR scheme is *irrelevant*.
- *Logic* generalises these schemes... and that is both an advantage and a disadvantage.

- Logic has a(nother) problem, it is inherently *monotonic*
- Once we have:

$$\Delta \vdash \phi$$

then

$$\Delta \cup \{\psi\} \vdash \phi$$

whatever ψ may be.

- So, once we know that C3P0 is a delivery robot, and so works during the day nothing can refute this ...
- ...even when we are told that as a night delivery robot he works at night.

- Trying to make logic *nonmonotonic* has consumed a lot of work over the last 20 years.
- One of the best solutions (there is no one best solution) is the use of *argumentation*.
- We establish whether ϕ holds by looking at the reasons (arguments) that support it:

$$(\{a, a \Rightarrow b, b \Rightarrow c\}, c)$$

- We deal with conflict between deductions through the notion of *acceptability*
- Acceptable arguments are ones not attacked, or ones whose attackers are defeated.

Summary

- This lecture introduced the idea of commonsense knowledge in AI.
- It then briefly surveyed representations of time and taxonomic knowledge, before talking about the general problem of nonmonotonic reasoning.
- Finally the lecture mentioned argumentation as a general approach to nonmonotonic reasoning.