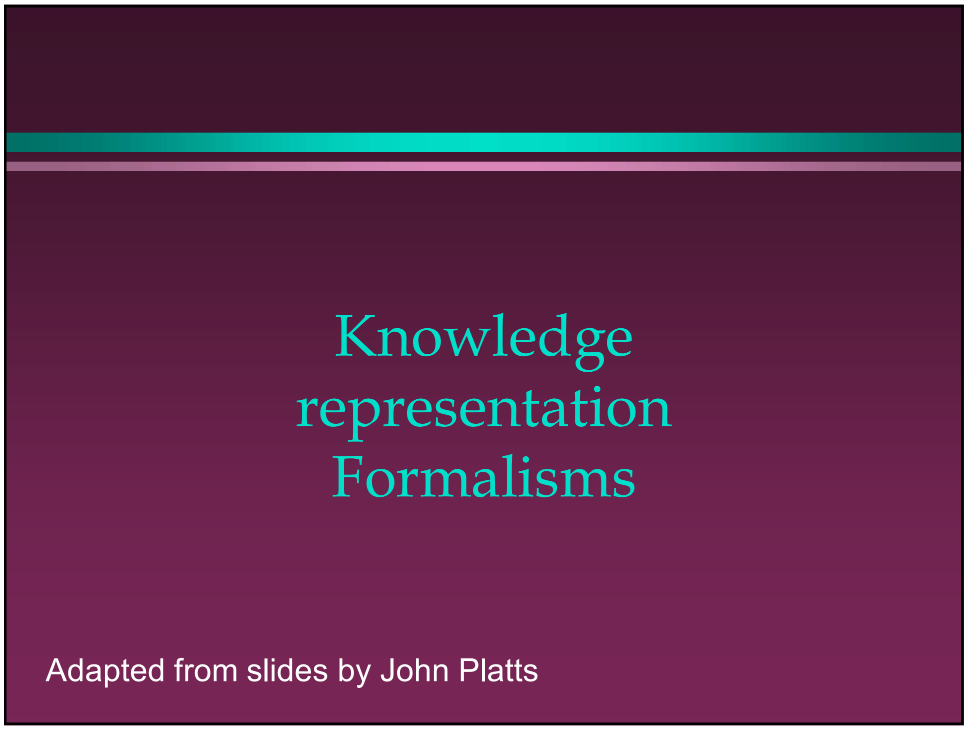
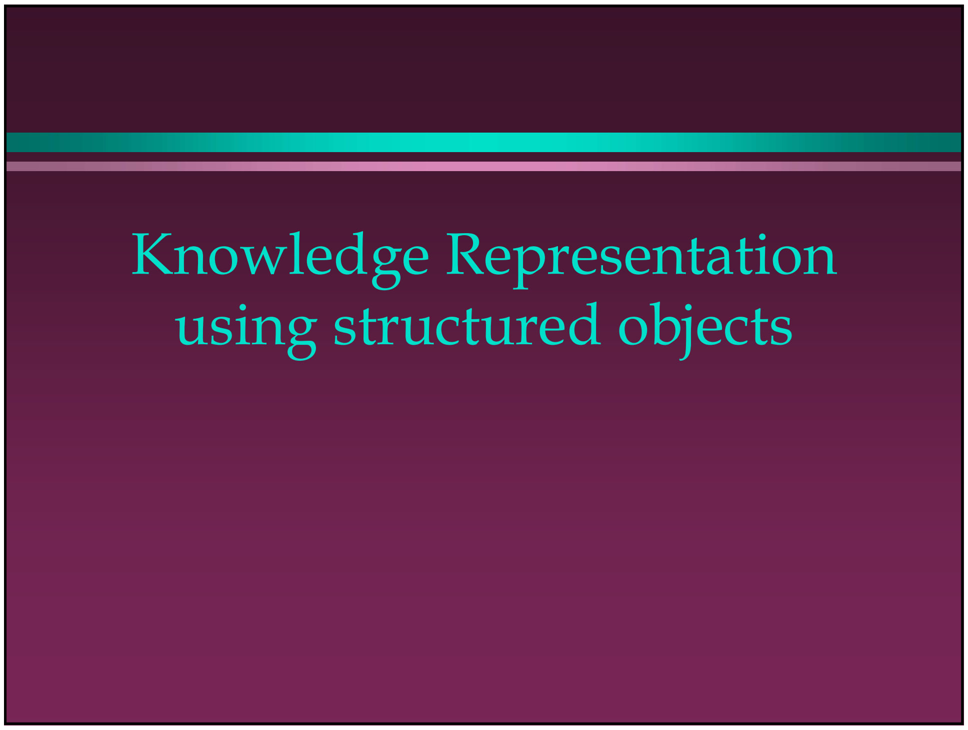


Knowledge representation Formalisms

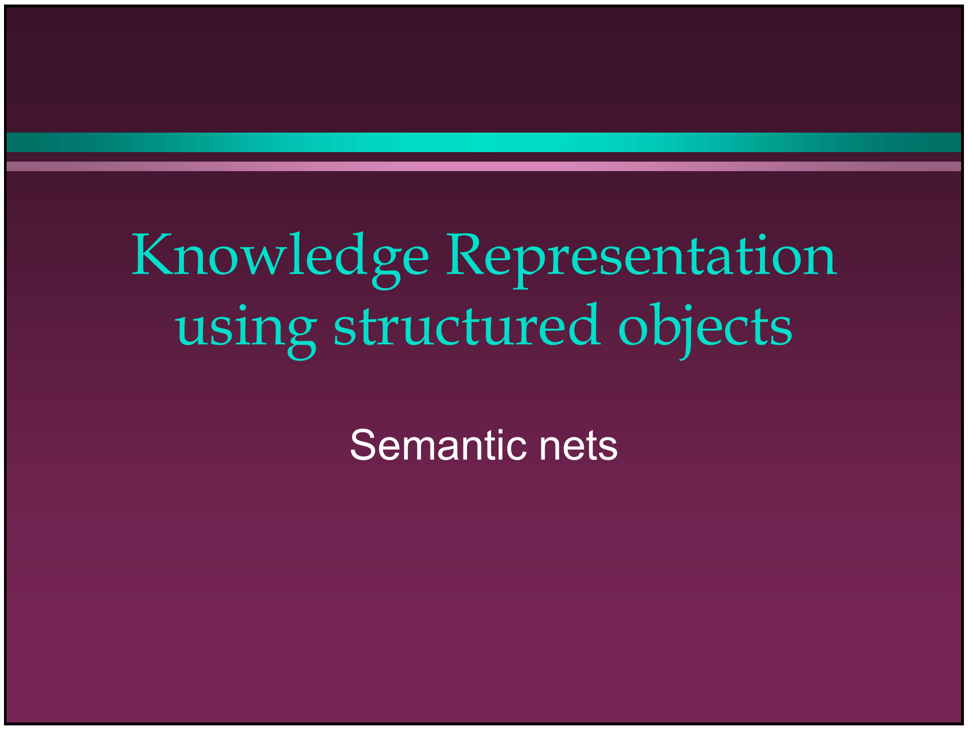
Adapted from slides by John Platts

1



Knowledge Representation using structured objects

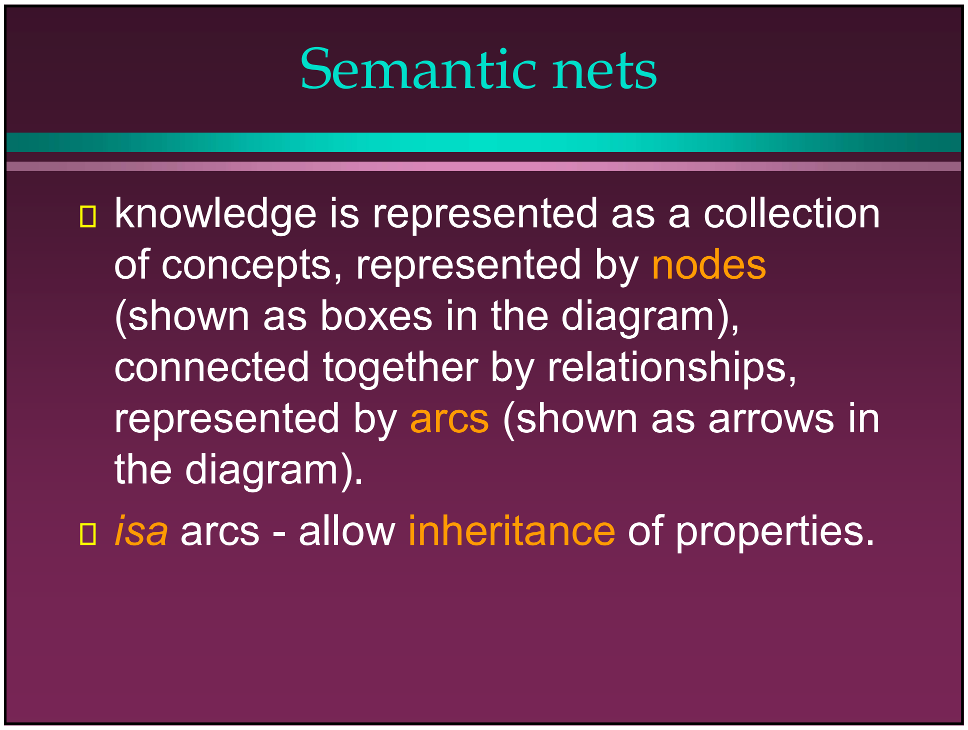
2



Knowledge Representation using structured objects

Semantic nets

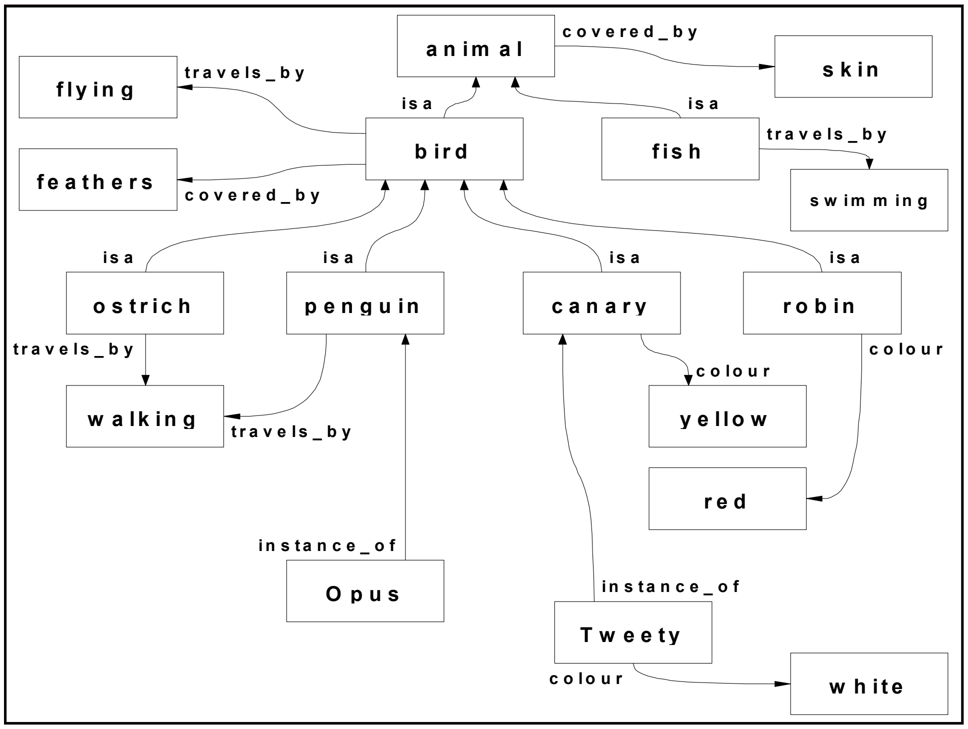
3



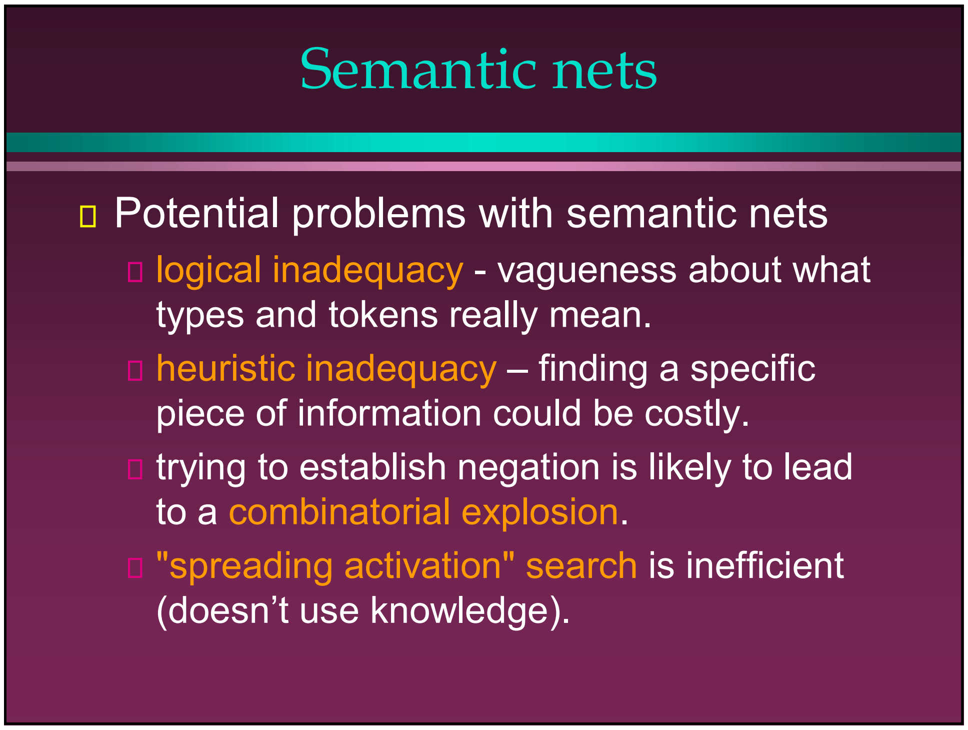
Semantic nets

- knowledge is represented as a collection of concepts, represented by nodes (shown as boxes in the diagram), connected together by relationships, represented by arcs (shown as arrows in the diagram).
- *isa* arcs - allow inheritance of properties.

4



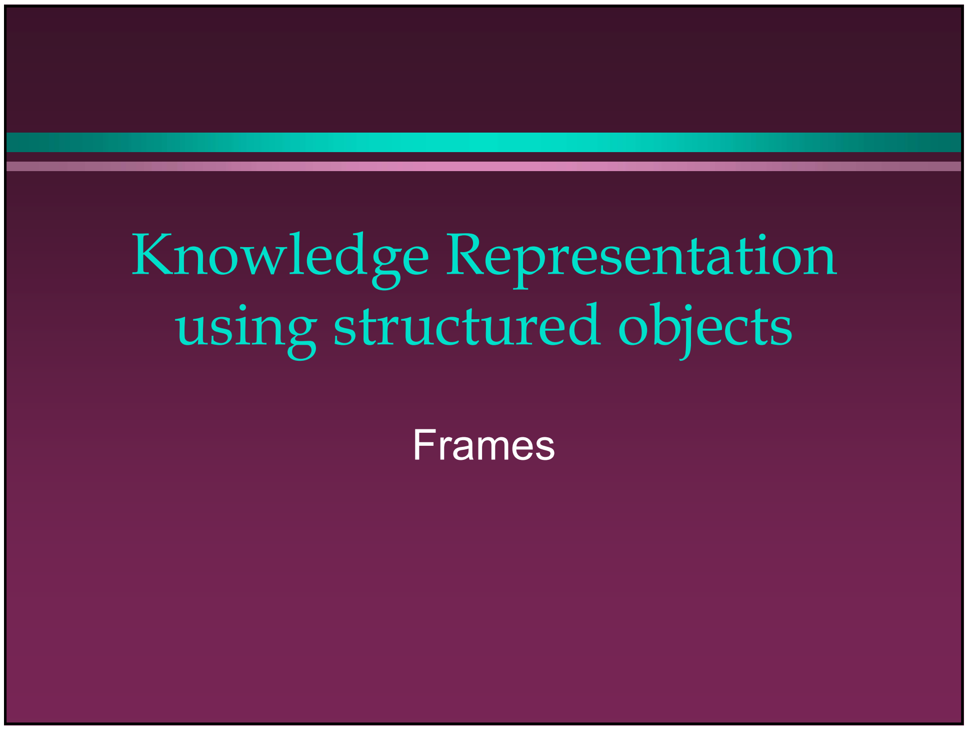
5



Semantic nets

- Potential problems with semantic nets
  - logical inadequacy - vagueness about what types and tokens really mean.
  - heuristic inadequacy – finding a specific piece of information could be costly.
  - trying to establish negation is likely to lead to a combinatorial explosion.
  - "spreading activation" search is inefficient (doesn’t use knowledge).

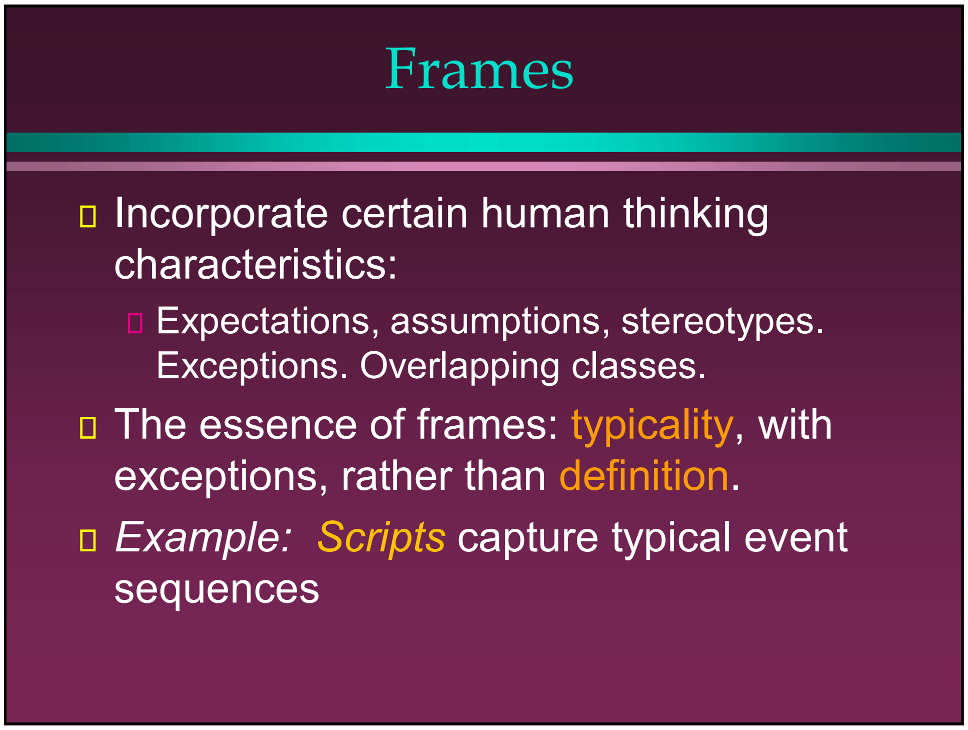
6



Knowledge Representation using structured objects

Frames

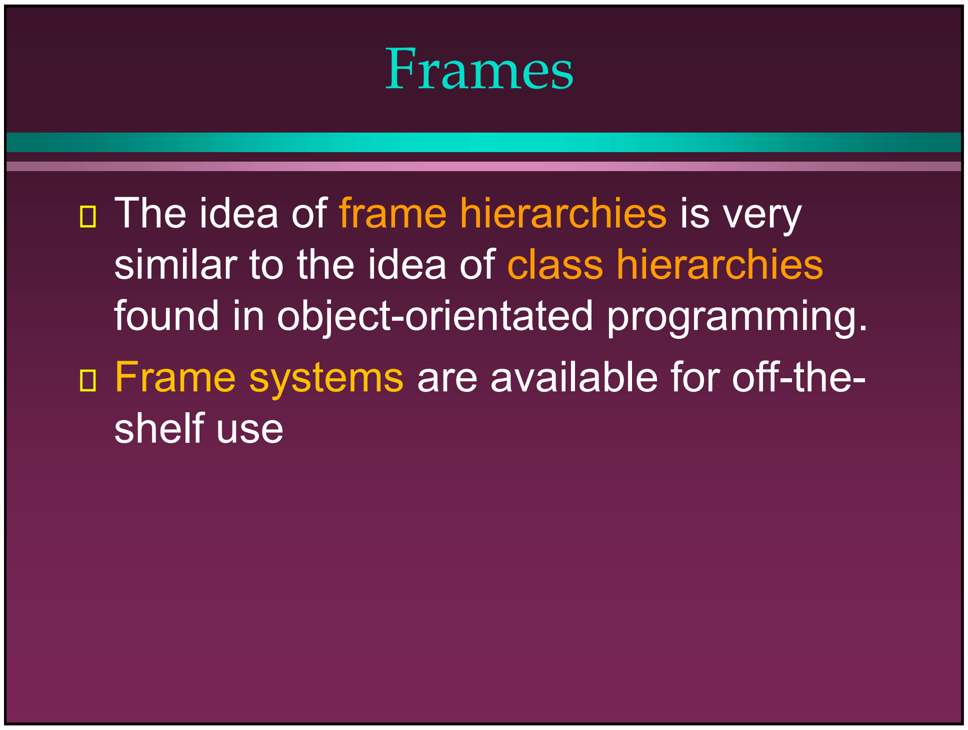
9



Frames

- Incorporate certain human thinking characteristics:
  - Expectations, assumptions, stereotypes. Exceptions. Overlapping classes.
- The essence of frames: typicality, with exceptions, rather than definition.
- *Example: Scripts* capture typical event sequences

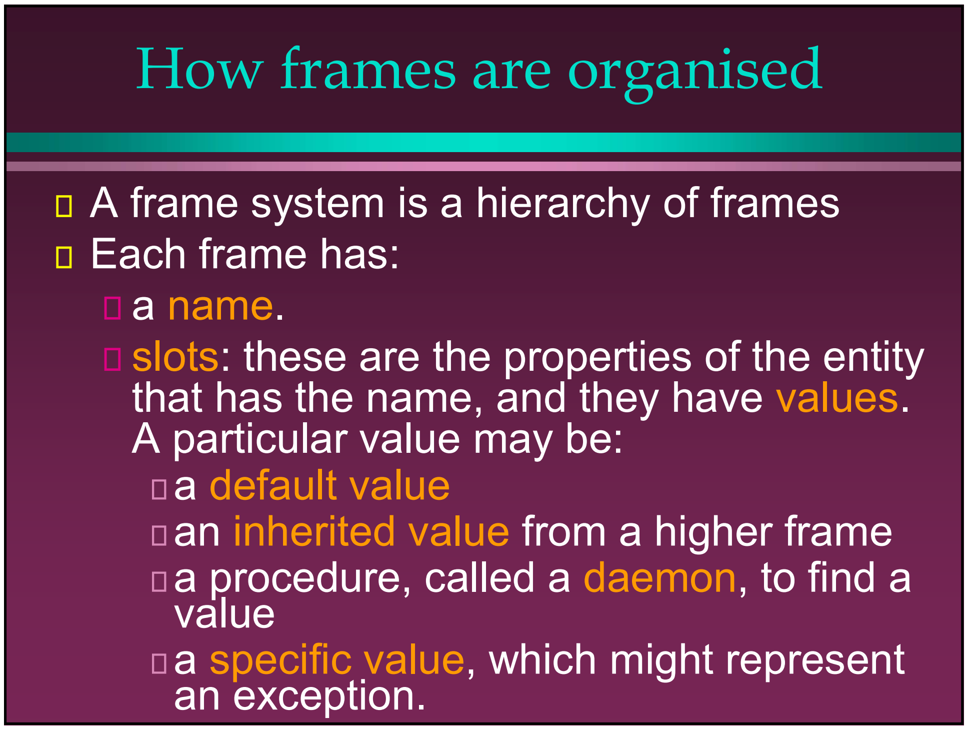
10



Frames

- The idea of frame hierarchies is very similar to the idea of class hierarchies found in object-orientated programming.
- Frame systems are available for off-the-shelf use

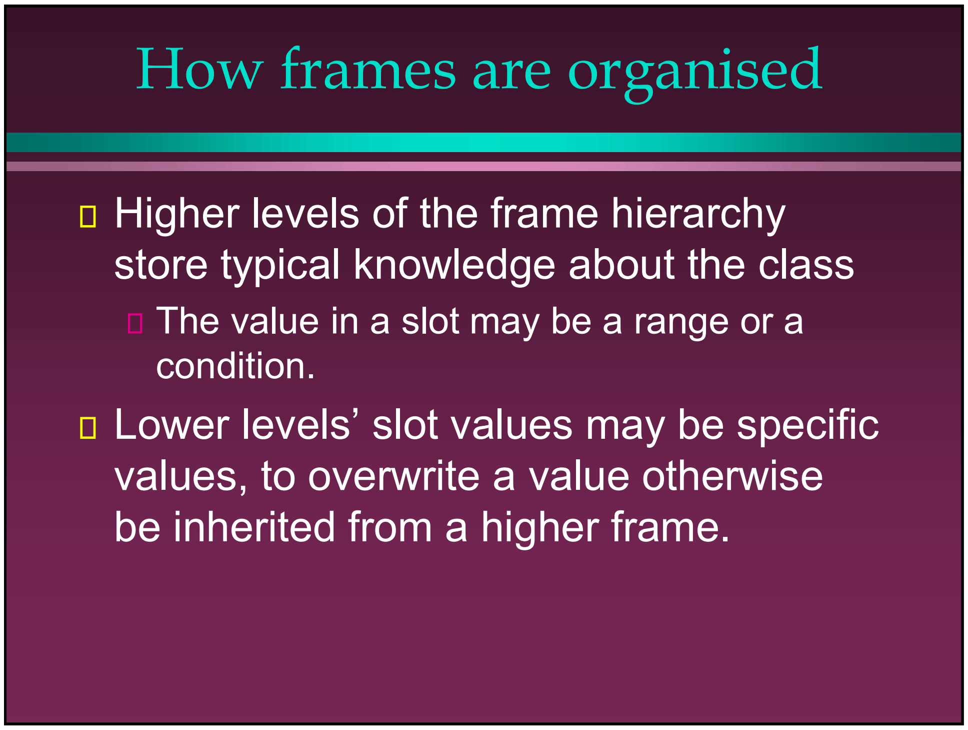
11



How frames are organised

- A frame system is a hierarchy of frames
- Each frame has:
  - a name.
  - slots: these are the properties of the entity that has the name, and they have values. A particular value may be:
    - a default value
    - an inherited value from a higher frame
    - a procedure, called a daemon, to find a value
    - a specific value, which might represent an exception.

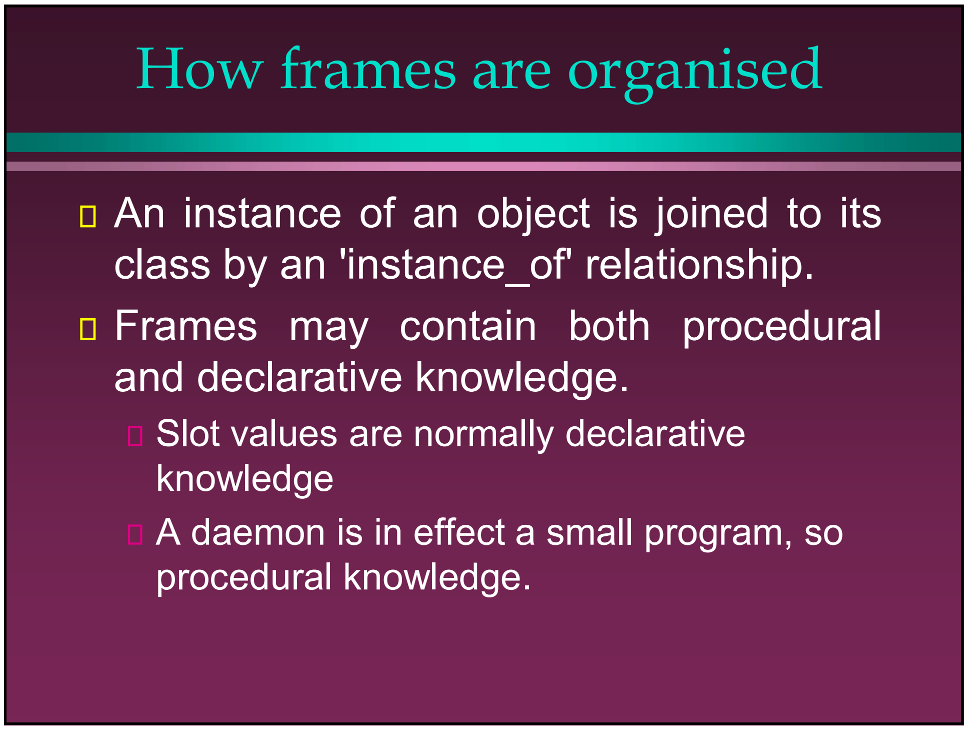
12



How frames are organised

- Higher levels of the frame hierarchy store typical knowledge about the class
  - The value in a slot may be a range or a condition.
- Lower levels' slot values may be specific values, to overwrite a value otherwise be inherited from a higher frame.

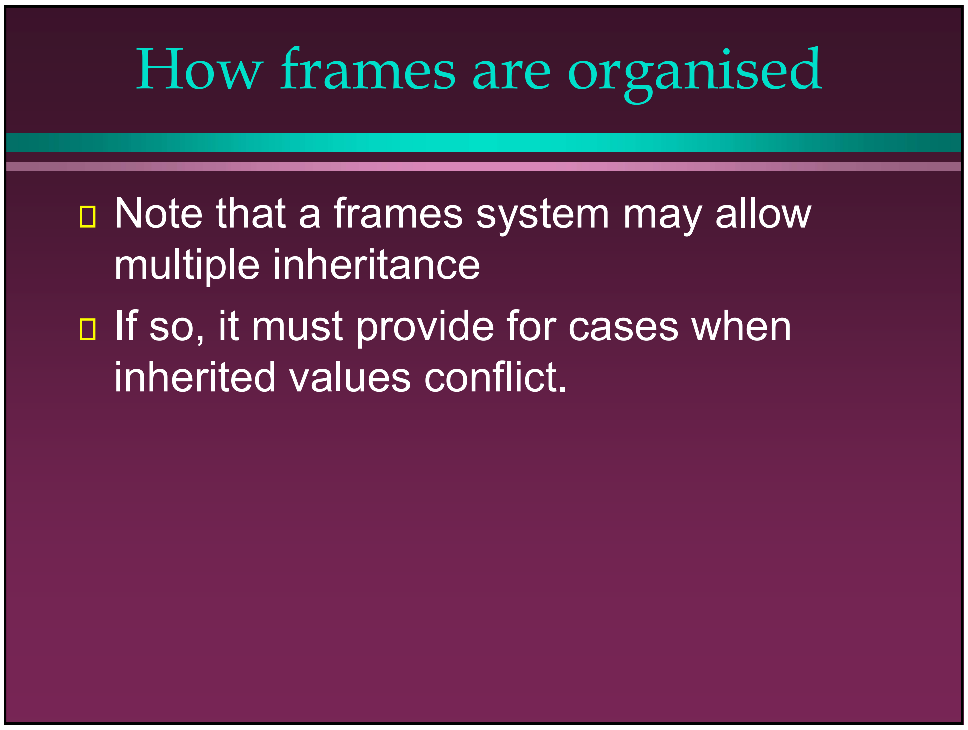
13



How frames are organised

- An instance of an object is joined to its class by an 'instance_of' relationship.
- Frames may contain both procedural and declarative knowledge.
  - Slot values are normally declarative knowledge
  - A daemon is in effect a small program, so procedural knowledge.

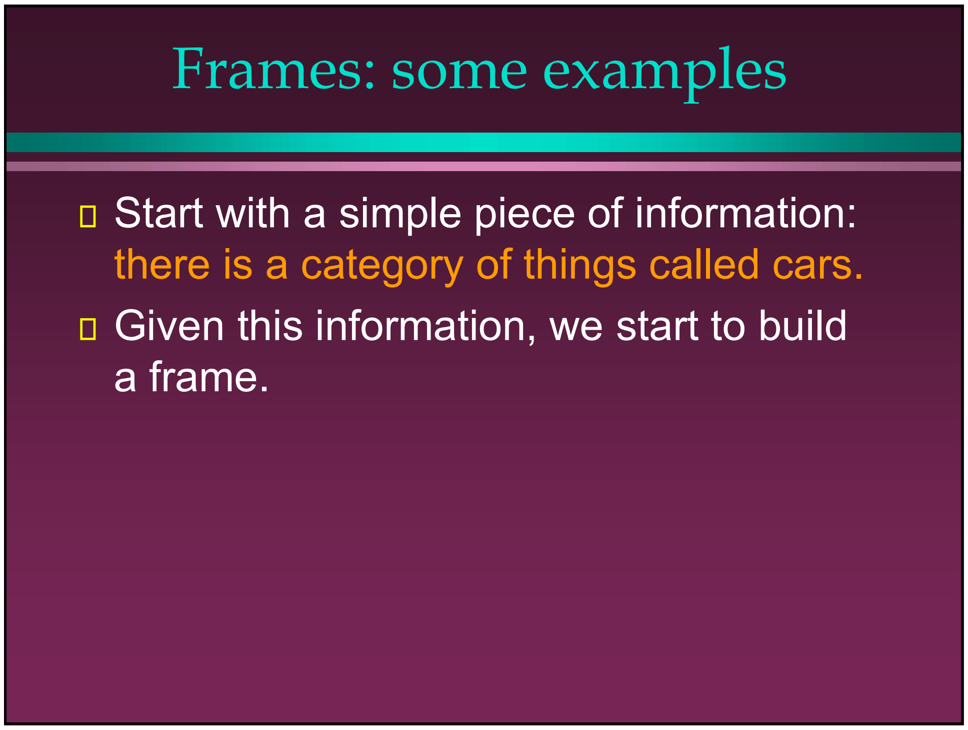
14



How frames are organised

- Note that a frames system may allow multiple inheritance
- If so, it must provide for cases when inherited values conflict.

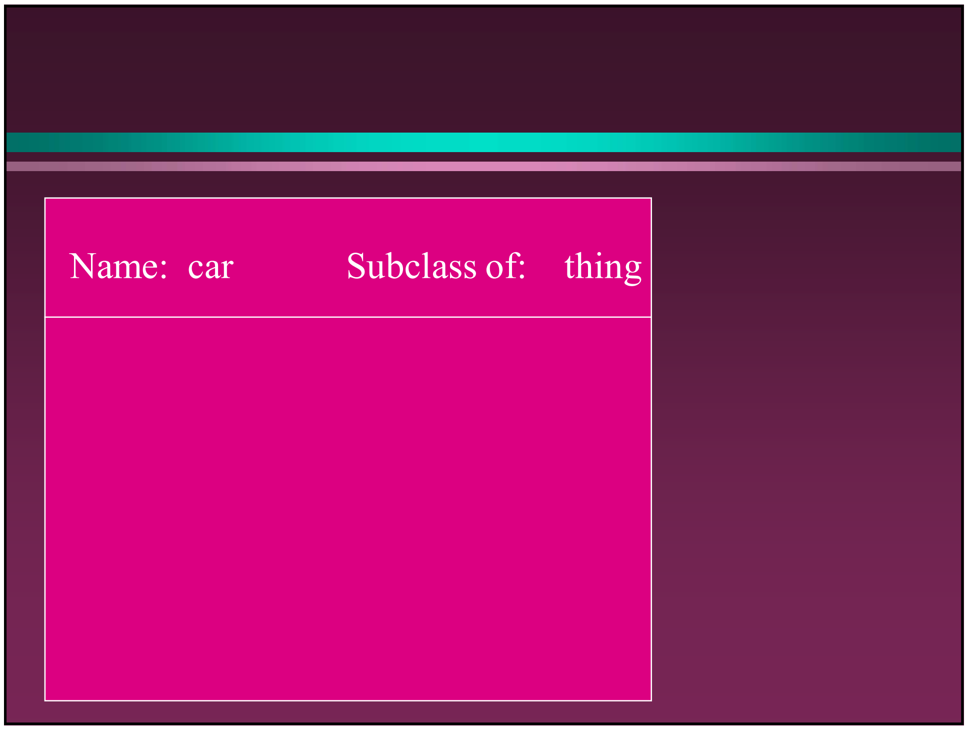
15



Frames: some examples

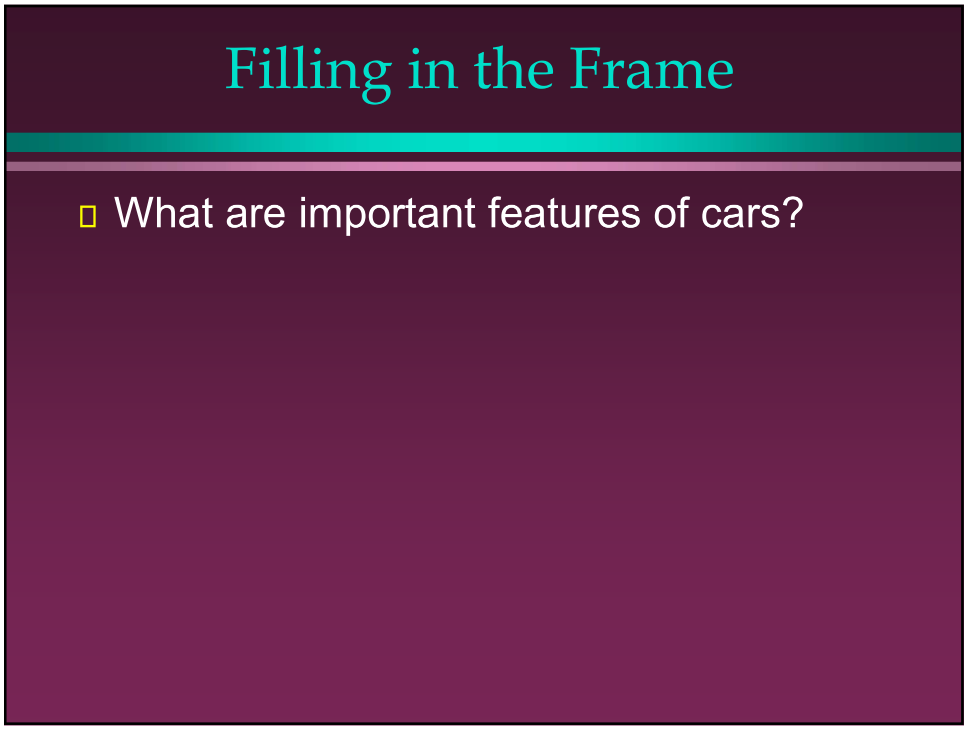
- Start with a simple piece of information: there is a category of things called cars.
- Given this information, we start to build a frame.

18



Name: car	Subclass of: thing

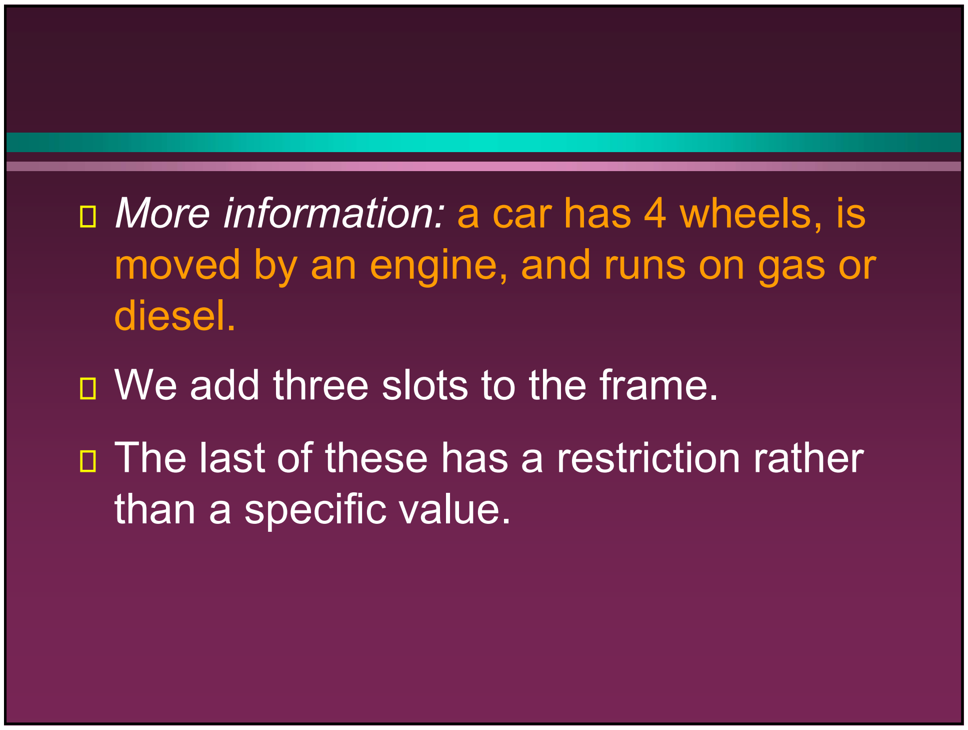
19



Filling in the Frame

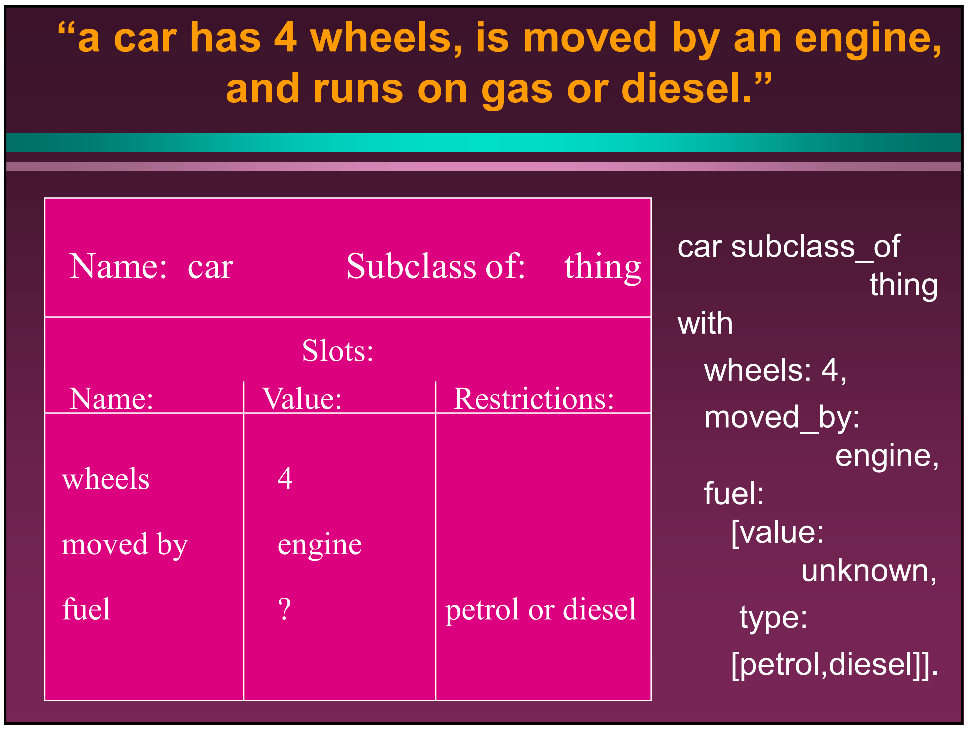
- What are important features of cars?

20



- *More information:* a car has 4 wheels, is moved by an engine, and runs on gas or diesel.
- We add three slots to the frame.
- The last of these has a restriction rather than a specific value.

21

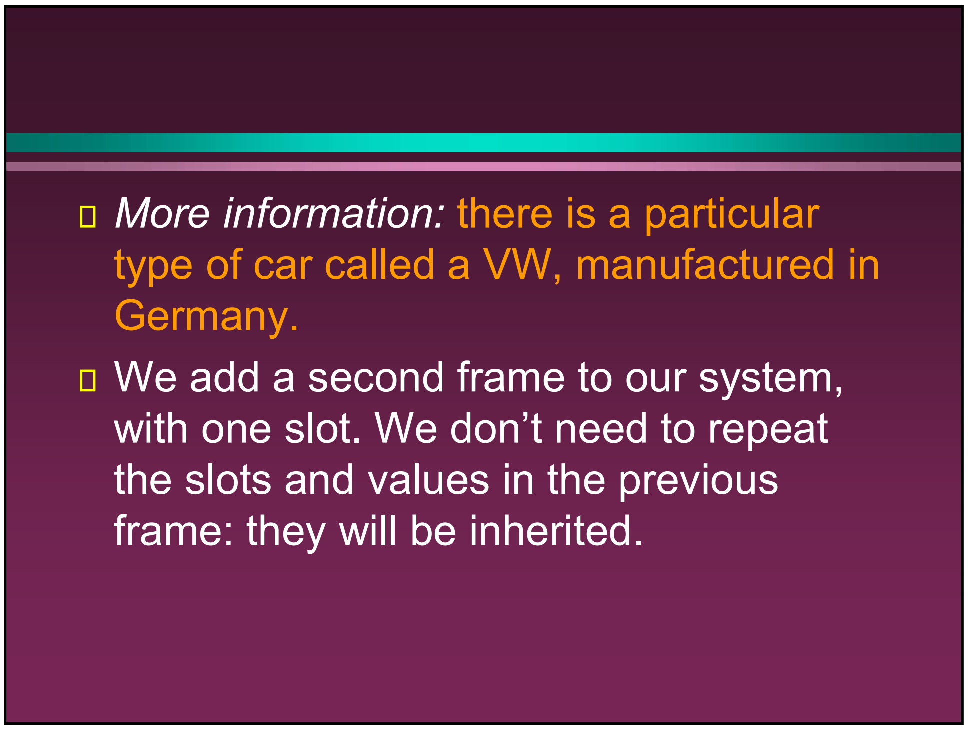


"a car has 4 wheels, is moved by an engine, and runs on gas or diesel."

Name: car	Subclass of: thing	
Slots:		
Name:	Value:	Restrictions:
wheels	4	
moved by	engine	
fuel	?	petrol or diesel

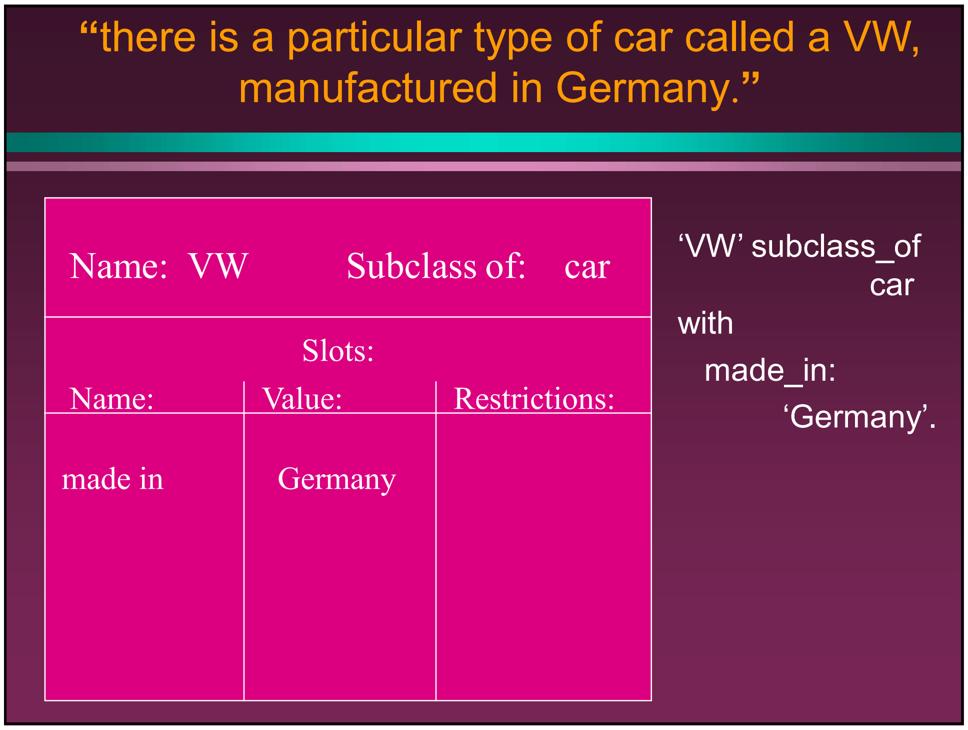
```
car subclass_of
              thing
with
  wheels: 4,
  moved_by:
            engine,
  fuel:
    [value:
          unknown,
     type:
    [petrol,diesel]].
```

22



- *More information:* there is a particular type of car called a VW, manufactured in Germany.
- We add a second frame to our system, with one slot. We don't need to repeat the slots and values in the previous frame: they will be inherited.

23

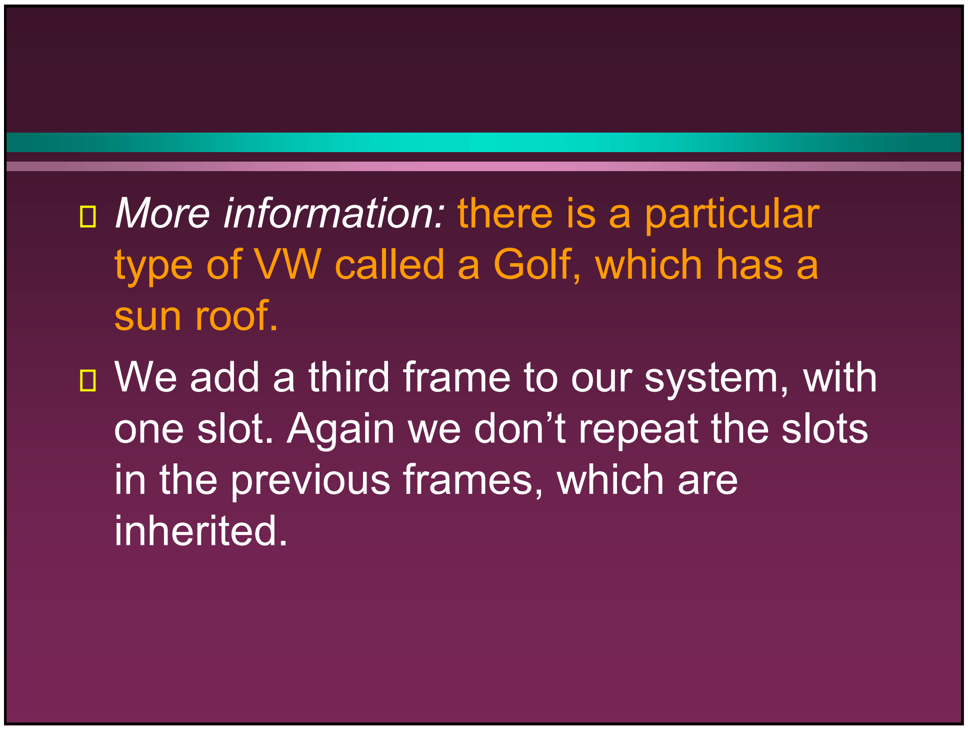


"there is a particular type of car called a VW, manufactured in Germany."

Name: VW	Subclass of: car	
Slots:		
Name:	Value:	Restrictions:
made in	Germany	

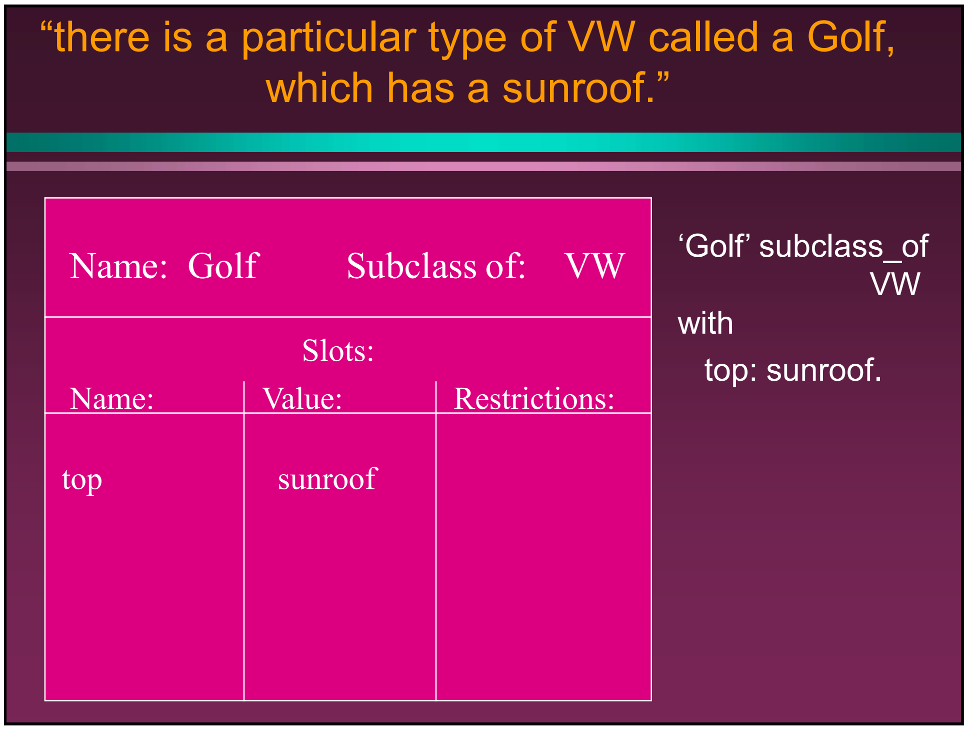
'VW' subclass_of car
with
made_in: 'Germany'.

24



- *More information:* there is a particular type of VW called a Golf, which has a sun roof.
- We add a third frame to our system, with one slot. Again we don't repeat the slots in the previous frames, which are inherited.

25

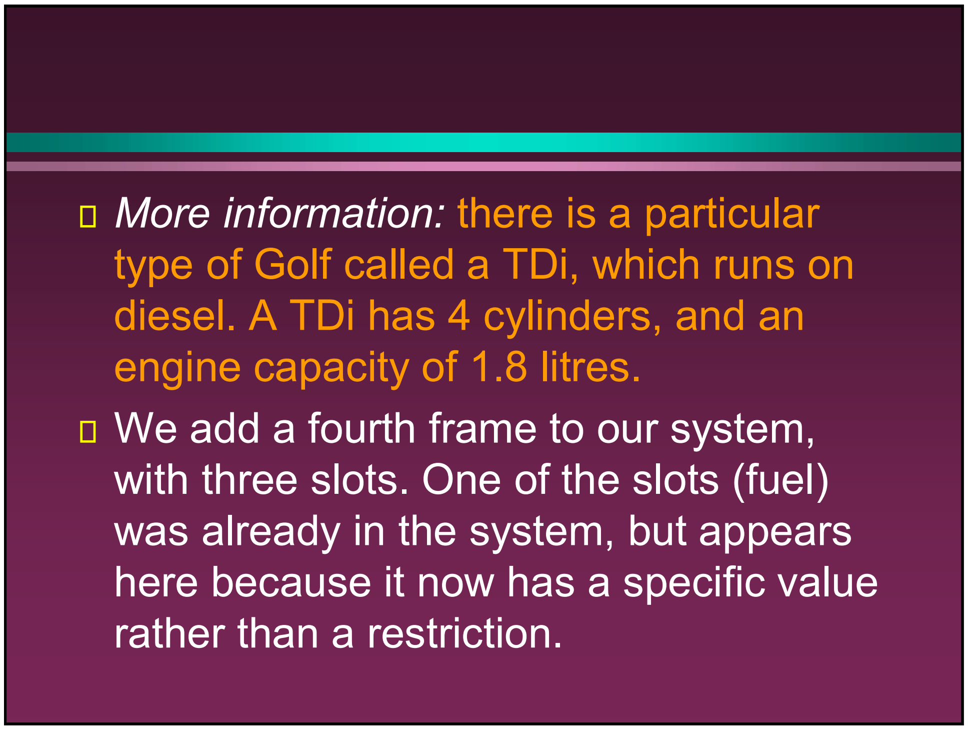


"there is a particular type of VW called a Golf, which has a sunroof."

Name: Golf	Subclass of: VW	
Slots:		
Name:	Value:	Restrictions:
top	sunroof	

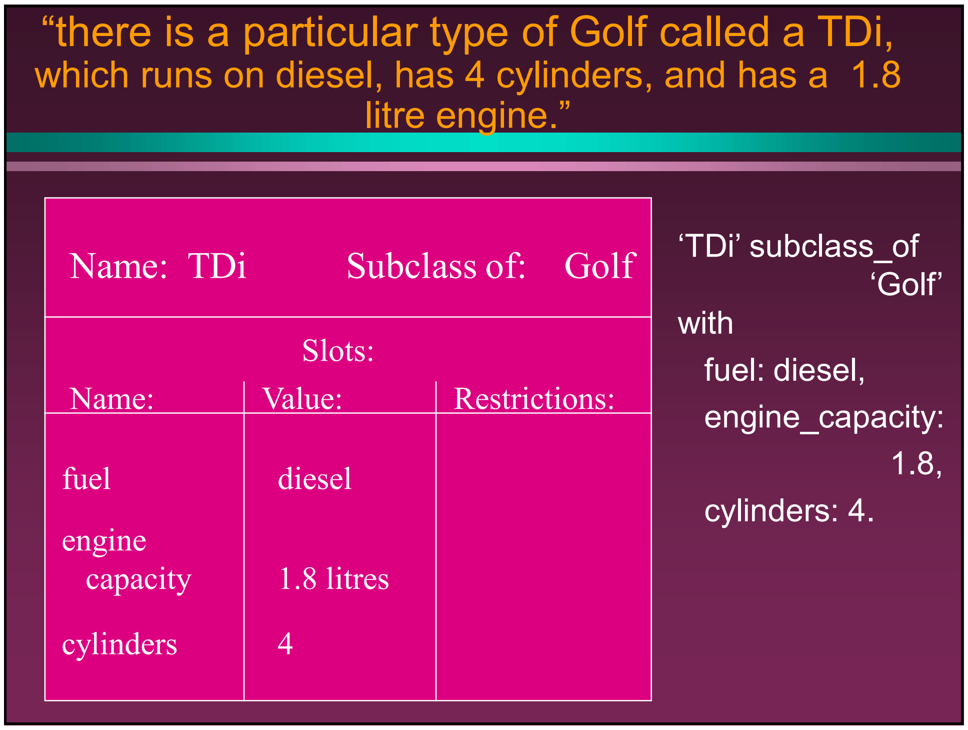
'Golf' subclass_of VW
with
top: sunroof.

26



- *More information:* there is a particular type of Golf called a TDi, which runs on diesel. A TDi has 4 cylinders, and an engine capacity of 1.8 litres.
- We add a fourth frame to our system, with three slots. One of the slots (fuel) was already in the system, but appears here because it now has a specific value rather than a restriction.

27



"there is a particular type of Golf called a TDi, which runs on diesel, has 4 cylinders, and has a 1.8 litre engine."

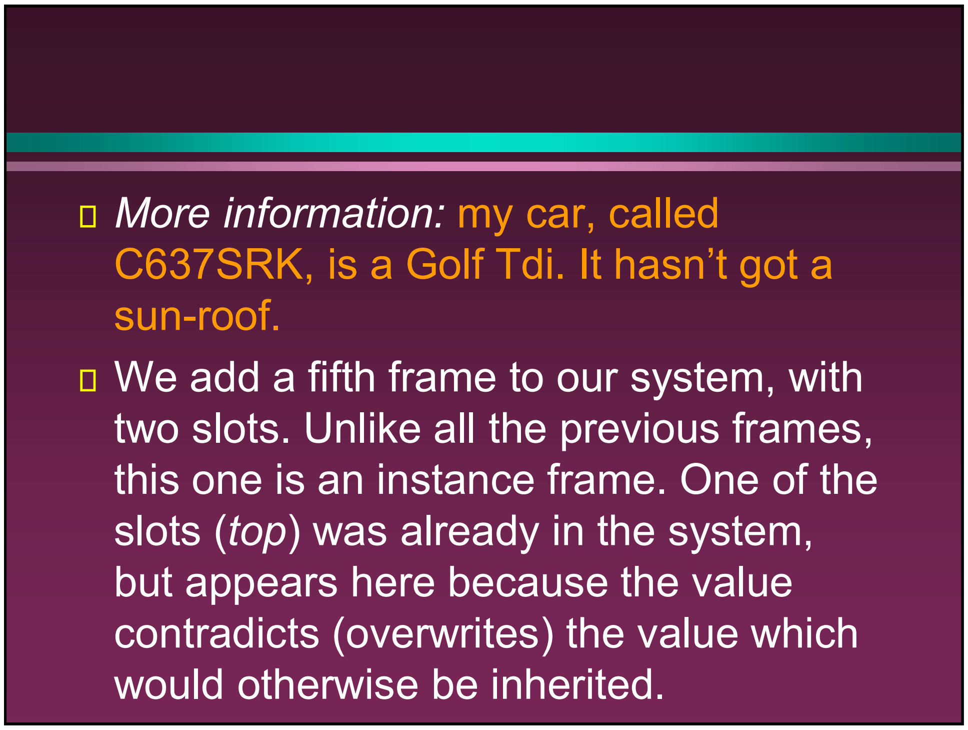
Name: TDi Subclass of: Golf

Slots:

Name:	Value:	Restrictions:
fuel	diesel	
engine capacity	1.8 litres	
cylinders	4	

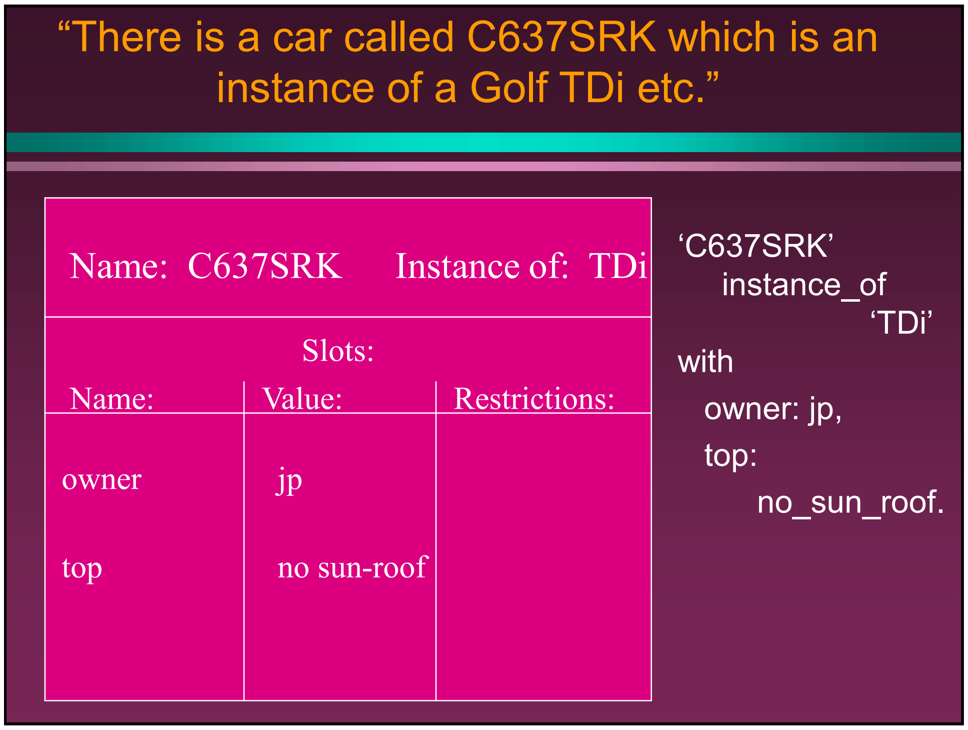
'TDi' subclass_of 'Golf'
with
fuel: diesel,
engine_capacity: 1.8,
cylinders: 4.

28



- *More information:* my car, called C637SRK, is a Golf Tdi. It hasn't got a sun-roof.
- We add a fifth frame to our system, with two slots. Unlike all the previous frames, this one is an instance frame. One of the slots (*top*) was already in the system, but appears here because the value contradicts (overwrites) the value which would otherwise be inherited.

29



"There is a car called C637SRK which is an instance of a Golf TDi etc."

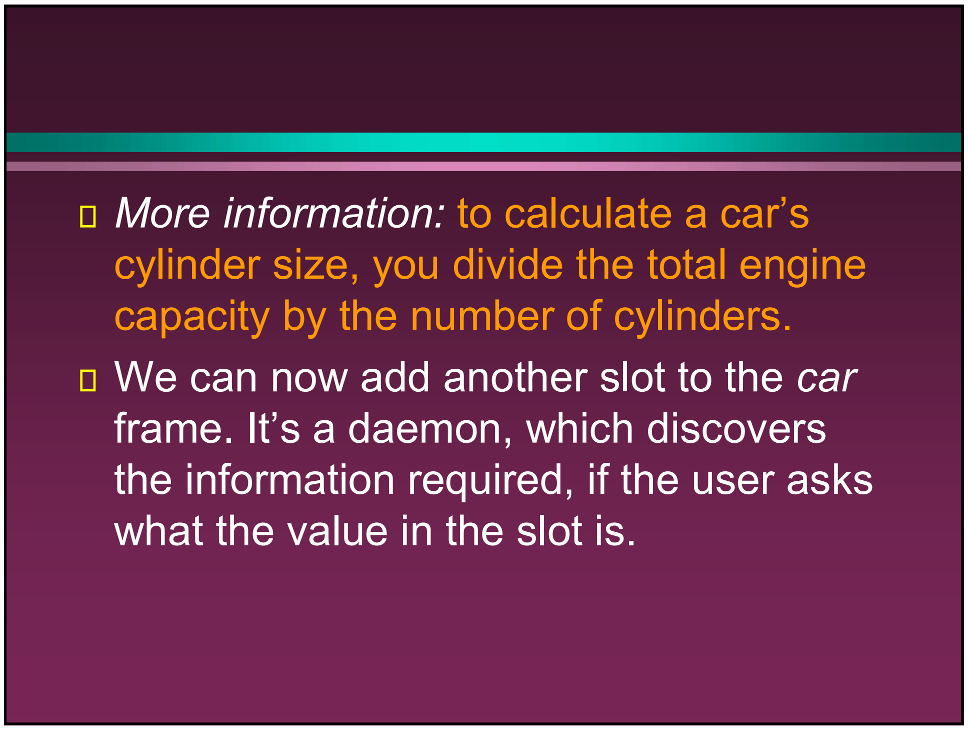
Name: C637SRK Instance of: TDi

Slots:

Name:	Value:	Restrictions:
owner	jp	
top	no sun-roof	

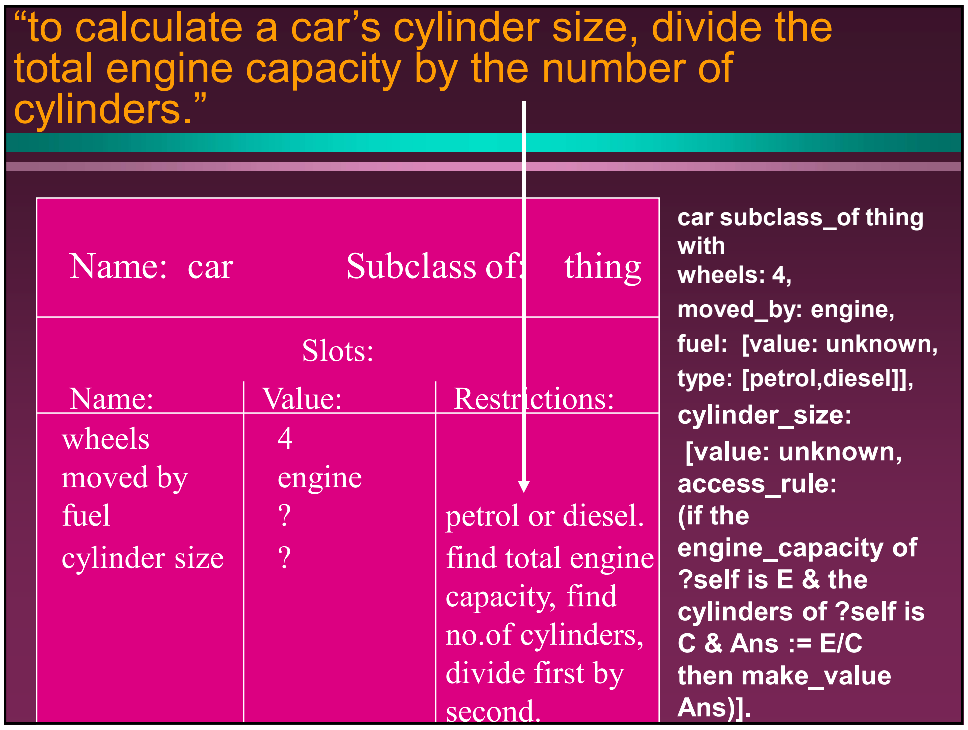
```
'C637SRK'
    instance_of
              'TDi'
with
  owner: jp,
  top:
      no_sun_roof.
```

30



- *More information:* to calculate a car's cylinder size, you divide the total engine capacity by the number of cylinders.
- We can now add another slot to the *car* frame. It's a daemon, which discovers the information required, if the user asks what the value in the slot is.

31



"to calculate a car's cylinder size, divide the total engine capacity by the number of cylinders."

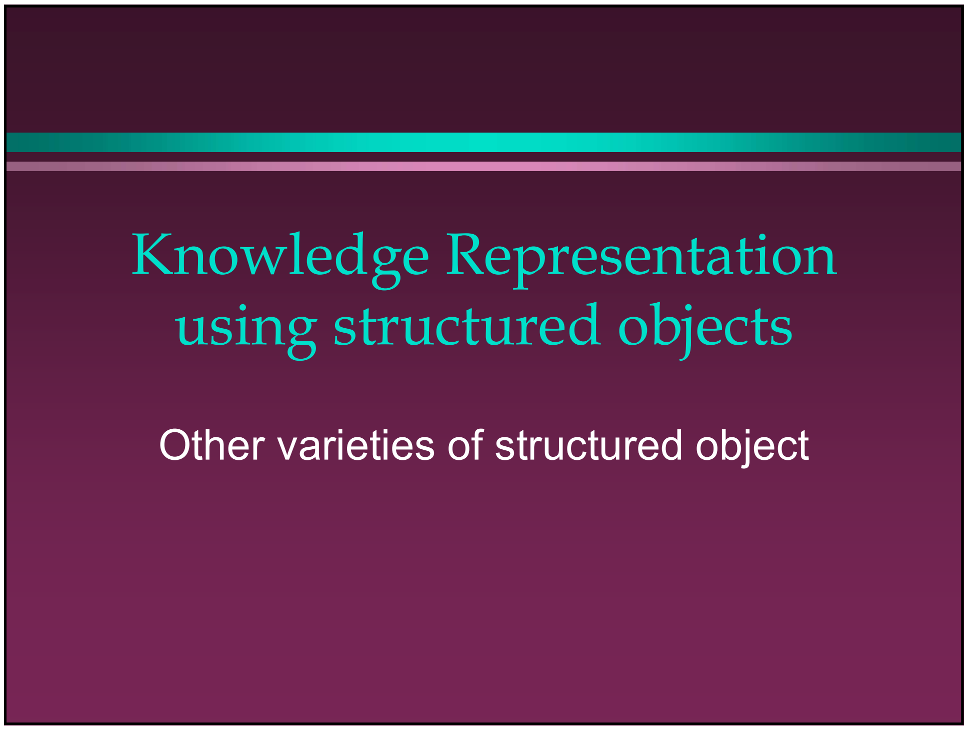
Name: car	Subclass of: thing	
Slots:		
Name:	Value:	Restrictions:
wheels	4	
moved by	engine	
fuel	?	petrol or diesel.
cylinder size	?	find total engine capacity, find no.of cylinders, divide first by second.

```
car subclass_of thing
with
wheels: 4,
moved_by: engine,
fuel:  [value: unknown,
type: [petrol,diesel]],
cylinder_size:
 [value: unknown,
access_rule:
(if the
engine_capacity of
?self is E & the
cylinders of ?self is
C & Ans := E/C
then make_value
Ans)].
```

32

33

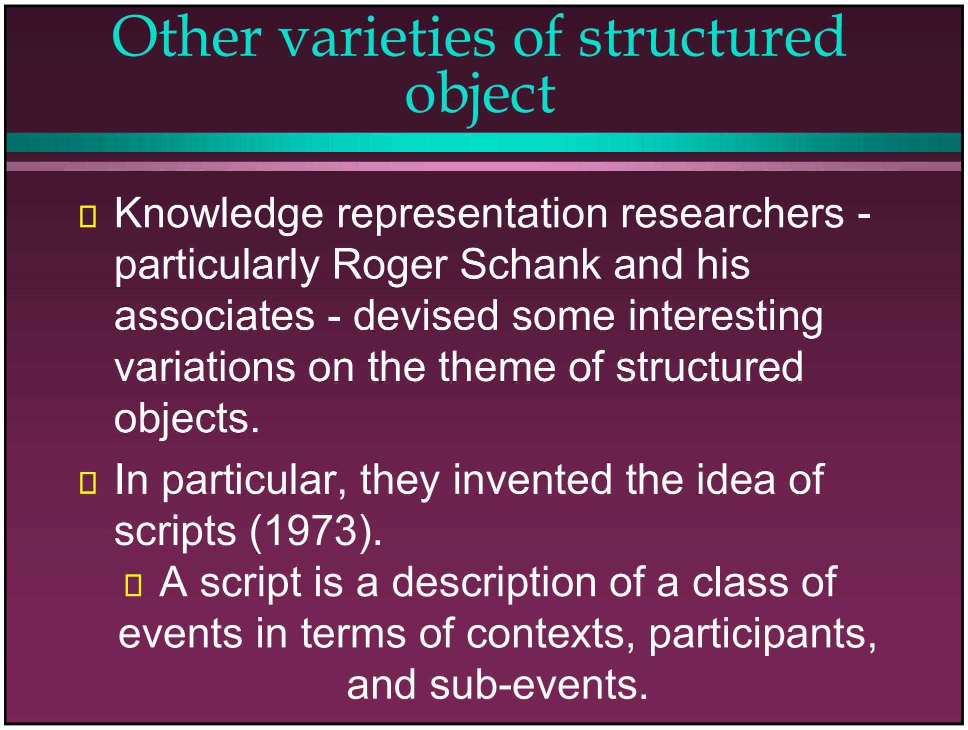
34



Knowledge Representation using structured objects

Other varieties of structured object

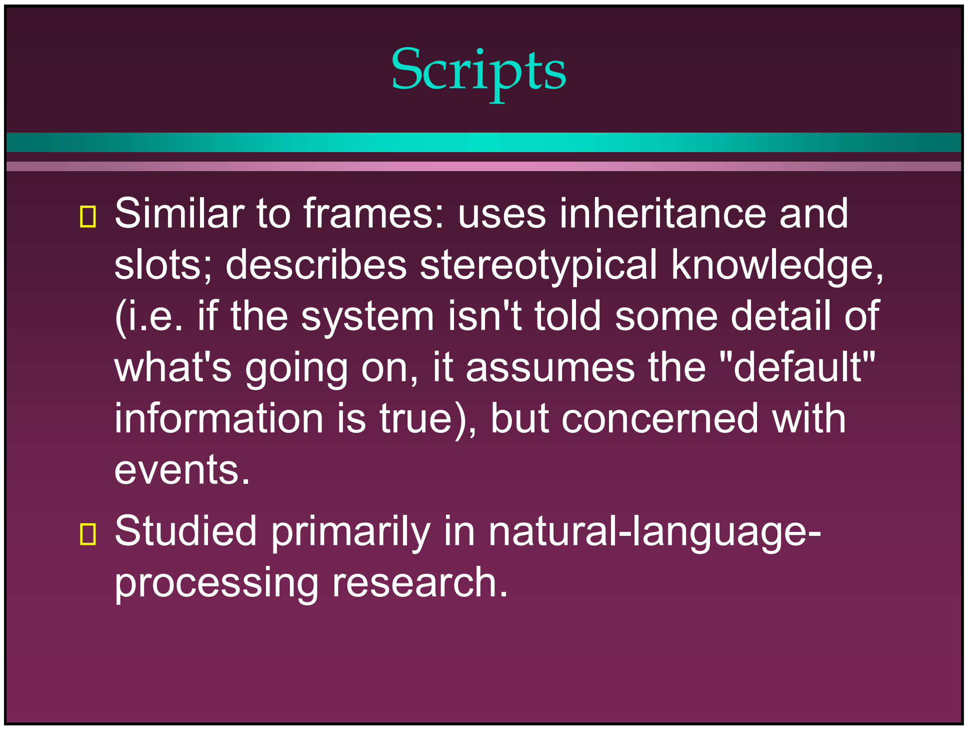
35



Other varieties of structured object

- Knowledge representation researchers - particularly Roger Schank and his associates - devised some interesting variations on the theme of structured objects.
- In particular, they invented the idea of scripts (1973).
  - A script is a description of a class of events in terms of contexts, participants, and sub-events.

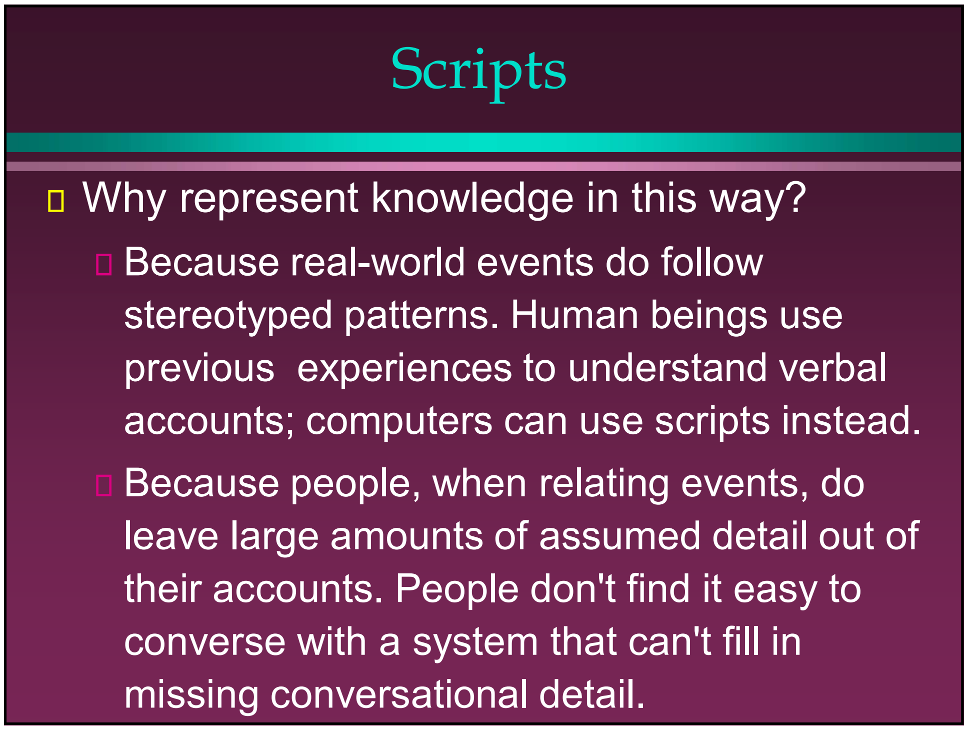
36



Scripts

- Similar to frames: uses inheritance and slots; describes stereotypical knowledge, (i.e. if the system isn't told some detail of what's going on, it assumes the "default" information is true), but concerned with events.
- Studied primarily in natural-language-processing research.

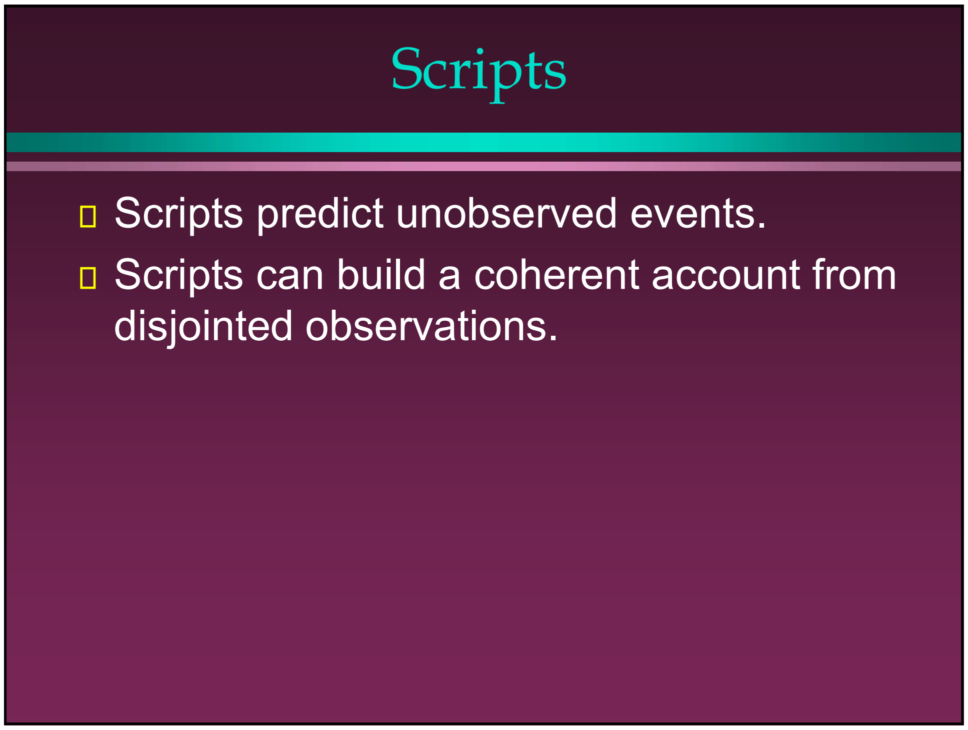
37



Scripts

- Why represent knowledge in this way?
  - Because real-world events do follow stereotyped patterns. Human beings use previous experiences to understand verbal accounts; computers can use scripts instead.
  - Because people, when relating events, do leave large amounts of assumed detail out of their accounts. People don't find it easy to converse with a system that can't fill in missing conversational detail.

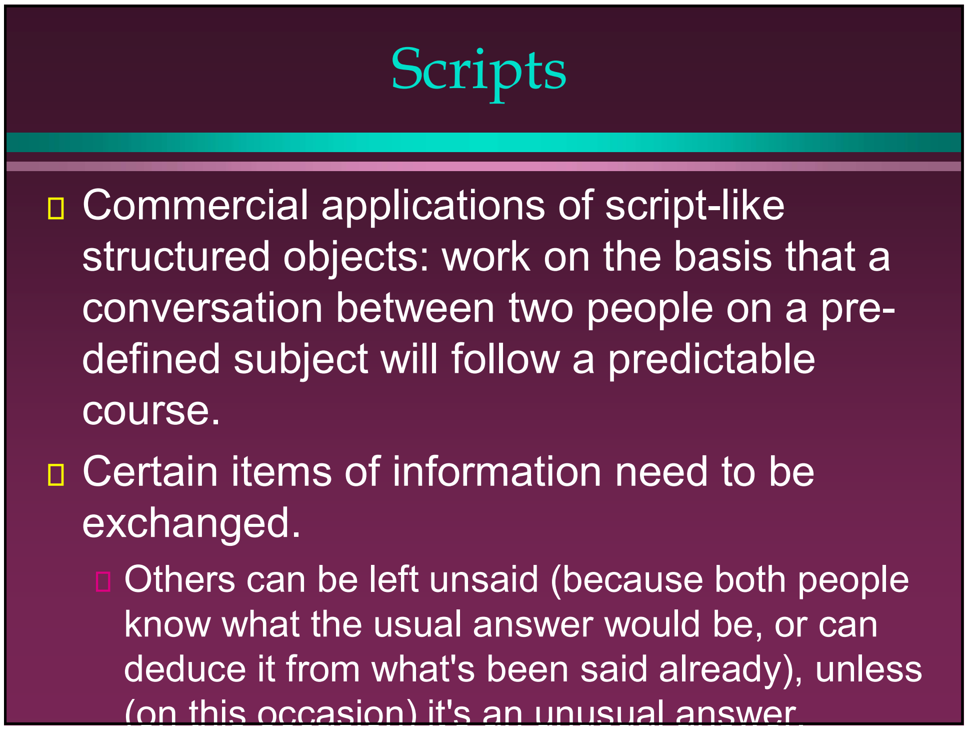
38



Scripts

- Scripts predict unobserved events.
- Scripts can build a coherent account from disjointed observations.

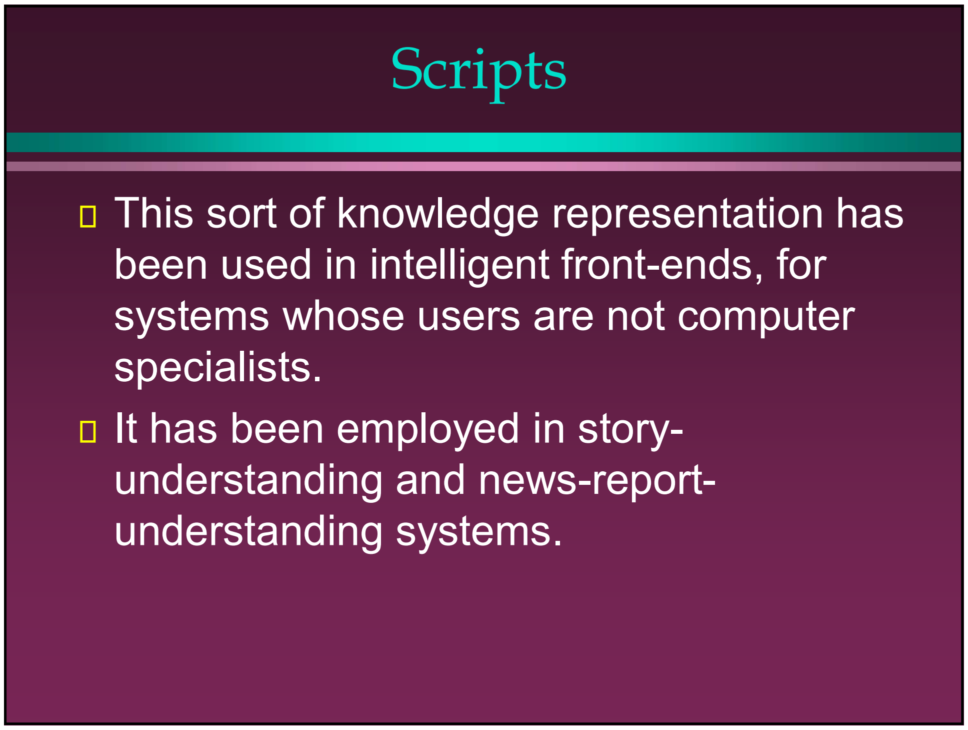
39



Scripts

- Commercial applications of script-like structured objects: work on the basis that a conversation between two people on a pre-defined subject will follow a predictable course.
- Certain items of information need to be exchanged.
  - Others can be left unsaid (because both people know what the usual answer would be, or can deduce it from what's been said already), unless (on this occasion) it's an unusual answer.

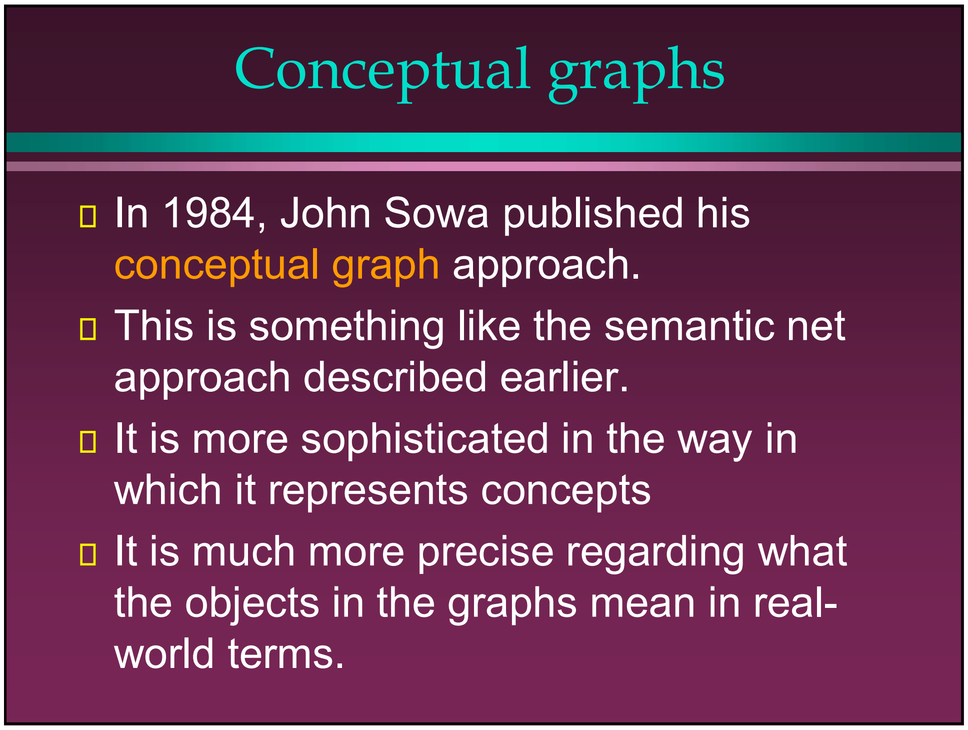
40



Scripts

- This sort of knowledge representation has been used in intelligent front-ends, for systems whose users are not computer specialists.
- It has been employed in story-understanding and news-report-understanding systems.

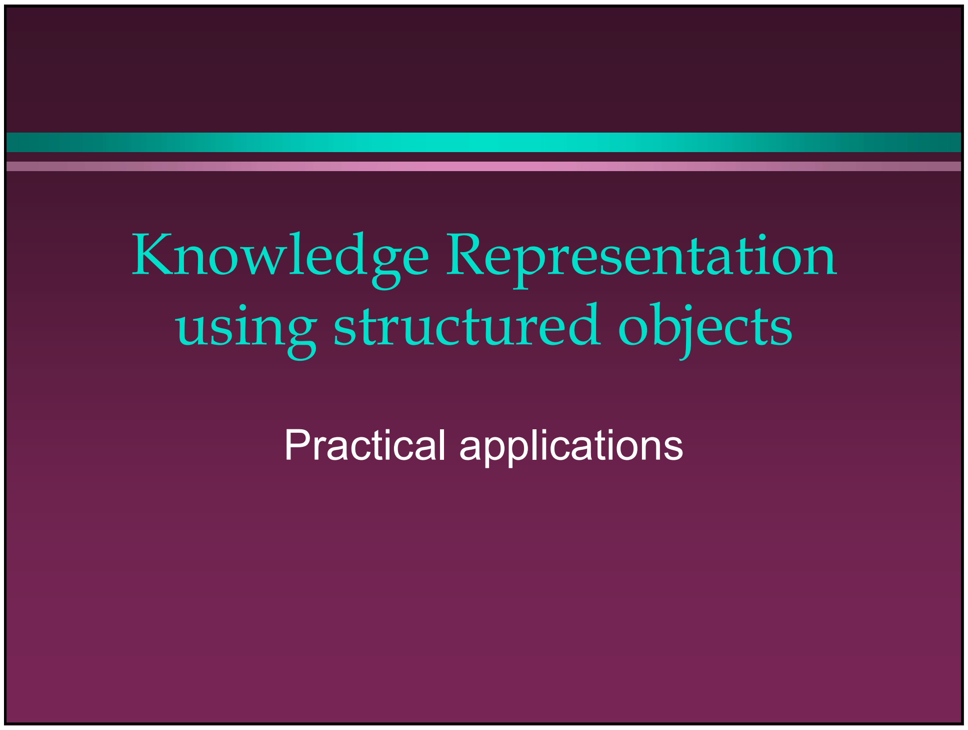
41



Conceptual graphs

- In 1984, John Sowa published his conceptual graph approach.
- This is something like the semantic net approach described earlier.
- It is more sophisticated in the way in which it represents concepts
- It is much more precise regarding what the objects in the graphs mean in real-world terms.

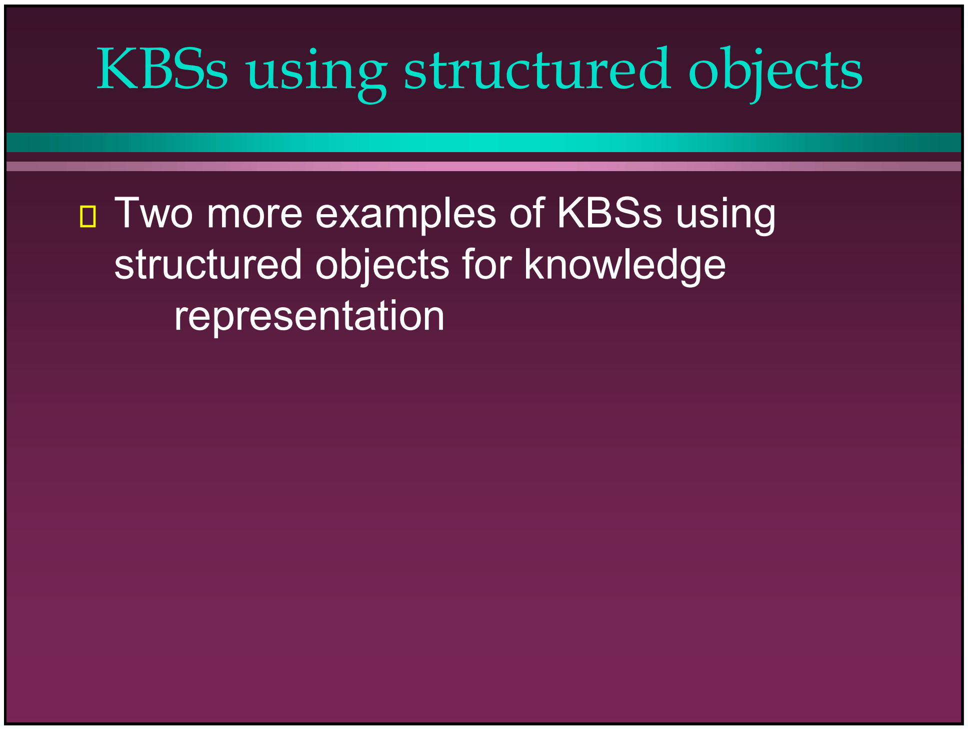
42



Knowledge Representation using structured objects

Practical applications

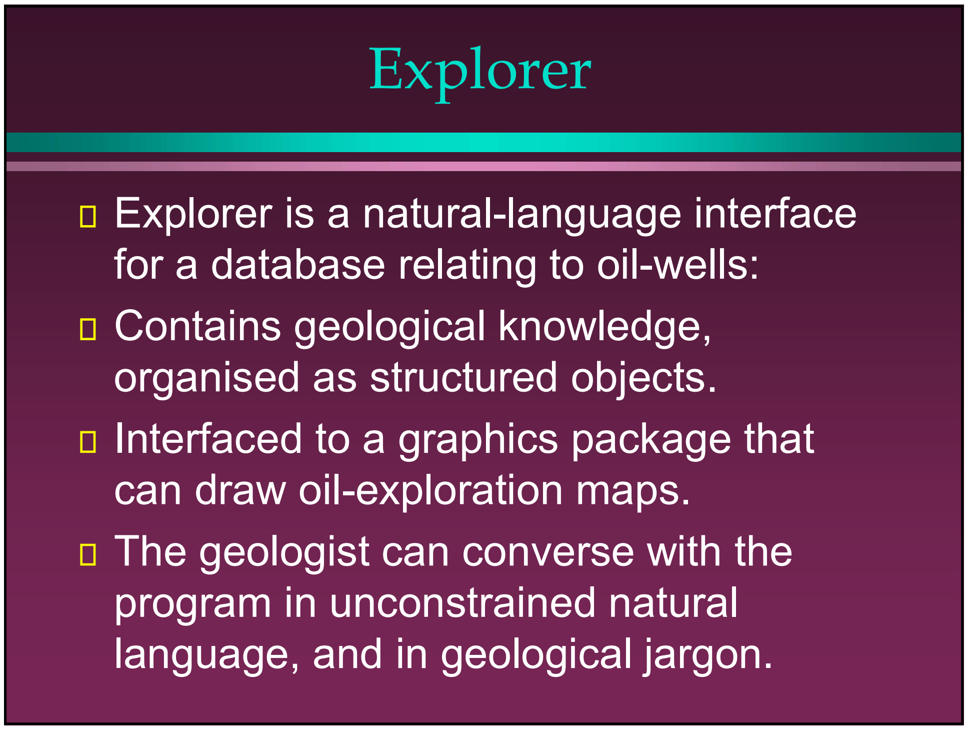
43



KBSs using structured objects

- Two more examples of KBSs using structured objects for knowledge representation

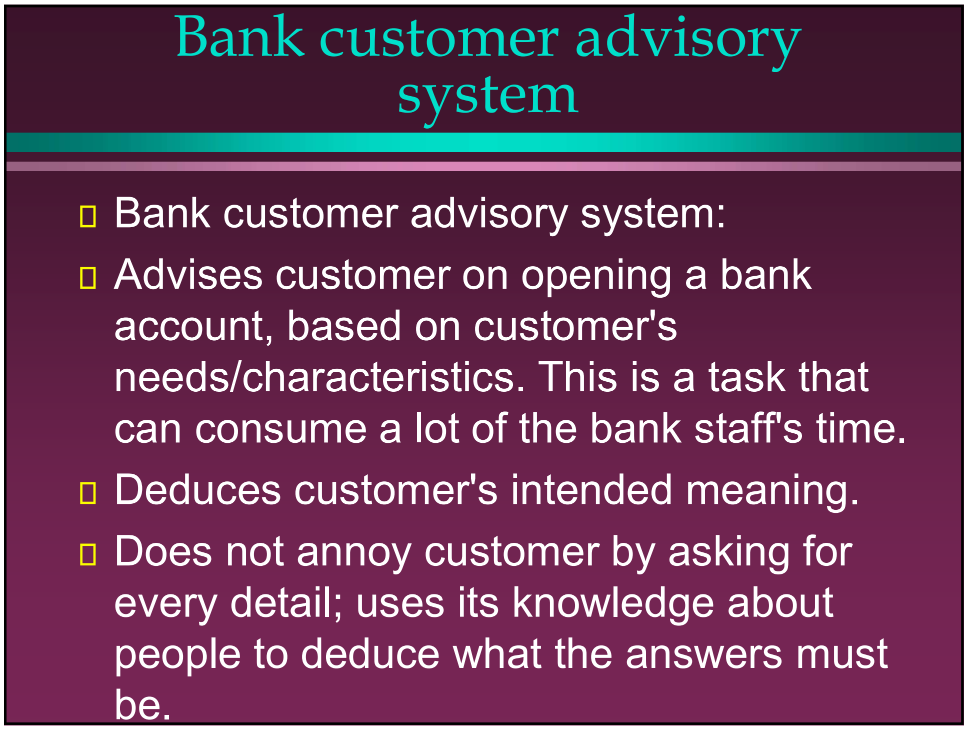
44



Explorer

- Explorer is a natural-language interface for a database relating to oil-wells:
- Contains geological knowledge, organised as structured objects.
- Interfaced to a graphics package that can draw oil-exploration maps.
- The geologist can converse with the program in unconstrained natural language, and in geological jargon.

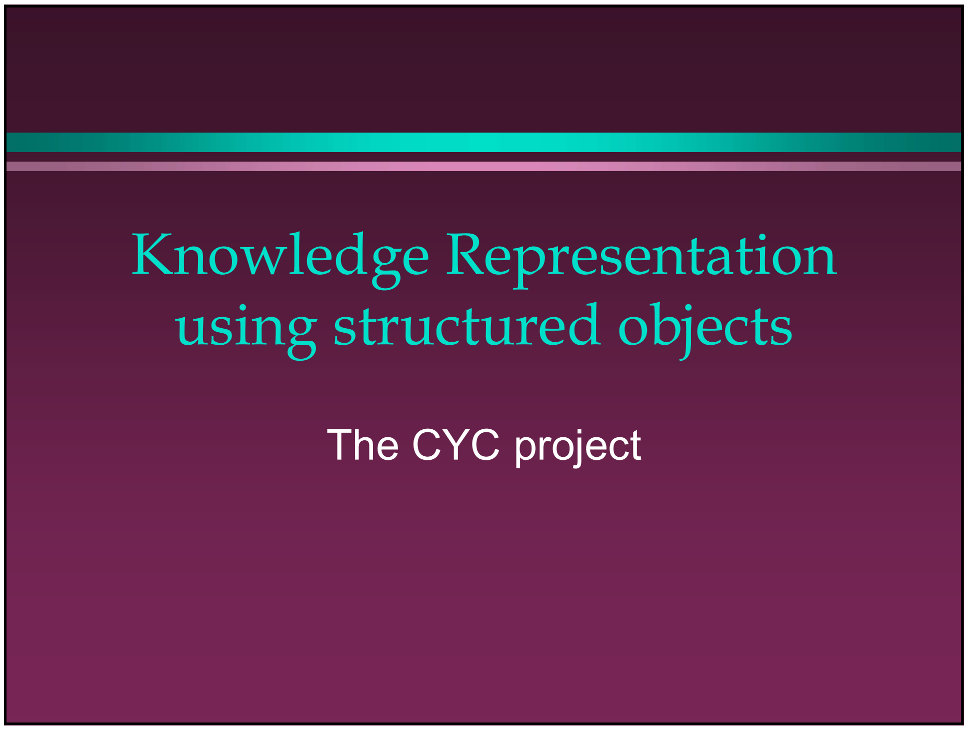
45



Bank customer advisory system

- Bank customer advisory system:
- Advises customer on opening a bank account, based on customer's needs/characteristics. This is a task that can consume a lot of the bank staff's time.
- Deduces customer's intended meaning.
- Does not annoy customer by asking for every detail; uses its knowledge about people to deduce what the answers must be.

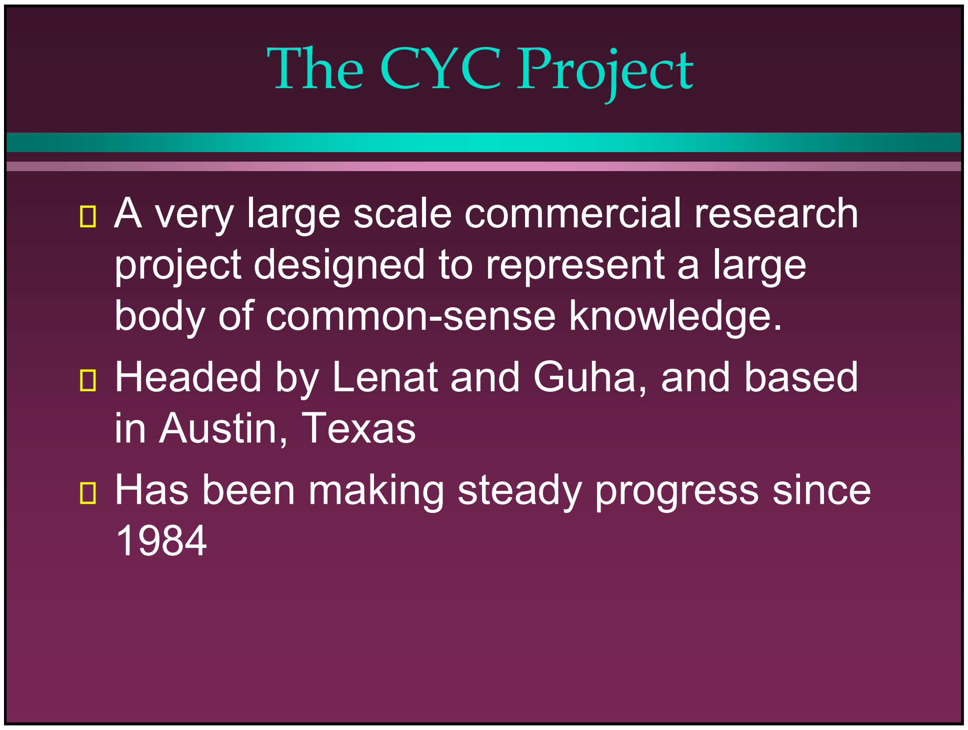
46



Knowledge Representation using structured objects

The CYC project

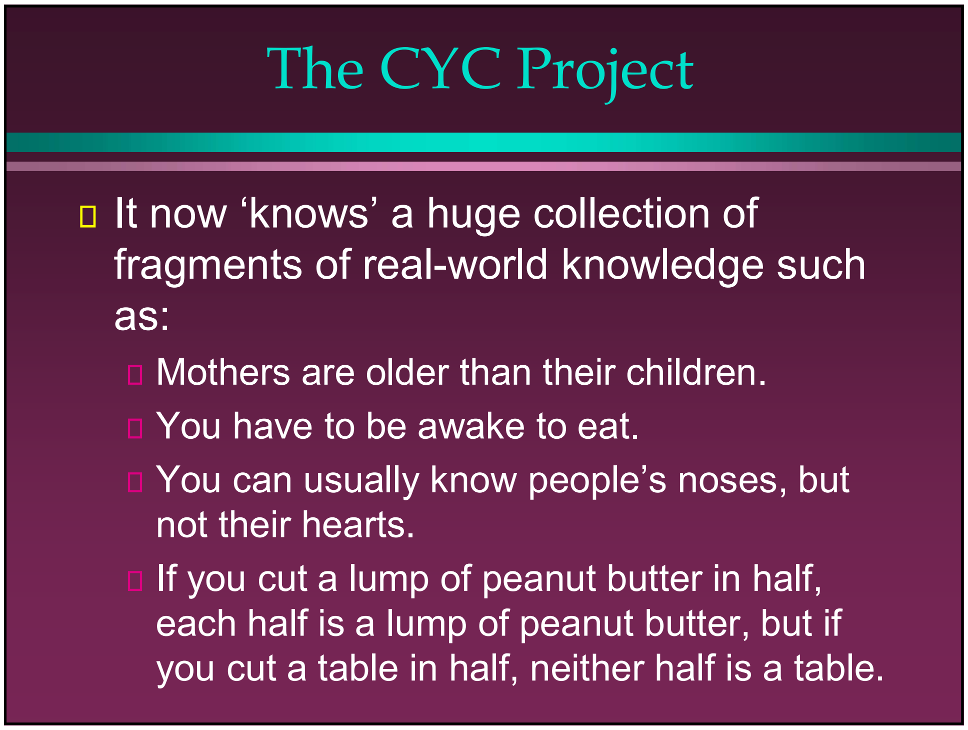
47



The CYC Project

- A very large scale commercial research project designed to represent a large body of common-sense knowledge.
- Headed by Lenat and Guha, and based in Austin, Texas
- Has been making steady progress since 1984

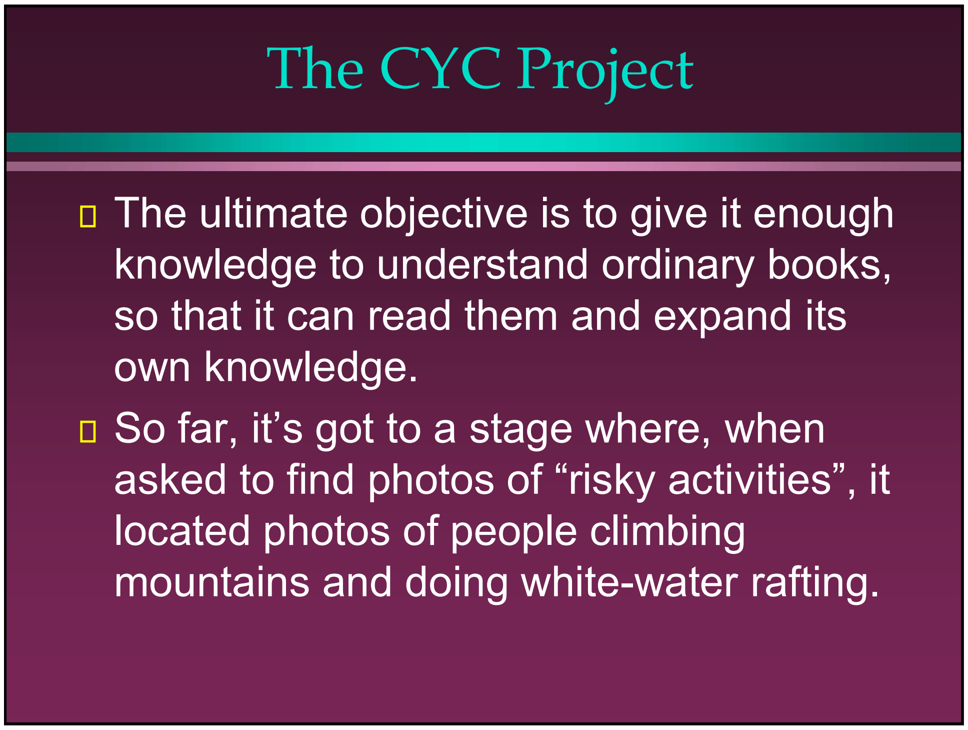
48



The CYC Project

- It now ‘knows’ a huge collection of fragments of real-world knowledge such as:
  - Mothers are older than their children.
  - You have to be awake to eat.
  - You can usually know people’s noses, but not their hearts.
  - If you cut a lump of peanut butter in half, each half is a lump of peanut butter, but if you cut a table in half, neither half is a table.

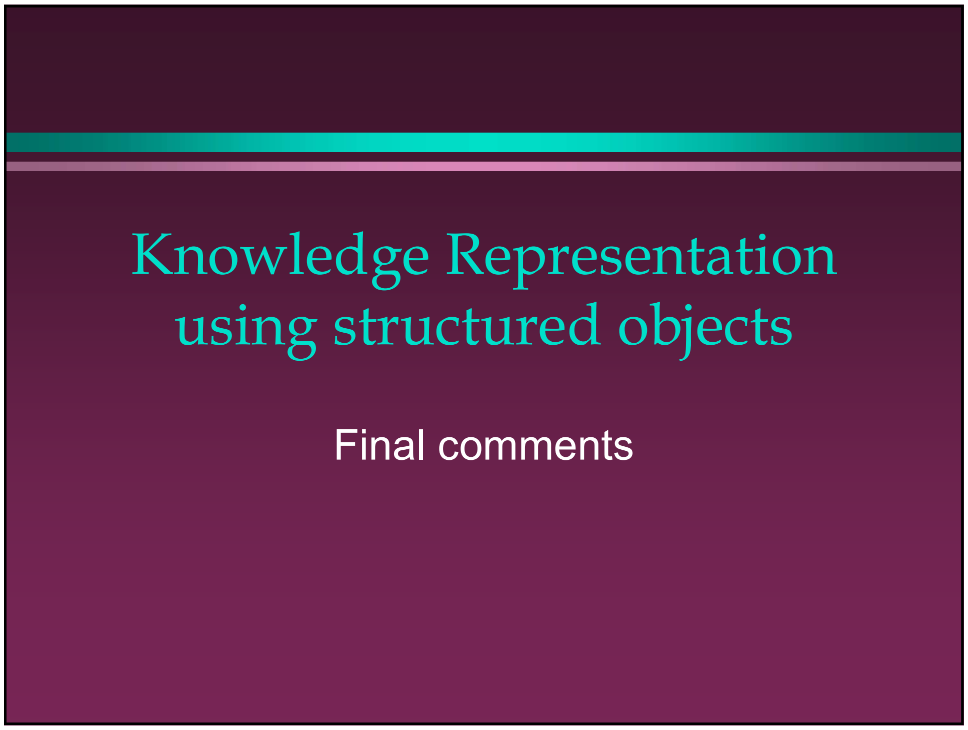
49



The CYC Project

- The ultimate objective is to give it enough knowledge to understand ordinary books, so that it can read them and expand its own knowledge.
- So far, it’s got to a stage where, when asked to find photos of “risky activities”, it located photos of people climbing mountains and doing white-water rafting.

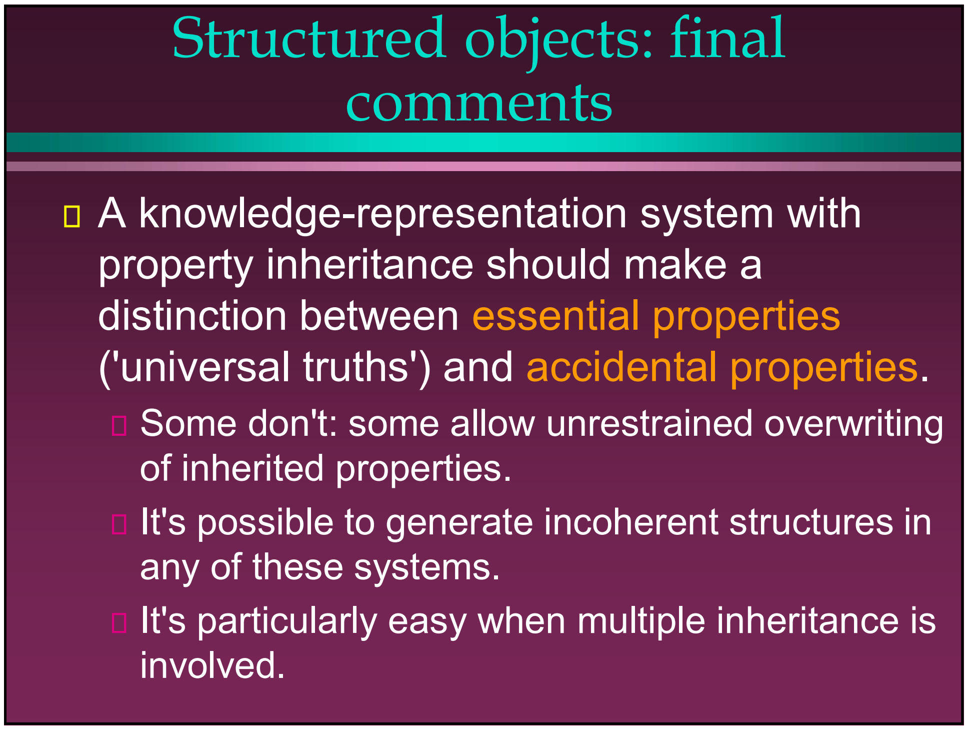
50



Knowledge Representation using structured objects

Final comments

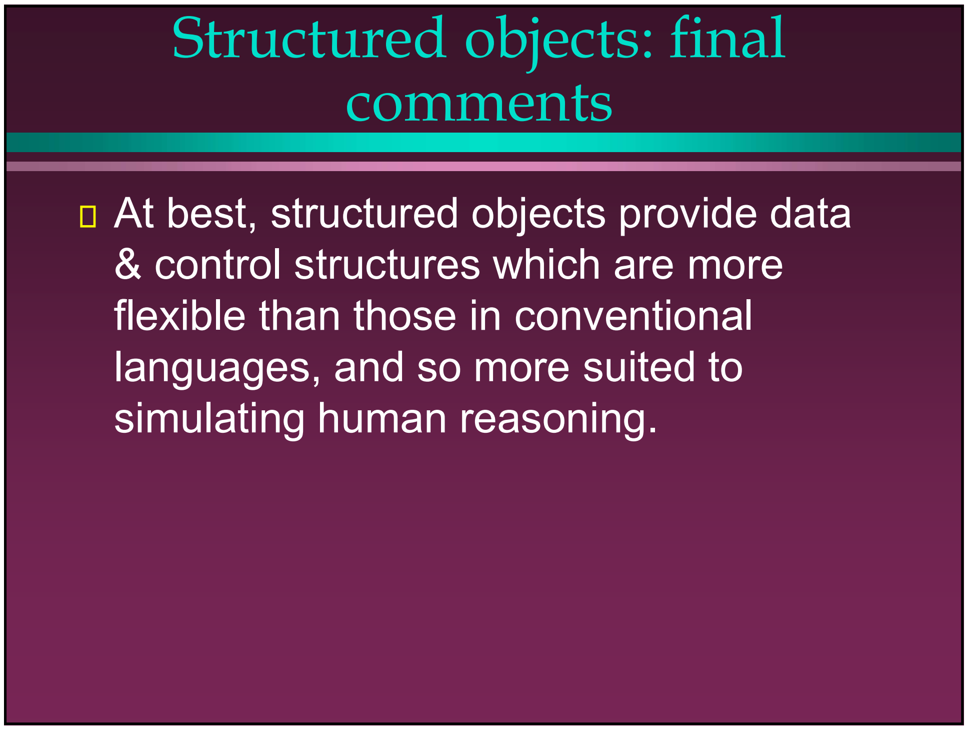
51



Structured objects: final comments

- A knowledge-representation system with property inheritance should make a distinction between essential properties ('universal truths') and accidental properties.
  - Some don't: some allow unrestrained overwriting of inherited properties.
  - It's possible to generate incoherent structures in any of these systems.
  - It's particularly easy when multiple inheritance is involved.

52



Structured objects: final comments

- At best, structured objects provide data & control structures which are more flexible than those in conventional languages, and so more suited to simulating human reasoning.

53