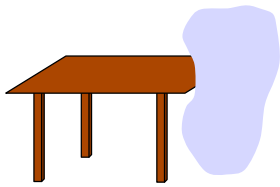
Logics of Image Interpretation

1

Describing Image Interpretation in Logical Terms

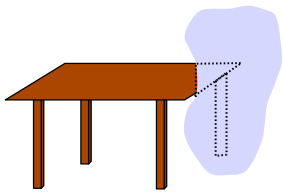
In 2D images (with possible occlusions) we never see the complete 3D reality.



?

deduction

"from the evidence I conclude that this is a table"



!

model construction

"my conceptual model of a table + hypothesised scene components explain the evidence"

Reiter & Mackworth 87, Matsuyama 90, Schröder 99

Animated slide!

2

Definition of Model Construction

An interpretation I = [D, φ, π] of a logical language maps
- **constant symbols of the language into individuals of a real-world domain D**
- **N-ary predicate symbols of the language into predicate functions over D^N**

A model of some clauses is an interpretation for which all clauses are true.

How to do model construction:

- **Establish mapping φ by assigning segmentation results to constant symbols**
- **Establish mapping π by assigning computational procedures to predicate symbols**
- **Construct model by finding clauses which are true**

Deciding whether a model exists is undecidable in FOPC!
There may be infinitely many models!

3

Some Problems with Model Construction for Scene Interpretation

Mapping φ

Establish mapping between real-world objects (as delivered by image analysis procedures) and constant symbols (as used in symbolic knowledge representation)

Problems: Segmentation performance, real-world objects not visible in a scene

Mapping π

Establish mapping between procedures which compute real-world relations (e.g. "touch") and predicate symbols of symbolic knowledge representation.

Problems: View-based procedures vs. 3D real-world relations, classification uncertainty

True clauses

Establish that all clauses of the symbolic knowledge base are true for the mappings φ and π.

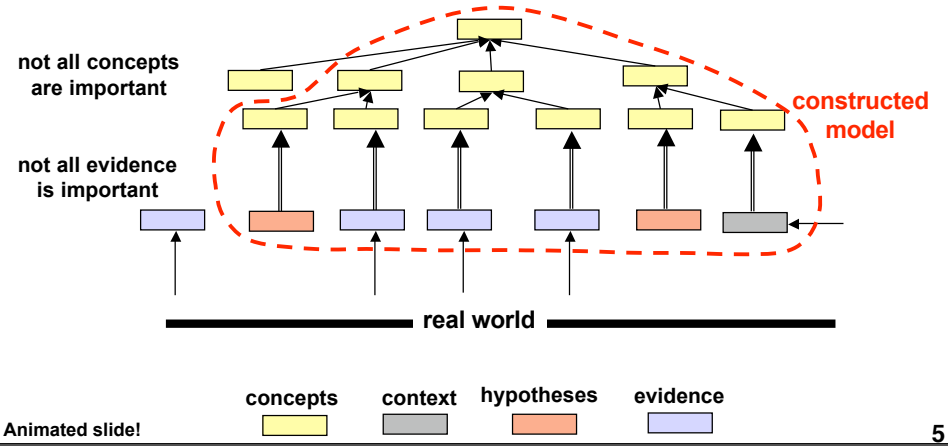
Problems: Many clauses of the knowledge base may be irrelevant for a concrete scene. A partial model may suffice for the vision task on hand.

4

So what is a Scene Interpretation?

Intuitively:

A scene interpretation is a scene description in terms of instantiated concepts consistent with evidence and context information.



Animated slide!

5

Partial Models

It seems plausible that a scene interpretation must not be proved to be consistent with all clauses of the conceptual knowledge base.

Example: Outdoor knowledge (e.g. about street traffic behavior) may not be relevant for indoor scenes (e.g. setting a table).

But there may be scenes where knowledge beyond the concrete scene may influence the interpretation.

Example: Knowing that a person has arrived outside of a house may affect the expected behavior of persons inside.

A good knowledge base provides aggregate concepts for all interrelated entities, often overlapping specific domains. In general, any two conceptual entities may be (indirectly) structurally connected (s. example next slide).

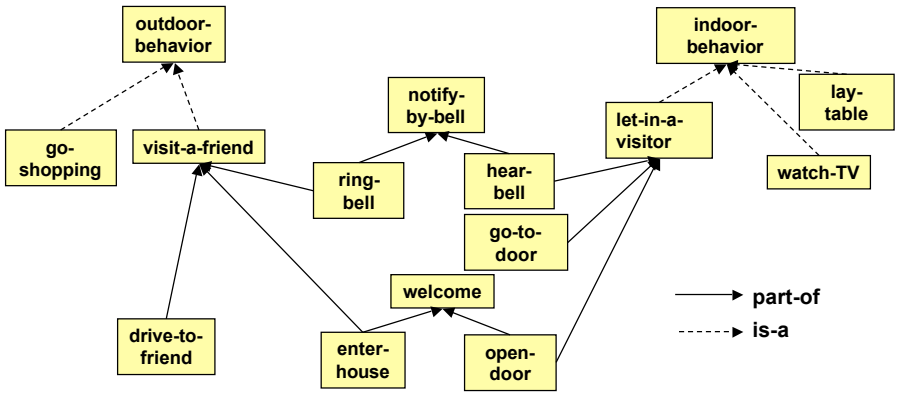
Partiality of scene interpretations depends on vision goals and context, not on structural boundaries of the conceptual knowledge base.

6

Interrelated Domains

Conceptual entities in seemingly disjoint domains may be interrelated, hence model construction for scene interpretations cannot be restricted by obvious boundaries.

Example: Outdoor behavior connected to indoor behavior



7

Finite Model Construction (Reiter & Mackworth 87)

- **An image consists of regions and chains (edges)**
- **The image elements constitute all constant symbols of an interpretation (domain closure assumption)**
- **Different constant symbols denote different image elements and vice versa (unique name assumption)**

Problem can be expressed in Propositional Calculus and solved as a constraint satisfaction problem.

For MAPSEE, scene interpretation amounts to finding a mapping p for predicates *road, river, shore, land, water.*

8

Constructing Partial Models

If image analysis provides the intended mappings φ and π from symbols into a real-world domain, model construction amounts to instantiating clauses of the conceptual knowledge base such that all clauses are true.

The interpretation steps introduced earlier allow to instantiate all concepts of the knowledge base.

Evidence matching
Assign evidence to object view classes or verify view hypotheses.

Aggregate instantiation
Infer an aggregate from (not necessarily all) parts.

Instance specialization
Refine instances along specialisation hierarchy or in terms of aggregate parts.

Instance expansion
Instantiate parts of an instantiated aggregate.

Instance merging
Merge identical instances constructed by different interpretation steps.

9

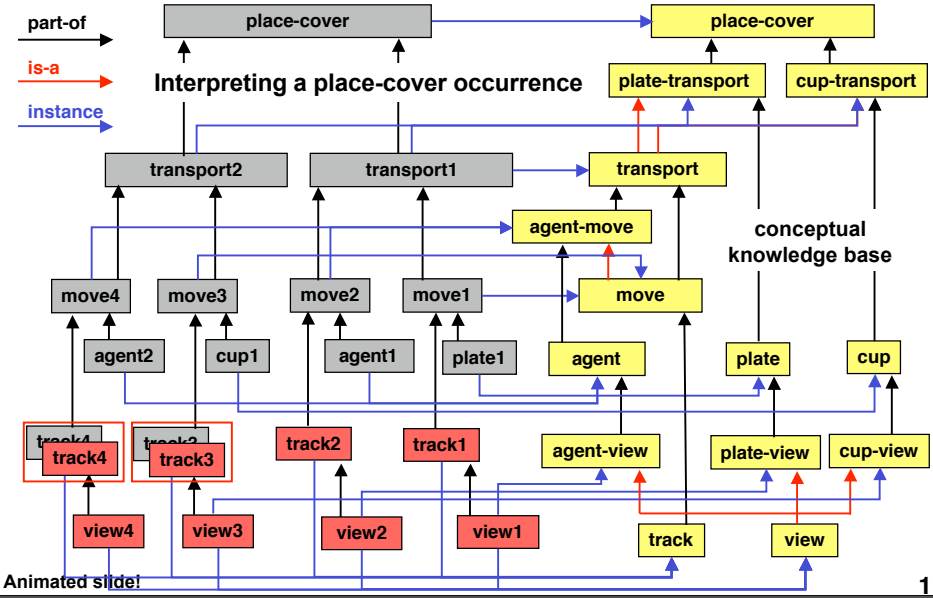
Practical Requirements for Partial Logical Models

- **Task-dependent scope and abstraction level**
  - **no need for checking all predicates**
    e.g. propositions outside a space and time frame may be uninteresting
  - **no need for maximal specialization**
    e.g. geometrical shape of "thing" suffices for obstacle avoidance
- **Partial model may not have consistent completion**
  - **uncertain propositions due to inherent ambiguity**
  - **predictions may be falsified**
- **Real-world agents need single "best" scene interpretation**
  - **requires uncertainty rating for evidence and context (propositions)**
  - **requires preference measure for scene interpretations**

⇒ **Logical model property provides only loose frame for possible scene interpretations.**

10

Example for Stepwise Model Construction



11

Description Logics

12

Description Logics for Knowledge Representation

DLs are a family of knowledge-representation formalisms

- **object-centered, roles and features (binary relations)**
- **necessary vs. sufficient attributes**
- **inference services**
  - subsumption check
  - consistency check
  - classification
  - abstraction
  - default reasoning
  - spatial and temporal reasoning
- **guaranteed correctness, completeness, decidability and complexity properties**
- **highly optimized implementations (e.g. RACER)**

13

Development of Description Logics

There exist several experimental and commercial developments of DLs, among them

- KL-ONE first conception of a DL (1985)
- CLASSIC commercial implementation by AT&T
- LOOM experimental system at USC
- FaCT experimental and commercial system (Horrocks, Manchester)
- RACER experimental and commercial system (Haarslev & Moeller)

There is active research on DLs:

- **extending the expressivity of concept languages**
- **decidability and tractability of inference services**
- **integration of predicates over concrete domains (e.g. numbers)**
- **highly optimized implementations**
- **developing new inference services (e.g. for scene interpretation)**

The Description Logic Handbook
F. Baader, D. Calvanese, D. MacGuinness, D. Nardi, P. Patel-Schneider (eds.)
Cambridge University Press, 2003

14

The RACER DL-System

- **Highly expressive DL** ALCQHI$_R$+
  - Role hierarchies with multiple parents
  - Qualified number restrictions (≥ n r C) (≤ n r C),
  - Inverse roles, transitive Roles
  - Integers and reals
- **Available as product RacerPro (http://www.racer-systems.com)**
  - Reasoner for the Semantic Web languages OWL/RDF
  - Evaluation copy for university research
  - Comprehensive manual
- **Developed in the Cognitive Systems Laboratory at Hamburg University**
  Research applications in
  - information management: TV-Assistant
  - content-based image retrieval
  - scene interpretation

see http://www.racer-systems.com/

15

RACER Concept Language

C	concept term
CN	concept name
R	role term
RN	role name

C -> *CN*
top
bottom
(not *C*)
(and *C1 ... Cn*)
(or *C1 ... Cn*)
(some *R C*)
(all *R C*)
(at-least *n R*)
(at-most *n R*)
(exactly *n R*)
(at-least *n R C*)
(at-most *n R C*)
(exactly *n R C*)
CDC

concept definition
(equivalent *CN C*)

concept axioms
(implies *CN C*)
(implies *C1 C2*)
(equivalent *C1 C2*)
(disjoint *C1 ... Cn*)

roles
R -> *RN*
(*RN role-props*)

role-props ->
((:transitive t)
(:feature t)
(:symmetric t)
(:reflexive t)
(:inverse CN)
(:domain CN)
(:range CN))

concrete-domain concepts
AN *attribute name*

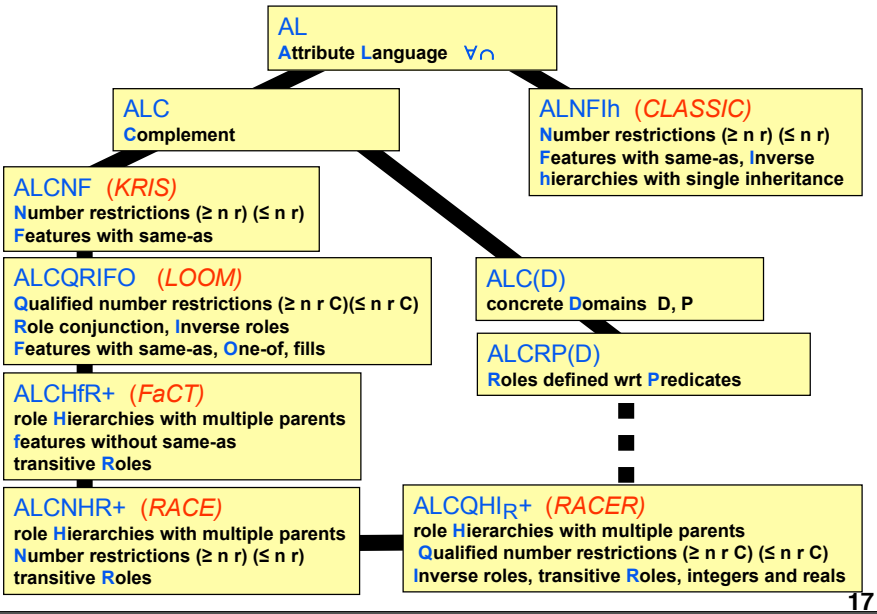
CDC -> (a *AN*)
(an *AN*)
(no *AN*)
(min *AN integer*)
(max *AN integer*)
(> *aexpr aexpr*)
(>= *aexpr aexpr*)
(< *aexpr aexpr*)
(<= *aexpr aexpr*)
(= *aexpr aexpr*)

aexpr -> *AN*
real
(+ *aexpr1 aexpr1**)
aexpr1

aexpr1 -> *real*
AN
(* *real AN*)

16

Family of Description Logics



17

Primitive and Defined Concepts

Concept expressions of a DL describe classes of entities in terms of properties (unary relations) and roles (binary relations).

The main building blocks are primitive oder defined concepts.

Primitive concepts: **concept => satisfied properties and relations**

satisfied properties and relations are necessary conditions for an object to belong to a class

Defined concepts: **concept <=> satisfied properties and relations**

satisfied properties and relations are necessary and sufficient conditions for an object to belong to a class

Primitive concept "person":
(implies person (and human (some has-gender (or female male))))

Defined concept "parent":
(equivalent parent (and person (some has-child person)))

18

Example of a TBox

```
(signature     :atomic-concepts  (person human female male woman man parent
                                  mother father grandmother aunt uncle sister brother)
               :roles      ((has-child :parent has-descendant)
                            (has-descendant :transitive t)
                            (has-sibling)
                            (has-sister :parent has-sibling)
                            (has-brother :parent has-sibling)
                            (has-gender :feature t)))
```

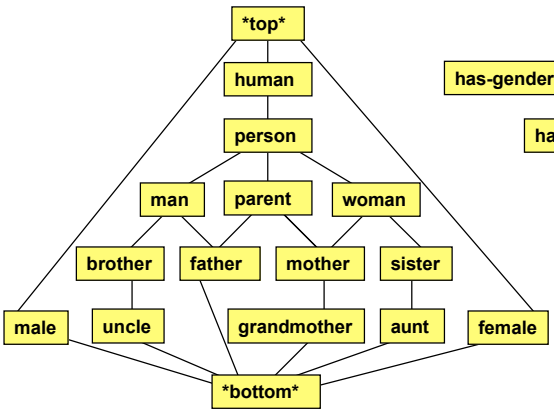
Signature of TBox

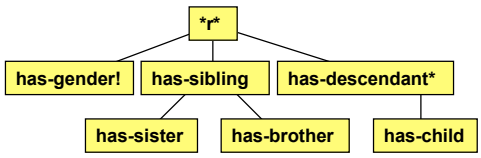
```
(implies person (and human (some has-gender (or female male))))
(disjoint female male)
(implies woman (and person (some has-gender female)))
(implies man (and person (some has-gender male)))
(equivalent parent (and person (some has-child person)))
(equivalent mother (and woman parent))
(equivalent father (and man parent))
(equivalent grandmother (and mother (some has-child (some has-child person))))
(equivalent aunt (and woman (some has-sibling parent)))
(equivalent uncle (and man (some has-sibling parent)))
(equivalent brother (and man (some has-sibling person)))
(equivalent sister (and woman (some has-sibling person)))
```

Concept axioms

19

Concept and Role Hierachies Implied by TBox





r universal role
! attribute (feature)
* transitive role

20

TBox Inferences

A DL system offers several inference services. At the core is a consistency test:

$$C \stackrel{?}{\models} \text{*bottom*}$$ **(the empty concept)**

Example: (and (at-least 1 has-child) (at-most 0 has-child)) $\models$ *bottom*

Consistency checking is the basis for several other inference services:

- **subsumption**
  (implies C1 C2) <=> (and C1 (not C2)) $\models$ *bottom*
- **classification of a concept expression**
  searches the existing concept hierarchy for the most special concept which subsumes the concept expression

Formal Semantics of Concept Expressions

D	**Set of all possible domain objects**
$E[C] \subseteq D$	**Extension of a concept expression C (represents meaning of C)**
$E[RN] \subseteq D \times D$	**Extension of a role RN (represents meaning of RN)**

Formal semantics of concept operations:

E[*bottom*] = { }

E[(and C_1 ... C_n)] = $E[C_1] \cap ... \cap E[C_n]$

E[(or C_1 ... C_n)] = $E[C_1] \cup \cdots \cup E[C_n]$

E[(all RN C)] = $\{x \mid \forall (x, y) \in E[RN] \Rightarrow y \in E[C]\}$

E[(some RN C)] = $\{x \mid \exists (x, y) \in E[RN] \wedge y \in E[C]\}$

ABox of a Description Logic System

TBox = terminological knowledge (concepts and roles)
ABox = assertional knowledge (facts)

An ABox contains:

- concept assertions (instance IN C)
  individual IN is instance of a concept expression C
- role assertions (related IN_1 IN_2 RN)
  individual IN_1 is related to IN_2 by role RN

- An ABox always refers to a particular TBox.
- An ABox requires unique names
- ABox facts are assumed to be incomplete (OWA).

  OWA = Open World Assumption
  (new facts may be added, hence inferences are restricted)

  CWA = Closed World Assumption
  (no facts may be added)

23

ABox Inferences

ABox inferences = inferring facts about ABox individuals

Typical queries:

- consistency — *is ABox consistent?*
- retrieval — *which individuals satisfy a concept expression?*
- classification — *what are the most special concept names which describe an individual?*

ABox consistency checking is in general more complicated than TBox consistency checking.

ABox consistent <=> there exists a "model" for ABox and TBox

All ABox inferences are based on the ABox consistency check.

24

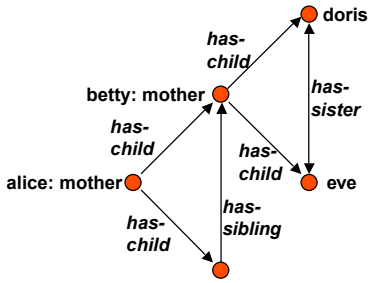
Example of ABox Queries

Contents of ABox

(instance alice mother)
(related alice betty has-child)
(related alice charles has-child)

(instance betty mother)
(related betty doris has-child)
(related betty eve has-child)

(instance charles brother)
(related charles betty has-sibling)
(instance charles (at-most 1 has-sibling))
(related doris eve has-sister)
(related eve doris has-sister)



charles: (and brother (at-most 1 has-sibling))

Questions and answers

(individual-instance? doris woman)
T
Is doris *instance of the concept* woman*?*

(individual-types eve)
((sister) (woman) (person) (human) (*top*))
Of which concept names is eve *an instance?*

(individual-fillers alice has-descendant)
(doris eve charles betty)
What are the descendants of eve*?*

(concept-instances sister)
(doris betty eve)
Which instances has the concept sister*?*

25

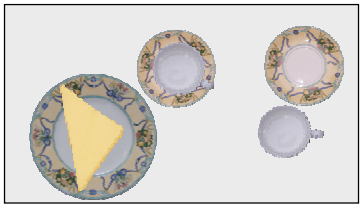
Table-Top Scene Description

TBox (excerpt):

```
(implies plate dish)
(implies saucer dish)
(implies cup dish)
(implies napkin cloth)
(equivalent cover
        (and configuration
            (exactly 1 has-part plate)
            (exactly 1 has-part (and saucer (some near plate)))
            (exactly 1 has-part (and cup (some on saucer)))
```

ABox (excerpt):

```
(instance plate1 plate)
(instance saucer1 saucer)
(instance saucer2 saucer)
(instance cup1 cup)
(instance cup2 cup)
(instance napkin1 napkin)
(instance cover1 cover)
(related saucer1 plate1 near)
((related cup1 saucer1 on)
(related napkin1 plate1 on)
```



Queries for Table-Top Scene Description

Queries:

(concept-instances cover)
⇒ (cover1)

(concept-instances (some on dish))
⇒ (cup1 napkin1)

(concept-instances (and cloth (some on plate))
⇒ (napkin1)

(concept-instances (not (some on saucer)))
⇒ ( ) *for OWA - a fact* (related (cup2 saucer3 on)) *could be added*
⇒ (cup2) *for CWA*

RACER Query Language

Interface language for retrieving patterns from an ABox

Basic retrieval command:

(retrieve <list-of-objects> <query-body>)

Example:

```
(retrieve (?x ?y ?z) (and (?x plate)
                          (?y saucer)
                          (?z cup)
                          (?x ?y near)
                          (?z ?y on)))
```

⇒ (((?x plate1) (?y saucer1) (?z cup1))
((?x plate2) (?y saucer2) (?z cup2)))

Note: Query language retrieval commands allow to retrieve patterns for which no individuals have been introduced.

28

Useful Extensions

Feature chains: (compose F1 ... Fn) **short: (F1 ∘ ... ∘ Fn)**

The composition of features F1 ... Fn is a feature whose fillers are the fillers of Fn applied to the fillers of Fn-1 applied to ... the fillers of F1.

Feature (chain) agreement: (same-as F1 F2) **short: (= F1 F2)**

Concept expression for elements which possess the same fillers for features F1 and F2.

Example: (same-as (has-plate ∘ has-colour) (has-saucer ∘ has-colour))

Requirement for a cover that plate and saucer have the same colour

Cannot be combined with expressive DLs without jeopardising decidability!

Instead of features, also roles may be composed, and a subset operator relates role-fillers similar to same-as for features.

Role-value map: (subset R1 R2)

Concept expression of elements where the fillers of role R1 are a subset of the fillers of role R2.

Causes undecidability even in DLs with low expressivity (e.g. CLASSIC).

29

Abstraction with Description Logics

Abstraction = omission of properties or relations, extending a concept, generalization

Examples:

- **Superordinate concept name of a concept expression (= concept classification)**
  (and person (some has-size tall)) → person
- **Generalization of concept expressions**
  (and (some has-occupation professor) (at-least 3 has-child))
  ⇩
  (and (some has-occupation civil-servant) (at-least 1 has-child))
- **Concept expression which subsumes several individuals**
  1. classify individuals
  2. determine least common subsumer (LCS)
  - **for RACER: trivial solution in terms of (OR C_1 ... C_n)**
  - **for DLs without OR: special abstraction operator LCS**

30