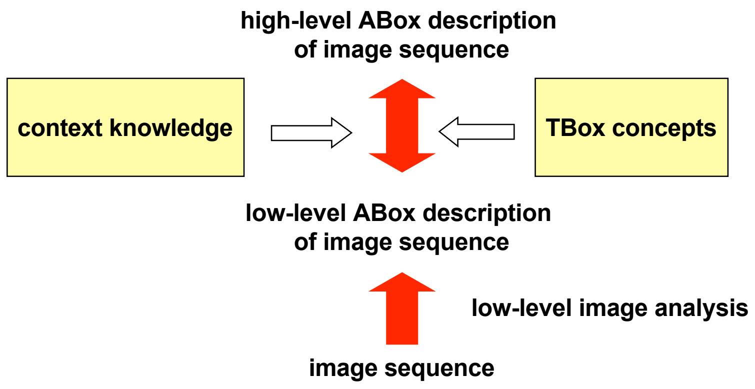
Using Description Logics for Scene Interpretation

Basic Structure for Scene Interpretation with a DL System



high-level ABox description
of image sequence

context knowledge

TBox concepts

low-level ABox description
of image sequence

low-level image analysis

image sequence

Meeting Basic Representational Requirements with a DL System

- object oriented representations
  yes, but needs user interface
- n-ary relations
  no, only binary relations
- taxonomies
  yes, automatically constructed from conceptdefinitions
- partonomies
  yes, can be represented by roles
- spatial and temporal relations
  can be computed from quantitative data via concrete domain extensions
- qualitative predicates
  can be computed from quantitative data via concrete domain extensions

---

Concrete Domain Concepts in RACER

```
CDC →      (a  AN)  (an  AN)
           (no  AN)
           (min  AN  integer)
           (max  AN  integer)
           (equal  AN  integer)
           (>  aexpr  aexpr)
           (>=  aexpr  aexpr)
           (<  aexpr  aexpr)
           (<=  aexpr  aexpr)
           (=  aexpr  aexpr)
aexpr →    AN
           real
           (+  aexpr1  aexpr1*)
           aexpr1
aexpr1 →   AN
           real
           (*  real  AN)
```

Example:
Quantitative constraints on the size of an object

(and (min size 13) (max size 20))

↑ ↑

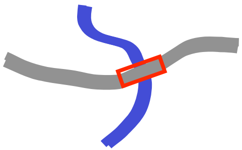
integer-valued attribute "size" receives values from low-level vision

DL Concept for a Cover

```
(equivalent  cover
            (and  configuration
                  (exactly  1  cv-pl  plate)
                  (exactly  1  cv-sc  (and  saucer  (some  near plate)))
                  (exactly  1  cv-cp  (and  cup  (some  on  saucer)))
                  (subset  cv-pl  (compose  cv-sc  near))
                  (subset  cv-sc  (compose  cv-cp  on))))
```

- **parts are expressed as qualified fillers of specific roles**
  e.g. cv-pl, cv-sc, cv-scp
- **sameness (or distinctness) of parts and properties of parts are expressed by the subset construct**
- **spatial constraints are modelled as primitive predicates**
  e.g. near, on

Example: DL Model for a Bridge



Assumptions:
Image analysis computes bottom-up

- **strips (= lengthy regions)**
- **colours**
- **spatial relations (touch, contain)**

TBox:

```
(equivalent bridge
     (and strip-section
          (some has-road road)
          (some has-river1 river)
          (some has-river2 river)
          (subset has-road ∘ contain)
          (subset has-river1 ∘ touch)
          (subset has-river2 ∘ touch)))
```

```
(equivalent strip-section
     (and (some within strip)
          (= has-width within ∘ has-width)))

(equivalent road
     (and strip
          (some has-colour road-colour)))

(equivalent river
     (and strip
          (some has-colour river-colour)))
```

Example ABox:

```
(instance strip1 strip)
(instance strip2 strip)
(instance strip3 strip)
• • •
```

```
(related strip1 blue has-colour)
(related strip2 blue has-colour)
(related strip3 greyhas- colour)
• • •
```

```
(related strip1 strip3  touch)
(related strip2 strip3  touch)
(related strip3 strip1 touch)
(related strip3 strip2 touch)
• • •
```

Problem: Generating instances of strip-section

Animated slide!

Simplified DL Concept for Placing a Cover

```
(equivalent  place-cover
        (and  agent-activity
                (exactly  1  pc-tp1  (and  transport  (some  tp-obj  plate)))
                (exactly  1  pc-tp2  (and  transport
                                                (some  tp-obj  saucer)
                                                (some  before  (and  transport  (some tp-obj  cup))))
                (exactly  1  pc-tp3  (and  transport  (some tp-obj  cup)))
                (subset  pc-tp3  (compose  pc-tp2  before))))
```

Severe disadvantage of purely symbolic spatial and temporal constraints:

Pairwise constraints must be computed bottom-up by low-level vision procedures irrespective of high-level concepts!

➡ **Express spatial and temporal constraints as predicates over concrete-domain elements**

Quantitative Spatial and Temporal Constraints

```
(equivalent  place-cover
        (and  agent-activity
                (exactly  1  pc-tp1  (and  transport  (some  tp-obj  plate))
                (exactly  1  pc-tp2  (and  transport  (some tp-obj  saucer))
                (exactly  1  pc-tp3  (and  transport  (some tp-obj  cup))
                (<=  pc-tp2 o tp-end  pc-tp3 o tp-end)
                (=  pc-beg  (minim  pc-tp1 o tp-beg  pc-tp2 o tp-beg  pc-tp3 o tp-beg))
                (=  pc-end  (maxim  pc-tp1 o tp-end  pc-tp2 o tp-end  pc-tp3 o tp-end))
                (<=  (-  pc-end  pc-beg)  max-duration))))
```

- **Equality and inequality as concrete domain predicates**
- **Specific constraints for each concept**
- **Incremental constraint computation required for prediction!**

  Example: (and (= cv-sc o sc-loc cv-cp o cp-loc))

  Known saucer position restricts expected cup positions

General Structure for Aggregate Definitions

```
(equivalent  <concept-name>
             (and  <parent-concept1> ... <parent-conceptN>
                   (<number-restriction1>  <role-name1>  <part-concept1>)
                   . . .
                   (<number-restrictionK>  <role-nameK>  <part-conceptK>)
                   <constraints between parts>))
```

Summary of DL constructs required for aggregates: ALCF(D)

=> aggregates can in principle be represented in RACER, however, not all syntax features are currently available

---

DL Reasoning Services

ABox consistency checking is at the heart of all reasoning services

Model construction is the method of choice for many DL reasoners

- **Concept satisfiability**
- **Concept subsumption**
- **Concept disjointness**
- **Concept classification**
- **TBox coherence**
- **ABox consistency w.r.t. a TBox**
- **Instance checking**
- **Most-specific atomic concepts of which an individual is an instance**
- **Instances of a concept**
- **Role fillers for a specified individual**
- **Pairs of individuals related by a specified role**
- **Conjunctive queries**

DL Reasoning Support for Scene Interpretation

- **Maintaining a coherent knowledge base**

  Scene interpretation may require extensive common-sense knowledge, intuitive knowledge representation is doomed
- **Maintaining consistent scene interpretations**

  A consistent ABox is a (partial) model and hence formally a (partial) scene interpretation => ABox consistency checking ensures consistent scene interpretations

ABox realization (computing most specific concepts for individuals) cannot be used in general:

- **scene interpretations cannot be deduced**
- **high-level individuals must be hypothesized before consistency check**

DL Support for Interpretation Steps

Aggregate instantiation
Determine aggregates for which an individual is a role filler
⇒ RACER query language

Instance specialization
Retrieve all specializations of a given concept
⇒ use specialization hierarchy

Instance expansion
Instantiate parts of an aggregate instance
⇒ easy service by looking up the aggregate definition

Instance merging
Determine whether it is consistent to unify two individual descriptions
=> unification by recursive specialization can be supported

Important missing service:
Preference measure for choosing "promising" alternatives

Extending Description Logics for Default Reasoning

Defaults for Preferences

Idea:

If deductive rules are not suitable for scene interpretation, why not use default rules which may apply in general but allow exceptions?

"Default rule" = inference rule in a situation lacking decisive knowledge

Classical example of AI literature:

All birds can fly.

Penguins cannot fly.

Tweety is a bird. => Tweety can fly.

Tweety is a penguin. => Tweety cannot fly.

↓ **nonmonotonic reasoning**

If the logical framework allows several interpretations, default rules may be used to select a preferred interpretation.

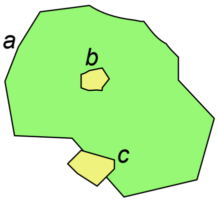
Terminological Default Theories

- **Default theory (W,D)**
  W = world description, D = set of defaults
- **Default rules [Reiter, 1980]**

$$\frac{\alpha \mid \beta 1, \beta 2, \ldots, \beta n}{\gamma}$$

  You may conclude γ if prerequisite α is true and γ is consistent with $\beta 1 \ldots \beta n$
- **Different sets of extensions of (W,D)**
  skeptical vs. credulous consequence
- **Terminological default theories [Baader & Hollunder, 1991]**
  - α, β, γ concept terms
  - W = ABox, D = set of closed default rules
  - restricted semantics, no skolemization
  - concept terms become ABox membership assertions
  - consequence problem decidable

Example: Hypothesis Generation Using Default Rules (1)



World description *W*
a : country
b : area
c : area
(a,b) : contains
(b,a) : inside
(a,c) : overlaps
(c,a) : overlaps

Task:
Generate hypotheses for a, b, c

Default rules

$$\frac{area \mid country}{country}$$

$$\frac{area \mid city}{city}$$

$$\frac{area \mid lake}{lake}$$

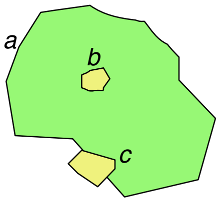
⇨

Default rules *D* closed over *W*

$\frac{a : area \mid a : country}{a : country}$	$\frac{a : area \mid a : city}{a : city}$	$\frac{a : area \mid a : lake}{a : lake}$
$\frac{b : area \mid b : country}{b : country}$	$\frac{b : area \mid b : city}{b : city}$	$\frac{b : area \mid b : lake}{b : lake}$
$\frac{c : area \mid c : country}{c : country}$	$\frac{c : area \mid c : city}{c : city}$	$\frac{c : area \mid c : lake}{c : lake}$

Animated slide!

Example: Hypothesis Generation Using Default Rules (2)



World description *W*

a : *country*
b : *area*
c : *area*
(*a*,*b*) : *contains*
(*b*,*a*) : *inside*
(*a*,*c*) : *overlaps*
(*c*,*a*) : *overlaps*

Extension E1

b : *city*
c : *lake*

Extension E2

b : *lake*
c : *lake*

2 mutually exclusive extensions E1 and E2

Default rules

$$\frac{\mathit{area} \mid \mathit{country}}{\mathit{country}} \qquad \frac{\mathit{area} \mid \mathit{city}}{\mathit{city}} \qquad \frac{\mathit{area} \mid \mathit{lake}}{\mathit{lake}}$$

Default rules *D* closed over W

$$\frac{a : \mathit{area} \mid a : \mathit{country}}{a : \mathit{country}} \qquad \frac{a : \mathit{area} \mid a : \mathit{city}}{a : \mathit{city}} \qquad \frac{a : \mathit{area} \mid a : \mathit{lake}}{a : \mathit{lake}}$$

$$\frac{b : \mathit{area} \mid b : \mathit{country}}{b : \mathit{country}} \qquad \frac{b : \mathit{area} \mid b : \mathit{city}}{b : \mathit{city}} \qquad \frac{b : \mathit{area} \mid b : \mathit{lake}}{b : \mathit{lake}}$$

$$\frac{c : \mathit{area} \mid c : \mathit{country}}{c : \mathit{country}} \qquad \frac{c : \mathit{area} \mid c : \mathit{city}}{c : \mathit{city}} \qquad \frac{c : \mathit{area} \mid c : \mathit{lake}}{c : \mathit{lake}}$$

Spatioterminological Background Knowledge for Example

TBox

building_region = area ∩ ∃ (has_area) . building_features
natural_region = ¬building_region
country_region ⊆ building_region ∩ large_area
city_region = building_region ∩ ¬large_area
river_region ⊆ natural_region ∩ area
lake_region ⊆ natural _region ∩ area
country = country_region ∩ ∀contains . ¬country_region ∩
∀overlaps . ¬country_region ∩
∀inside . ¬country_region
city = city_region ∩ ∃inside . country_region
lake ⊆ lake_region
river ⊆ river_region ∩ "overlaps . ¬lake_region ∩
∀inside . ¬lake_region ∩
∀contains . ⊥
river_flowing_into_lake = river ∩ ∃touches . ¬lake_region

Spatioterminological Default Theories with „ABox Patterns“

Default rules with ABox patterns instead of concept terms:

precondition pattern | justification pattern
consequence pattern

=> use concept memberships to conclude relationships

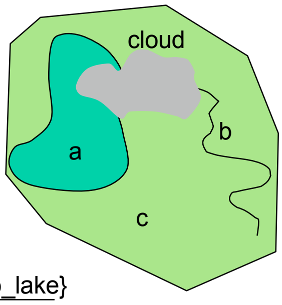
=> use relationships to conclude concept memberships

Example: **{ X: lake, Y: river } | { (X, Y) : disjoint }**
{ (X, Y) : disjoint }

Conclude that a lake and a river are disjoint as long as this does not lead to inconsistency.

Example: River Flowing into Lake

ABox
a : lake
b : river
c : country



Default rules

{ X: lake, Y: river_flowing_into_lake } | { (X, Y) : touches }
{ (X, Y) : touches }

{ X: river, Y: country, (X, Y) : inside} | {X : river_flowing_into_lake}
{X : river_flowing_into_lake}

⇩

Extension
a : river_flowing_into_lake
(a, b) : touches

Animated slide!

How Useful are Defaults for Scene Interpretation?

- Defaults can be used as preference rules for the selection of interpretation steps
- Defaults can be integrated into reasoning services
- Default reasoning (computing extensions) is computationally expensive
- Defaults are domain and task dependent
- Defaults become unwieldy if their number grows
  (compare with rule-based expert systems)