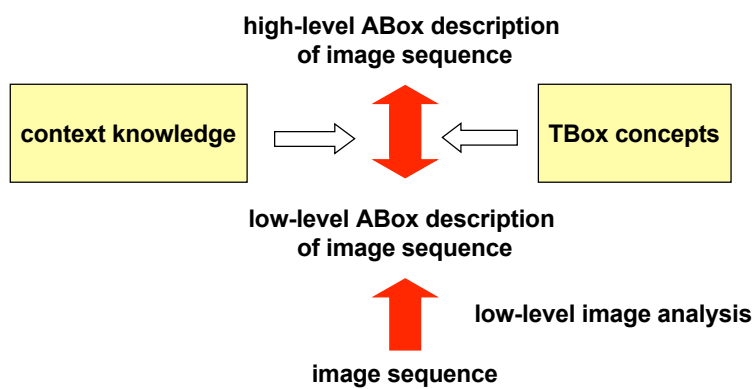


Using Description Logics for Scene Interpretation

1

Basic Structure for Scene Interpretation with a DL System



2

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

3

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

4

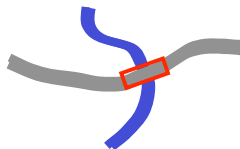
DL Concept for a Cover

```
(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-cp
- **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

5

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 o contain)
  (subset has-river1 o touch)
  (subset has-river2 o touch)))
```

```
(equivalent strip-section
 (and (some within strip)
  (= has-width within o 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)      (related strip1 blue has-colour)  (related strip1 strip3 touch)
(instance strip2 strip)      (related strip2 blue has-colour)  (related strip2 strip3 touch)
(instance strip3 strip)      (related strip3 greyhas- colour)  (related strip3 strip1 touch)
...                          ...                                  (related strip3 strip2 touch)
...                          ...                                  ...
```

Problem: Generating instances of strip-section

Animated slide!

6

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**

7

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

8

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

9

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

10

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**

11

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**

12

Extending Description Logics for Default Reasoning

13

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. \Rightarrow Tweety can fly.

Tweety is a penguin. \Rightarrow Tweety cannot fly.

↓
nonmonotonic
reasoning

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

14

Terminological Default Theories

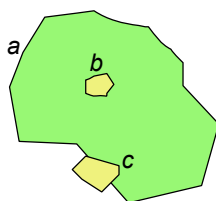
- **Default theory (W,D)**
W = world description, D = set of defaults
- **Default rules [Reiter, 1980]**

$$\frac{\alpha | \beta_1, \beta_2, \dots, \beta_n}{\gamma}$$

You may conclude γ if prerequisite α is true and γ is consistent with $\beta_1 \dots \beta_n$
- **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

15

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

$\frac{\text{area} | \text{country}}{\text{country}}$
 $\frac{\text{area} | \text{city}}{\text{city}}$
 $\frac{\text{area} | \text{lake}}{\text{lake}}$



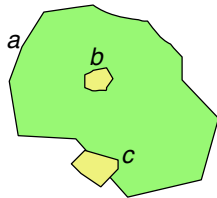
Default rules D closed over W

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

Animated slide!

16

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

$\frac{\text{area} \mid \text{country}}{\text{country}}$
 $\frac{\text{area} \mid \text{city}}{\text{city}}$
 $\frac{\text{area} \mid \text{lake}}{\text{lake}}$

Default rules D closed over W

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

17

Spatioterminological Background Knowledge for Example

TBox

building_region = area \cap \exists (has_area) . building_features
 natural_region = \neg building_region
 country_region \subseteq building_region \cap large_area
 city_region = building_region \cap \neg large_area
 river_region \subseteq natural_region \cap area
 lake_region \subseteq natural_region \cap area
 country = country_region \cap \forall contains . \neg country_region \cap
 \forall overlaps . \neg country_region \cap
 \forall inside . \neg country_region
 city = city_region \cap \exists inside . country_region
 lake \subseteq lake_region
 river \subseteq river_region \cap "overlaps . \neg lake_region \cap
 \forall inside . \neg lake_region \cap
 \forall contains . \perp
 river_flow_into_lake = river \cap \exists touches . \neg lake_region

18

Spatioterminological Default Theories with „ABox Patterns“

Default rules with ABox patterns instead of concept terms:

$$\frac{\text{precondition pattern} \mid \text{justification pattern}}{\text{consequence pattern}}$$

=> use concept memberships to conclude relationships

=> use relationships to conclude concept memberships

Example: $\frac{\{ X: \text{lake}, Y: \text{river} \} \mid \{ (X, Y) : \text{disjoint} \}}{\{ (X, Y) : \text{disjoint} \}}$

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

19

Example: River Flowing into Lake

ABox

a : lake
b : river
c : country

Default rules

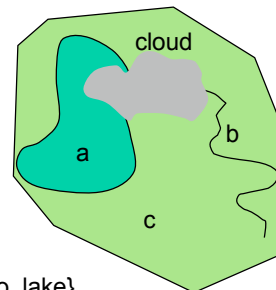
$\frac{\{ X: \text{lake}, Y: \text{river_flowing_into_lake} \} \mid \{ (X, Y) : \text{touches} \}}{\{ (X, Y) : \text{touches} \}}$

$\frac{\{ X: \text{river}, Y: \text{country}, (X, Y) : \text{inside} \} \mid \{ X: \text{river_flowing_into_lake} \}}{\{ X: \text{river_flowing_into_lake} \}}$



Extension

a : river_flow_into_lake
(a, b) : touches



Animated slide!

20

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)