Description Logics: A Logical Foundation of the Semantic Web and its Applications

Volker Haarslev
Concordia University, Computer Science Department
1455 de Maisonneuve Blvd. W.
Montreal, Quebec H3G 1M8, Canada

http://www.cs.concordia.ca/~faculty/haarslev/
haarslev@cs.concordia.ca

Idea of the Semantic Web

Tim Berners-Lee, James Hendler, Ora Lassila: *The Semantic Web*

- World Wide Web
  - medium of
    - documents for people rather than of
    - information that can be manipulated automatically
  - augment web pages with data targeted at computers
  - add documents solely for computers
  - called semantic markup
- ...transforms into the Semantic Web
- Find meaning of semantic data by following
  - hyperlinks to definitions of key terms and
  - rules for reasoning about data logically
- Spur development of automated web services
  - highly functional agents

Typical Information Retrieval Example

- Suppose you are a salesperson, who wishes to find a Ms. Cook you met at a trade conference last year
  - you don't remember her first name but
  - you remember she worked for one of your clients and
  - her daughter is a student of your alma mater
- An intelligent search agent can
  - ignore pages relating to cooks, cookies, Cook Islands, etc.
  - find pages of companies your clients are working for
  - follow links to or find private home pages
  - check whether a daughter is still in school
  - match with students from your alma mater
- If you already have the Semantic Web

Basic Web Technology

- Uniform Resource Identifier (URI)
  - foundation of the Web
  - identify items on the Web
  - uniform resource locator (URL): special form of URI
- Extensible Markup Language (XML)
  - send documents across the Web
  - allows anyone to design own document formats (syntax)
  - can include markup to enhance meaning of document's content
  - machine readable
- Resource Description Framework (RDF)
  - make machine-processable statements
  - triple of URIs: subject, predicate, object
  - intended for information from databases

Schemas and Ontologies for the Web

- Usual assumption: data is nearly perfect
  - book rating with scale 1-10 instead of really_good,...,really _bad
  - conversion without meaning difficult
  - information newly tagged with has_author instead of creator_of
- Even worse: URIs have no meaning
- Solution: schemas and ontologies
- RDF Schemas: author is subclass of contributor
- DARPA Agent Markup Language with Ontology Inference Layer (DAML+OIL)
  - add semantics: has_author is the inverse relation of creator_of
  - now we understand the meaning of has_author
  - has_author(book,author) ≡ creator_of(author,book)

A Logical Foundation for the Semantic Web

- Systems can understand basic concepts such as
  - subclass
  - inverse relation, etc.
- Even better
  - state (any) logical principle
  - permit computers to reason (by inference) using these principles
  - *an employee sells more than 100 items per day ⇒ bonus*
  - follow semantic links to construct a proof for your conclusions
  - exchange proofs between agents (and human users)

- DAML+OIL is a syntactic variant of a well-known and very expressive description logic

Why Description Logics?

- Designed to represent knowledge
- Based on formal semantics
- Inference problems have to be decidable
- Probably the most thoroughly understood set of formalisms in all of knowledge representation
- Computational space has been thoroughly mapped out
- Wide variety of systems have been built
  - however, only very few highly optimized systems exist
- Wide range of logics developed
  - from very simple (no disjunction, no full negation)
  - to very expressive (comparable to DAML+OIL)
- Very tight coupling between theory and practice

Description Logics: Introduction (1)

- Origins
  - structured inheritance networks
  - frame-based representations
- Factual world
  - named individuals, e.g., charles, elizabeth
  - (binary) relationships between individuals, e.g., has_child
- Descriptions form hierarchical knowledge
  - two disjoint alphabets: concept and role names
  - roles denote binary descriptions, e.g., has_child(x,y)
  - concepts denote unary descriptions, e.g.,
    $\text{parent}(x) \equiv \text{person}(x) \wedge$
    $\exists y : (\text{has\_child}(x,y) \wedge \text{person}(y))$

Description Logics: Introduction (2)

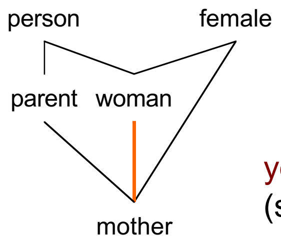
- Important syntactic feature: variable-free notation
  - constructors: $\sqcap$, $\sqcup$, $\neg$, $\exists$, $\forall$
  - standard description logic *ALC*
- Description of concept parent
  - parent $\equiv$ person $\sqcap$ $\exists$has_child.person
- We add two concepts
  - woman $\equiv$ female $\sqcap$ person
  - mother $\equiv$ female $\sqcap$ parent
- What type of inferences are interesting?
  - satisfiability of (named) concepts
  - subsumption of (named) concepts

Inference Service: Concept Satisfiability

- The concepts woman, mother, parent are satisfiable
- However, the concept $\neg$woman $\sqcap$ mother is unsatisfiable
- Why? We unfold the definition of woman and mother
  - $\neg$woman $\sqcap$ mother $\equiv$
  - $\neg$(female $\sqcap$ person) $\sqcap$ female $\sqcap$ parent $\equiv$
  - ($\neg$female $\sqcup$ $\neg$ person) $\sqcap$ female $\sqcap$ parent $\equiv$
  - ($\neg$female $\sqcup$ $\neg$ person) $\sqcap$ female $\sqcap$ parent $\equiv$
  - $\neg$person $\sqcap$ female $\sqcap$ parent $\equiv$
  - $\neg$person $\sqcap$ female $\sqcap$ person $\sqcap$ $\exists$has_child.person $\equiv$
  - $\neg$person $\sqcap$ female $\sqcap$ person $\sqcap$ $\exists$has_child.person
  - $\lightning$
- The conjunct $\neg$woman $\sqcap$ mother can never be satisfied

Inference Service: Concept Subsumption

- Consider the question
  Is a mother always a woman?
- Subsumes the concept woman the concept mother?
- Description logic reasoners offer the computation of a subsumption hierarchy (taxonomy) of all named concepts



parent ≡ person ⊓ ∃has_child.person
woman ≡ person ⊓ female
mother ≡ parent ⊓ female

yes, woman subsumes mother
(see also proof on previous slide)

Description Logics: Semantics (1)

- Translation to first-order predicate logic usually possible
- Declarative and compositional semantics preferred
- Standard Tarski-style interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$

Syntax	**Semantics**	
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, A is a concept name	Concepts
¬C	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	
C ⊓ D	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	
C ⊔ D	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	
∀R.C	$\{ x \in \Delta^{\mathcal{I}} \mid \forall y: (x,y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}} \}$	
∃R.C	$\{ x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}} \}$	
R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, R is a role name	Roles
C ⊑ D	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	Axioms
C ≡ D	$C^{\mathcal{I}} = D^{\mathcal{I}}$	

Description Logics: Concept Examples

- $\text{woman} \equiv \text{person} \sqcap \text{female}$
- $\text{parent} \equiv \text{person} \sqcap \exists\text{has\_child.person}$
- $\text{mother} \equiv \text{parent} \sqcap \text{female}$
- $\text{mother\_having\_only\_female\_kids} \equiv \text{mother} \sqcap \forall\text{has\_child.female}$
- $\text{mother\_having\_only\_daughters} \equiv \text{woman} \sqcap \text{parent} \sqcap \forall\text{has\_child.woman}$

equivalent

- $\text{grandma} \equiv \text{woman} \sqcap \exists\text{has\_child.parent}$
- $\text{great\_grandma} \equiv \text{woman} \sqcap \exists\text{has\_child.}\exists\text{has\_child.parent}$

Description Logics: Concept Examples

- $\text{woman} \equiv \text{person} \sqcap \text{female}$
- $\text{parent} \equiv \text{person} \sqcap \exists\text{has\_child.person}$
- $\text{mother} \equiv \text{parent} \sqcap \text{female}$
- $\text{mother\_having\_only\_female\_kids} \equiv \text{mother} \sqcap \forall\text{has\_child.female}$
- $\text{mother\_having\_only\_daughters} \equiv \text{woman} \sqcap \text{parent} \sqcap \forall\text{has\_child.woman}$

equivalent

- $\text{grandma} \equiv \text{woman} \sqcap \exists\text{has\_child.parent}$
- $\text{great\_grandma} \equiv \text{woman} \sqcap \exists\text{has\_child.}\exists\text{has\_child.parent}$

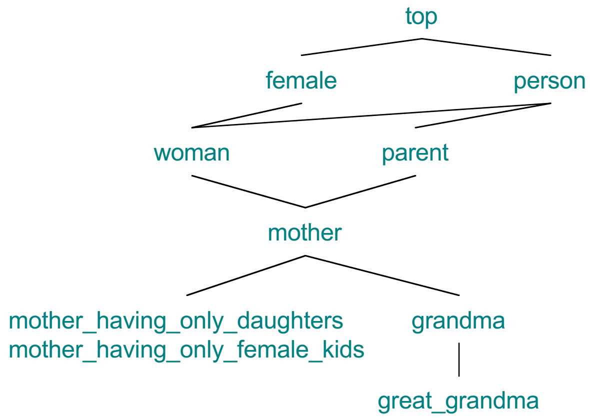
Description Logics: Semantics (2)

- Interpretation domain can be chosen arbitrarily
- Distinguishing features of description logics
  - domain can be infinite
  - open world assumption
- A concept C is satisfiable iff there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$
  - $\mathcal{I}$ is called a model of C
- Subsumption can be reduced to satisfiability
  - subsumes(C,D) $\Leftrightarrow$ $\neg$sat($\neg C \sqcap D$)
  - denoted as $C \sqsupseteq D$ or $D \sqsubseteq C$

Description Logics: TBox

- A collection of concept axioms is called a TBox (Terminological Box)
- Satisfiability of concepts defined w.r.t. a TBox $\mathcal{T}$
- Inference services
  - TBox coherence: List all unsatisfiable concept names in $\mathcal{T}$
  - compute subsumption hierarchy (taxonomy) of concept names in $\mathcal{T}$
- Why emphasize concept names?
  - ontological decisions of users
  - important concepts will be named

Example Taxonomy



Description Logics: Individuals

- How can we assert knowledge about individuals?
- Assertional axioms
  - concept assertion for an individual a
    - a:C satisfied iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$
    - example: elizabeth:mother
  - role assertion for two individuals a and b
    - (a,b):R satisfied iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
    - example: (elizabeth,charles):has_child
- Unique name assumption
  - Different names denote different individuals
  - $a^{\mathcal{I}} \neq b^{\mathcal{I}}$

Description Logics: ABox (1)

- A collection of assertional axioms is called an ABox (Assertional Box)
- Satisfiability of assertions defined w.r.t.
  - ABox $\mathcal{A}$
  - TBox $\mathcal{T}$
- Inference services
  - ABox satisfiability: Is the collection $\mathcal{A}$ of assertions satisfiable?
  - Instance checking: instance?(a,C,$\mathcal{A}$)
    Is a an instance of concept C or subsumes C the individual a?
  - ABox realization: compute for all individuals in $\mathcal{A}$ their most-specific concept names w.r.t. TBox $\mathcal{T}$

Description Logics: ABox (2)

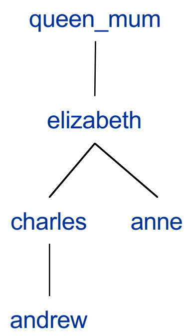
- New basic inference service: ABox satisfiability
  - asat($\mathcal{A}$)
- All other inference services can be reduced to asat
  - instance checking:
    instance?(a,C,$\mathcal{A}$) ≡ ¬asat($\mathcal{A}$ ∪ {a:¬C})
  - concept satisfiability:
    sat(C) ≡ asat({a:C})
  - concept subsumption:
    subsumes(C,D) ≡ ¬sat(¬C ⊓ D) ≡ ¬asat({a:¬C ⊓ D})
- Open world assumption
  - $\mathcal{A}$ = {andrew:male, (charles,andrew):has_child}
  - Does instance?(charles,∀has_child.male, $\mathcal{A}$) hold?

No.
Why?
(See later)

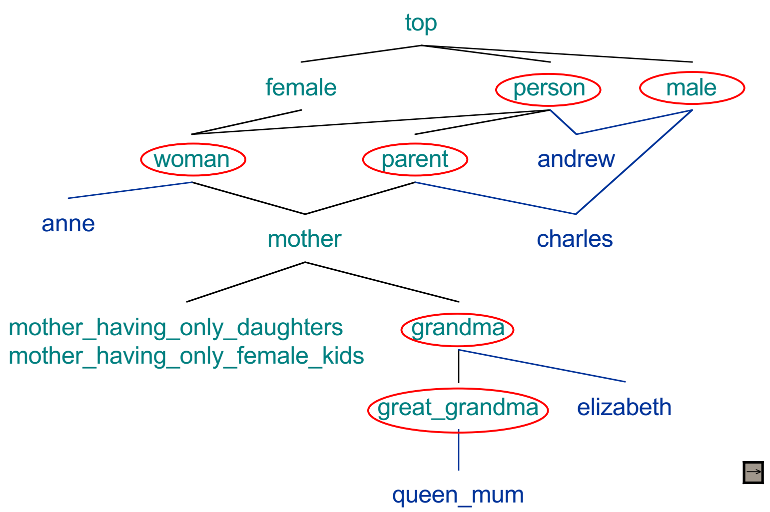
Description Logics: ABox Example

- (male $\sqsubseteq \neg$female) additional axiom ensuring disjointness

- queen_mum : woman
- (queen_mum,elizabeth) : has_child
- elizabeth : woman
- (elizabeth,charles) : has_child
- (elizabeth,anne) : has_child
- charles : parent $\sqcap$ male
- anne : woman
- (charles,andrew) : has_child
- andrew : person $\sqcap$ male



TBox Taxonomy plus Individuals



Open World Assumption

- Can we prove that instance?(charles,∀has_child.male,$\mathcal{A}$) holds?
- No. Although the ABox contains only knowledge about one male child, it is unknown whether additional information about a female child might be added later.
- In order to prevent this, we could add
  - charles : ∀has_child.male or
  - assert that information about a second child will not be addded in the future, i.e., close a role for an individual
  - Not possible in the logic *ALC* since we need so-called number restrictions

More Description Logics Constructors

- Number restrictions on roles (*N* resp. *Q*)
  - simple: $\exists_{\geq 3}$has_child or $\exists_{\leq 5}$has_child
  - qualified: $\exists_{\geq 2}$has_child.male or $\exists_{\leq 1}$has_child.female
- Role hierarchies (*H*)
  - has_son ⊑ has_child, has_daughter ⊑ has_child
  - $\exists_{\geq 2}$has_son ⊓ $\exists_{\geq 2}$has_daughter ⊓ $\exists_{\leq 4}$has_child
- Transitive roles (R_+)
  - has_ancestors declared as transitive: ∀has_ancestors.human
  - has_parent ⊑ has_ancestors
- Inverse roles (*I*): has_parent ≡ has_child$^-$
- Terminological cycles: human ⊑ $\exists_{\geq 2}$has_parent.human
- General axioms
  - woman ⊓ ∃has_child.∃has_child.person ⊑ grandma

Tableau Methods

- How can we prove the satisfiability of a concept?
- Achieved by applying tableau methods
  - set of completion rules operating on constraint sets or tableaux
  - clash triggers
- Proof procedure
  - transform all concepts into negation normal form, e.g., $\neg(C \sqcap D) \rightarrow \neg C \sqcup \neg D$, $\neg\exists R.C \rightarrow \forall R.\neg C$
  - apply completion rules in arbitrary order as long as possible
  - application of rules
    - stops in case of a clash
    - terminates if no completion rule is applicable
  - satisfiable iff a clash-free tableau can be derived

Completion Rules for the Logic *ALC*

Clash trigger
$\{a{:}C, a{:}\neg C\} \subseteq \mathcal{A}$

Conjunction rule
if 1. $a{:}C \sqcap D \in \mathcal{A}$, and
2. $\{a{:}C, a{:}D\} \not\subseteq \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{a{:}C, a{:}D\}$

Disjunction rule
if 1. $a{:}C \sqcup D \in \mathcal{A}$, and
2. $\{a{:}C, a{:}D\} \cap \mathcal{A} = \varnothing$
then $\mathcal{A}' = \mathcal{A} \cup \{a{:}C\}$ **or**
$\mathcal{A}' = \mathcal{A} \cup \{a{:}D\}$

Role exists restriction rule
if 1. $a{:}\exists R.C \in \mathcal{A}$, and
2. $\neg\exists b \in \mathcal{O}{:}\ \{(a,b){:}R, b{:}C\} \subseteq \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{(a,b){:}R, b{:}C\}$
with b fresh in $\mathcal{A}$

Role value restriction rule
if 1. $a{:}\forall R.C \in \mathcal{A}$, and
2. $\exists b \in \mathcal{O}{:}\ (a,b){:}R \in \mathcal{A}$, and
3. $\{b{:}C\} \notin \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{b{:}C\}$

Proof for Concept Satisfiability

- Subsumes the concept woman the concept mother?
- Is the concept ¬woman ⊓ mother unsatisfiable?
- Application of completion rules
  - $\mathcal{A}_0$ = {a: (¬female⊔¬person) ⊓ female ⊓ person ⊓ ...} (*conjunction rule*)
  - $\mathcal{A}_1$ = {a:¬female⊔¬person, a:female, a:person, ...} (*disjunction rule*)
  - $\mathcal{A}_2$ = {a:¬female⊔¬person, a:female, a:person, ..., a:¬female}
  - ↯ (clash between a:female and a:¬female detected)
  - $\mathcal{A}_1$ = {a:¬female⊔¬person, a:female, a:person, ...} (*disjunction rule*)
  - $\mathcal{A}_3$ = {a:¬female⊔¬person, a:female, a:person, ..., a:¬person}
  - ↯ (clash between a:person and a:¬person detected)
- The concept ¬woman ⊓ mother is unsatisfiable
- The concept woman subsumes the concept mother

Reasoning with Description Logics

- RACER: Reasoner for ABoxes and Concept Expressions Renamed
- Based on sound and complete algorithms
- Worst case complexity for many description logics
  - PSpace, e.g., the logic *ALC*
  - ExpTime, e.g., the logic *ALC* with general axioms
  - NexpTime
    - the logic $ALCQHIR_{+}(D^{-})$ supported by RACER
    - the DAML+OIL logic
- Highly optimized reasoners required
  - average complexity usually much better
- RACER is still the only reasoner for ABoxes

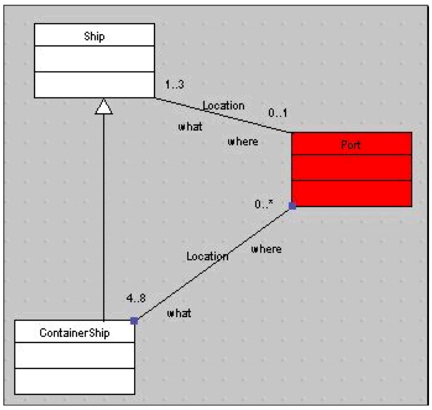
RACER System

- First system for $\mathcal{ALCQHI}_{R+}$ with ABoxes
  - sublogic of DAML+OIL
- Multiple TBoxes, multiple ABoxes
- Standalone server versions available for Linux and Windows (with Java interface)
- Newly added: concrete domains
  - represent constraints with linear inequations over the Reals
  - for instance: the relationship between the Celsius and Fahrenheit scales
- Almost finished
  - XML / RDF / DAML+OIL interface
- Standardized interface (API) is being devolped

Selected Optimization Techniques

- State of the art optimization techniques employed
- Novel optimization techniques for
  - SAT reasoning
    - dependency-directed backtracking
    - semantic branching
    - caching
    - process qualified number restrictions with Simplex procedure
  - TBox reasoning
    - transformation of general axioms
    - classification order / clustering of nodes
    - fast test for non-subsumption: sound but incomplete
  - ABox reasoning
    - graph transformation
    - fast test for non-subsumption
    - data-flow techniques for realization
    - dependency-driven divide-and-conquer for instance checks

Application: UML Verification



XML representation
created by UML
Editor or Tool

Agent

$\mathrm{Ship} \sqsubseteq \exists_{\leq 1}\mathrm{what\_location\_where.Port}$

$\mathrm{ContainerShip} \sqsubseteq \mathrm{Ship}$

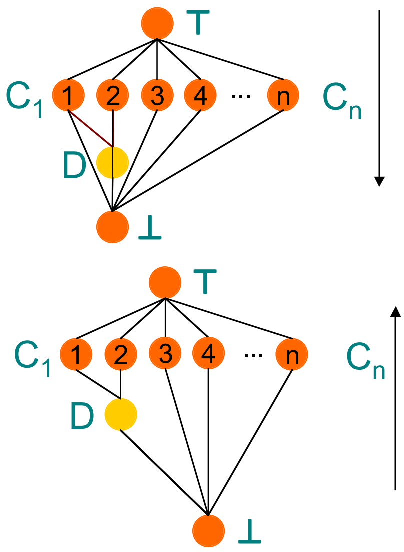
$\mathrm{Port} \sqsubseteq \exists_{\geq 1}\mathrm{what\_location\_where}^{-}.\mathrm{Ship} \sqcap$
$\exists_{\leq 3}\mathrm{what\_location\_where}^{-}.\mathrm{Ship} \sqcap$
$\exists_{\geq 4}\mathrm{what\_location\_where}^{-}.\mathrm{ContainerShip} \sqcap$
$\exists_{\leq 8}\mathrm{what\_location\_where}^{-}.\mathrm{ContainerShip}$

Application: Ontology Engineering

- UMLS thesaurus (Unified Medical Language System)
- Transformation into logic *ALCNH*
  - TBox with cycles, role hierarchy, and simple number restrictions
- UMLS knowledge bases
  - 200,000 concept names, 80,000 role names
- Optimization of TBox classification
  - topological sorting
    - achieving smart ordering for classification of concept names
  - dealing with domain and range restrictions of roles
    - transformation of special kind of general axioms
  - clustering of nodes in the taxonomy
  - speed up from several days to ~10 hours
  - new processors: ~3 hours

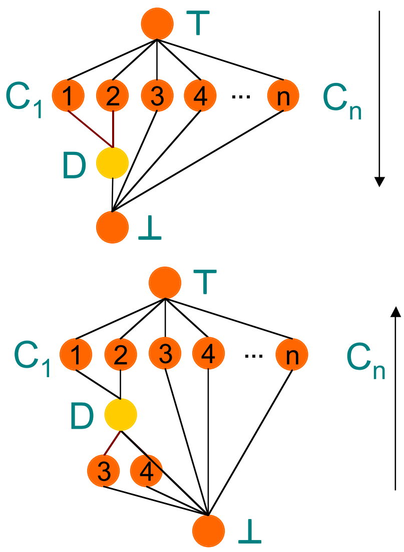
TBox Classification: Inserting a Concept

- Insert new concept D into existing taxonomy w.r.t subsumption relationship
- 1. Top-search phase
  - traverse from top
  - determine parents of D
    - C_1 and C_2
  - $\mathrm{SAT}(\neg C_1 \sqcap D)$, ..., $\mathrm{SAT}(\neg C_n \sqcap D)$
- 2. Bottom-search phase
  - traverse from bottom
  - determine children of D
    - C_3 and C_4
  - $\mathrm{SAT}(C_1 \sqcap \neg D)$, ..., $\mathrm{SAT}(C_n \sqcap \neg D)$

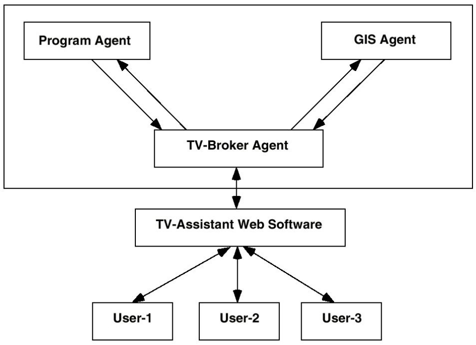


TBox Classification: Inserting a Concept

- Insert new concept D into existing taxonomy w.r.t subsumption relationship
- 1. Top-search phase
  - traverse from top
  - determine parents of D
    - C_1 and C_2
  - $\mathrm{SAT}(\neg C_1 \sqcap D)$, ..., $\mathrm{SAT}(\neg C_n \sqcap D)$
- 2. Bottom-search phase
  - traverse from bottom
  - determine children of D
    - C_3 and C_4
  - $\mathrm{SAT}(C_1 \sqcap \neg D)$, ..., $\mathrm{SAT}(C_n \sqcap \neg D)$



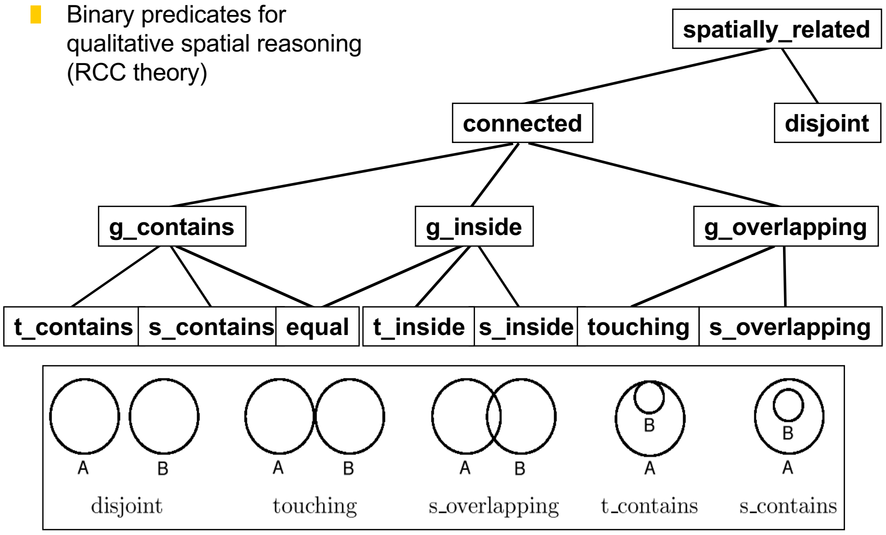
Application: Distributed Agents



- Specialized reasoner for TV programs
- Specialized reasoner for data from Geographical Information Systems (GIS)
- Broker agent as mediator

Spatial Reasoning with Description Logics

- Binary predicates for qualitative spatial reasoning (RCC theory)

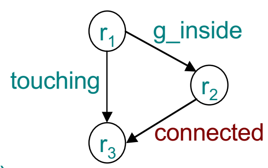


Example: Paradise Cottage (1)

- A paradise cottage
  - it is a cottage
  - suitable for fishing
    - located in the immediate vicinity of a river
    - simplification: estate touches a river
  - located in a mosquito-free forest
    - simplification: a mosquito-free forest does not overlap with a river
- Specification with *ALCRP(D)*
  - fishing_cottage ≡ cottage ⊓ ∃is_touching.river
  - mosquito_free_forest ≡ forest ⊓ ∀is_connected.¬river
  - paradise_cottage ≡ fishing_cottage ⊓ ∃is_g_inside.forest ⊓ ∀is_g_inside.mosquito_free_forest
- What is your opinion: dream or reality?

Example: Paradise Cottage (2)

- A situation, where a region r_1 (cottage) is located inside another region r_2 (forest) and the region r_1 touches a third region r_3 (river), implies that r_2 must be connected with r_3



- g_inside(r_1,r_2) ∧ touching(r_1,r_3) ⇒ connected(r_2, r_3)
- The concept paradise_cottage is unfortunately unsatisfiable due to induced spatial constraints
  - a mosquito-free forest is not allowed to be spatially connected with a river
  - only detectable with the logic *ALCRP(D)*

Future Research (1)

- Integration of spatial reasoning into description logics
  - bioinformatics
  - (semantics of) spatial queries
  - geographical information systems
- Extend support for very expressive description logics
  - integration of individuals into concept descriptions
  - concrete domains
    - non-linear, multivariate systems of inequations
- Development of new optimization techniques
  - inverse roles
  - individuals in concept descriptions
  - complex (and very large) knowledge bases

Future Research (2)

- Support of Semantic Web
- Support for databases
  - schemas
  - query subsumption
  - database integration
- Development of (industrial) applications
  - geographical information systems
  - telecommunication systems / mobile systems
  - computer vision
  - matchmaking of services
  - natural language understanding
  - ...

Other Areas of Interest

- Diagrammatic reasoning
- Visual languages / notations
- Knowledge management / engineering
- Software engineering (for AI)
- Object-oriented design
- Programming languages / paradigms
- ...