Racer - An Inference Engine for the Semantic Web

Volker Haarslev
Concordia University
Department of Computer Science and Software Enineering
http://www.cse.concordia.ca/~haarslev/

Collaboration with:
Ralf Möller, Hamburg University of Science and Technology

Basic Web Technology (1)

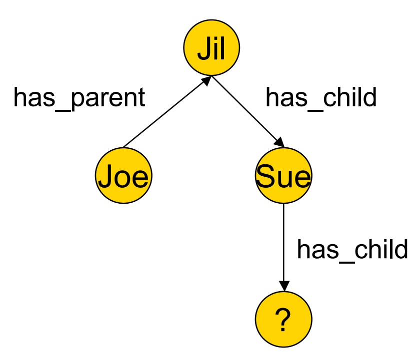
- Uniform Resource Identifier (URI)
  - foundation of the Web
  - identify items on the Web
  - uniform resource locator (URL): special form of URI
- Extensible Markup Language (XML)
  - send documents across the Web
  - allows anyone to design own document formats (syntax)
  - can include markup to enhance meaning of document's content
  - machine readable

Basic Web Technology (2)

- Resource Description Framework (RDF)
  - make machine-processable statements
  - triple of URIs: subject, predicate, object
  - intended for information from databases
- Ontology Web Language (OWL)
  - based on
    - RDF
    - description logics (as part of automated reasoning)
    - syntax is XML
  - knowledge representation in the web

What is Knowledge Representation?

- How would one argue that a person is an uncle?
- We might describe family relationships by a relation
  - has_parent and its inverse has_child
- Now can can define an uncle
  - a person (Joe) is an uncle if and only if
    - he is male
    - he has a parent (Jil) and this parent has a second child
    - this child (Sue) is itself a parent
  - Sue is called a sibling of Joe and vice versa



Volker Haarslev
Department of Computer Science and Software Engineering
Concordia University, Montreal

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
- Ontology Web Language (OWL)
  - 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)

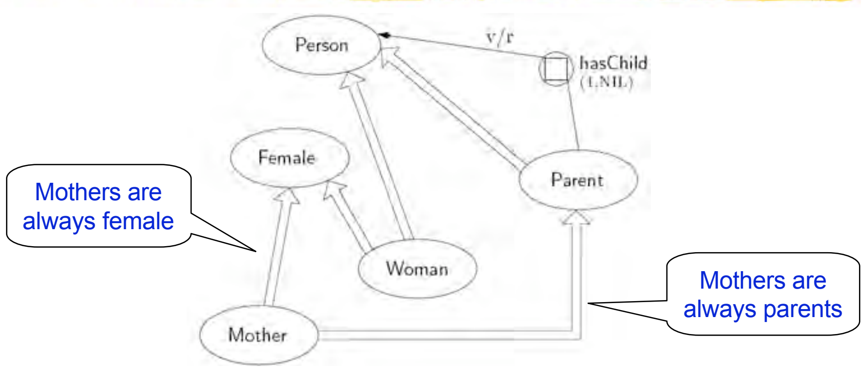
OWL Variants

- Three variants
  - OWL Full represents union of OWL syntax and RDF
    - gives you unrestricted expressive power
  - OWL DL restricted to decidable fragment of first-order logic
    - syntactic variant of well-known description logic
  - OWL Lite restricted subset of OWL DL
    - “Easier to implement”

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 OWL)
- Very tight coupling between theory and practice

Origins of Description Logics

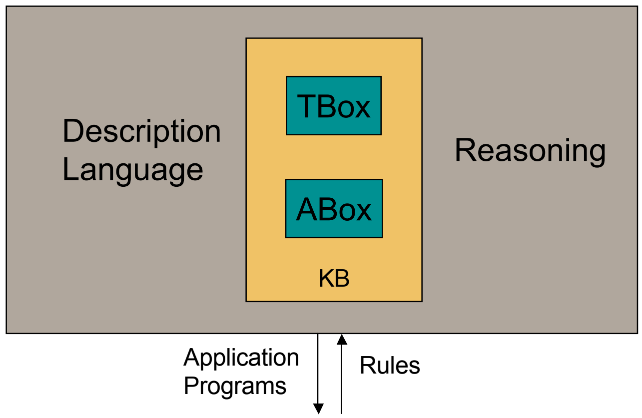


- Knowledge concerning persons, parents, etc.
- Described as semantic network
- Semantic networks without a sematics

Volker Haarslev
Department of Computer Science and Software Engineering
Concordia University, Montreal

Description Logic System

Architecture of a Description Logic System



Description Languages: $\mathcal{AL}$

- Translation to first-order predicate logic 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
T	$T^{\mathcal{I}} = \Delta^{\mathcal{I}}$
$\bot$	$\bot^{\mathcal{I}} = \varnothing$
$\neg A$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y: (x,y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
$\exists R.T$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in R^{\mathcal{I}}\}$
R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, R is a role name

- person ⊓ female
- person ⊓ ∃has_child.T
- person ⊓ ¬female
- person ⊓ ∀has_child.⊥

Volker Haarslev
Department of Computer Science and Software Engineering
Concordia University, Montreal

More $\mathcal{AL}$ Family Members

- Disjunction ($\mathcal{U}$):
  $C \sqcup D$ $C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- Full existential quantification ($\mathcal{E}$):
  $\exists R.C$ $\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
- Number restrictions ($\mathcal{N}$):
  $\exists_{\geq n}R$ $\{x \in \Delta^{\mathcal{I}} \mid \|\{y \mid (x,y) \in R^{\mathcal{I}}\}\| \geq n\}$
  $\exists_{\leq n}R$ $\{x \in \Delta^{\mathcal{I}} \mid \|\{y \mid (x,y) \in R^{\mathcal{I}}\}\| \leq n\}$
- Full negation ($\mathcal{C}$):
  $\neg C$ $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- person ⊓ ($\exists_{\leq 1}$has_child ⊔ ($\exists_{\geq 3}$has_child ⊓ ∃has_child.female))

DLs as Fragments of Predicate Logic

- Any concept D as unary predicate with 1 free variable
- Any role R as primitive binary predicate
- ∃R.C corresponds to
  $\exists y.\ R(x,y) \wedge C(y)$
- ∀R.C corresponds to
  $\forall y.\ R(x,y) \Rightarrow C(y)$
- $\exists_{\geq n}R$ corresponds to
  $\exists y_1,\ldots,y_n.\ R(x,y_1) \wedge \ldots \wedge R(x,y_n) \wedge \forall i<j.\ y_i \neq y_j$
- $\exists_{\leq n}R$ corresponds to
  $\forall y_1,\ldots,y_{n+1}.\ R(x,y_1) \wedge \ldots \wedge R(x,y_{n+1}) \Rightarrow \exists i<j.\ y_i = y_j$
- Last two examples demonstrate advantage of variable-free syntax

Inference Services

- Consistency of class description
  - catch design errors
  - example: vegetarian eats meat
- Subsumption between classes
  - example: a mother is always a parent
- Taxonomy of class names (classification)
  - ordered by subsumption relationship
  - from very general to very specific
- Consistency of individual descriptions
  - Is the knowledge specified for an individual joe consistent with other known individuals and classes
  - joe (vegetarian) makes a reservation for a restaurant that offers only meals containing meat
- Find classes that match known instances
  - if susy is female and has a child, she is an instance of mother

OWL Class Constructors

Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer
complementOf	$\neg C$	¬Male
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}
allValuesFrom	$\forall P.C$	∀hasChild.Doctor
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer
maxCardinality	$\leqslant n P$	⩽1hasChild
minCardinality	$\geqslant n P$	⩾2hasChild

- XMLS datatypes as well as classes in ∀P.C and ∃P.C
  - E.g., ∀hasAge.nonNegativeInteger
- Arbitrarily complex nesting of constructors
  - E.g., Person ⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)

OWL Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	$\text{Human} \sqsubseteq \text{Animal} \sqcap \text{Biped}$
equivalentClass	$C_1 \equiv C_2$	$\text{Man} \equiv \text{Human} \sqcap \text{Male}$
disjointWith	$C_1 \sqsubseteq \neg C_2$	$\text{Male} \sqsubseteq \neg\text{Female}$
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	$\{\text{President\_Bush}\} \equiv \{\text{G\_W\_Bush}\}$
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	$\{\text{john}\} \sqsubseteq \neg\{\text{peter}\}$
subPropertyOf	$P_1 \sqsubseteq P_2$	$\text{hasDaughter} \sqsubseteq \text{hasChild}$
equivalentProperty	$P_1 \equiv P_2$	$\text{cost} \equiv \text{price}$
inverseOf	$P_1 \equiv P_2^-$	$\text{hasChild} \equiv \text{hasParent}^-$
transitiveProperty	$P^+ \sqsubseteq P$	$\text{ancestor}^+ \sqsubseteq \text{ancestor}$
functionalProperty	$\top \sqsubseteq \leqslant 1P$	$\top \sqsubseteq \leqslant 1\text{hasMother}$
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1P^-$	$\top \sqsubseteq \leqslant 1\text{hasSSN}^-$

- Axioms (mostly) reducible to inclusion ($\sqsubseteq$)
  - $C \equiv D$ iff both $C \sqsubseteq D$ and $D \sqsubseteq C$

OWL Examples: Simple Named Classes

- Domain of wines
- `<owl:Class rdf:ID="Winery"/>`
- `<owl:Class rdf:ID="Region"/>`
- `<owl:Class rdf:ID="ConsumableThing"/>`

```
<owl:Class rdf:ID="PotableLiquid">
<rdfs:subClassOf rdf:resource="#ConsumableThing"/>
...
</owl:Class>
```

Volker Haarslev
Department of Computer Science and Software Engineering
Concordia University, Montreal

Individuals

- We declare an individual named CentralCoastRegion as an instance of class Region

```
<Region rdf:ID="CentralCoastRegion"/>
```

Class Name

Individual Name

Import of Ontologies

- There exists an ontology about food containing class grape

```
<owl:Class rdf:ID="Grape">
  ...
</owl:Class>
```

- Class WineGrape is declared as subclass of class grape imported from the food ontology

```
<owl:Class rdf:ID="WineGrape">
  <rdfs:subClassOf rdf:resource="&food;Grape"/>
</owl:Class>
```

Volker Haarslev
Department of Computer Science and Software Engineering
Concordia University, Montreal

Object Properties

- We define an object property madeFromGrape
  - its domain is Wine
  - its range is WineGrape

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>
```

- Individual LindemansBin65Chardonnay is related via property madeFromGrape to indiviual ChardonnayGrape
  - Inference: instance of class Wine

```
<owl:Thing rdf:ID="LindemansBin65Chardonnay">
  <madeFromGrape rdf:resource="#ChardonnayGrape"/>
</owl:Thing>
```

Complex Classes

- A more complete declaration of class Wine

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#madeFromGrape"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
      1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

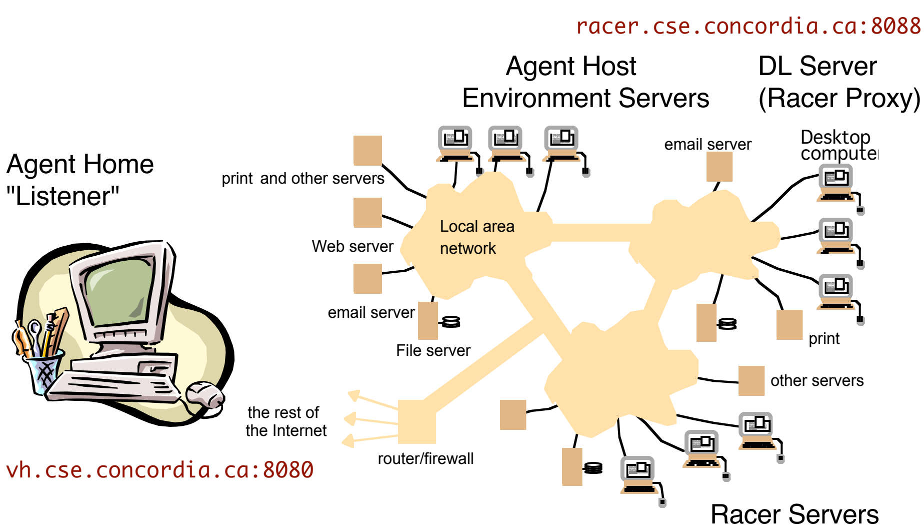
Anonymous class for things with at least one madeFromGrape property

Volker Haarslev
Department of Computer Science and Software Engineering
Concordia University, Montreal

Racer: Reasoning with OWL

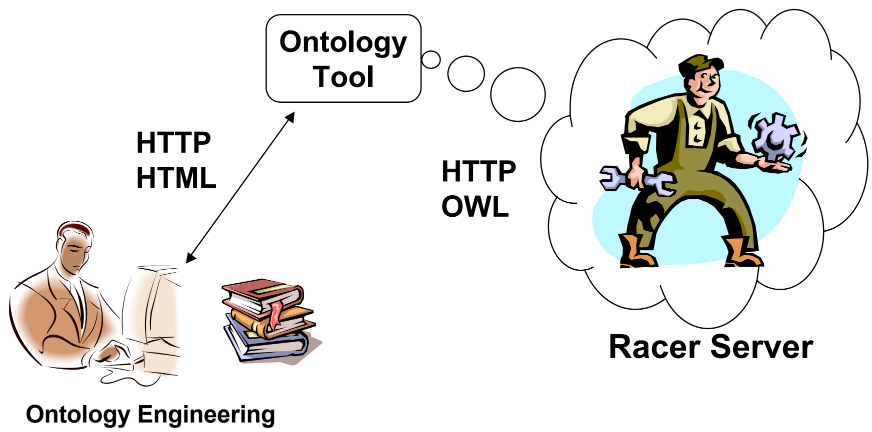
- Based on sound and complete algorithms
- Worst case complexity
  - high for OWL DL
  - reasonable for OWL Lite
- Highly optimized reasoners required
  - average complexity usually ok
- Supports multiple ontologies
- Standalone server versions available for Linux and Windows (with Java/C++ API)
- Network based APIs supported (HTTP, TCP/IP)
- RACER is still the only true reasoner for individuals
- http://www.cse.concordia.ca/~haarslev/racer/

Agent Scenario



Volker Haarslev
Department of Computer Science and Software Engineering
Concordia University, Montreal

Racer as OWL Reasoning Agent

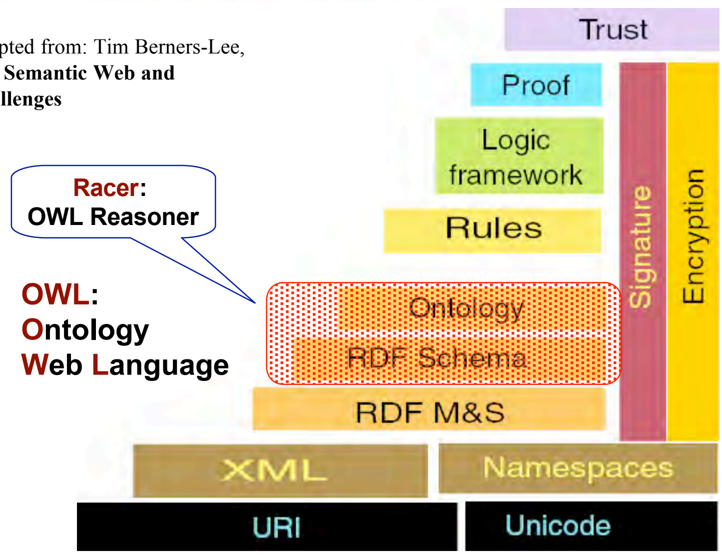


Application: Ontology Engineering

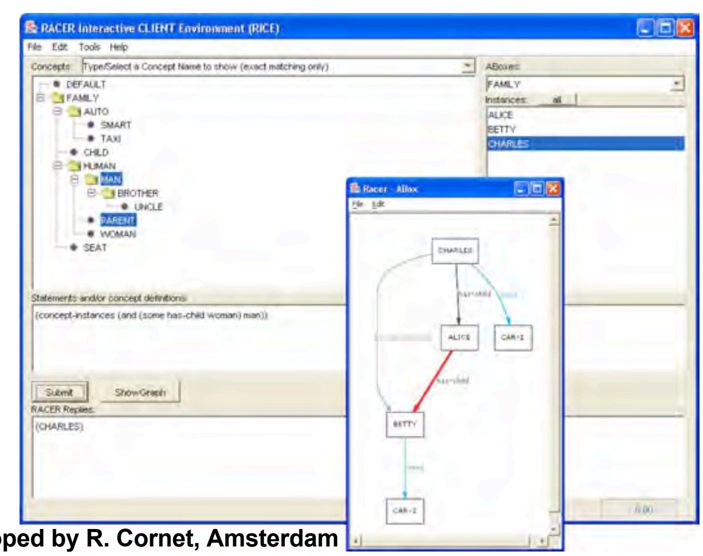
- UMLS thesaurus (Unified Medical Language System)
- Transformation into description logic
- UMLS knowledge bases
  - 200,000 class names, 80,000 property names
- Optimization of ontology classification
  - topological sorting
    - achieving smart ordering for classification of class names
  - dealing with domain and range restrictions of properties
    - transformation of special kind of general axioms
  - clustering of nodes in the taxonomy
  - speed up from several days to ~10 hours
  - more optimizations and new processors: below 3 hours of CPU time

Semantic Tower

Adapted from: Tim Berners-Lee, **The Semantic Web and Challenges**

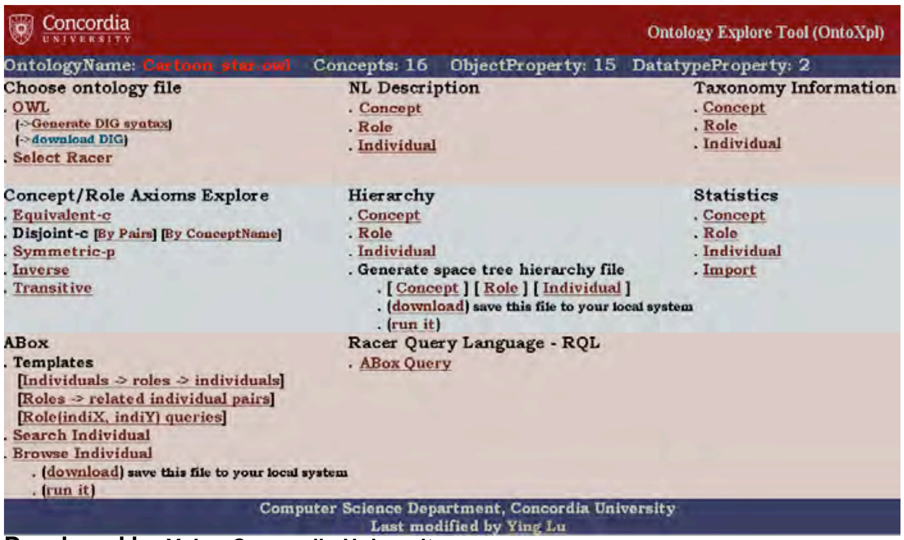


RICE: Racer Interactive Client Environment



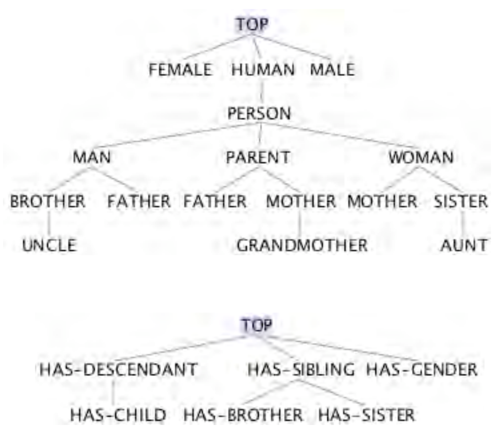
Developed by R. Cornet, Amsterdam

OntoXpl: OWL Ontology Explorer



Developed by Y. Lu, Concordia University

Ontology about Family Relationships



Information about Class "Person"

Ancestor:	• HUMAN • TOP
Parents:	• HUMAN
Children:	• MAN • PARENT • WOMAN
Descendant:	• AUNT • BOTTOM • BROTHER • FATHER • GRANDMOTHER • MAN • MOTHER • PARENT • SISTER • UNCLE • WOMAN
Roles used by this concept:	• HAS-GENDER
Instances of this concept:	• ALICE • BETTY • CHARLES • DORIS • EVE

OWL View of Class "Person"

```
(rdfs:subClassOf)
        (owl:Class)
                (owl:intersectionOf rdf:parseType="Collection")
                        (owl:Class rdf:about="HUMAN")
                        (/owl:Class)
                        (owl:Restriction)
                                (owl:onProperty rdf:resource="HAS-GENDER")
                                (/owl:onProperty)
                                (owl:someValuesFrom)
                                        (owl:Class)
                                                (owl:unionOf rdf:parseType="Collection")
                                                        (owl:Class rdf:about="FEMALE")
                                                        (/owl:Class)
                                                        (owl:Class rdf:about="MALE")
                                                        (/owl:Class)
                                                (/owl:unionOf)
                                        (/owl:Class)
                                (/owl:someValuesFrom)
                        (/owl:Restriction)
                (/owl:intersectionOf)
        (/owl:Class)
(/rdfs:subClassOf)
```

OWL View of Class "Person"

```
It is the anonymous subclass of
   A concept HUMAN
   and
   it has a filler in the role HAS-GENDER
   at least one (or more than one) of its instances is(are):
   A concept FEMALE
         or
   A concept MALE
```

nRQL: New Racer Query Language

- Searching for complex role-filler graph structures in an ABox
  - Looking for a "Disney mouse", who has nieces, and is a friend of Mickey

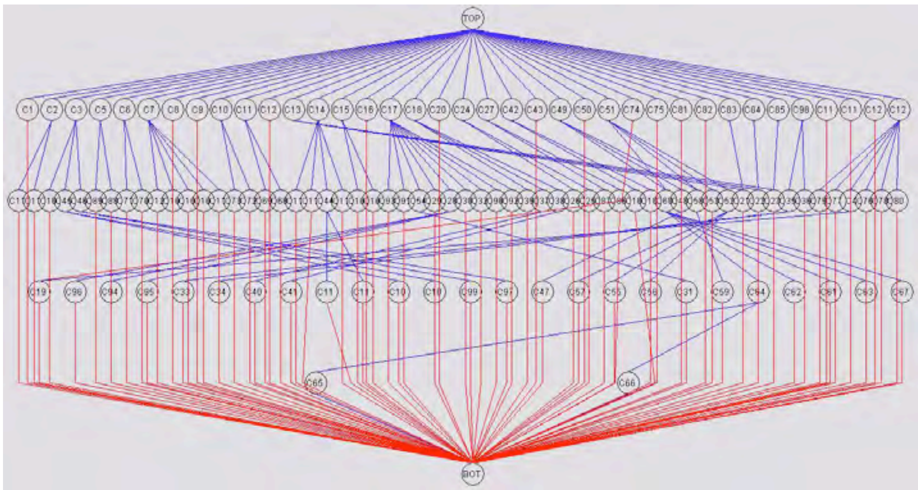
Please input the Racer Query Language here: [*manual*]

```
(retrieve (?disneyMouse ?niece)
   (and
      (?disneyMouse |http://a.com/ontology#Disney_mouse|)
      (?niece |http://a.com/ontology#Disney_mouse|)
      (?niece ?disneyMouse |http://a.com/ontology#Is_niece_of|)
      (?disneyMouse |http://a.com/ontology#Mickey| |http://a.com/ontology#Is_friend_of|)
   )
)
```

Query Result is:

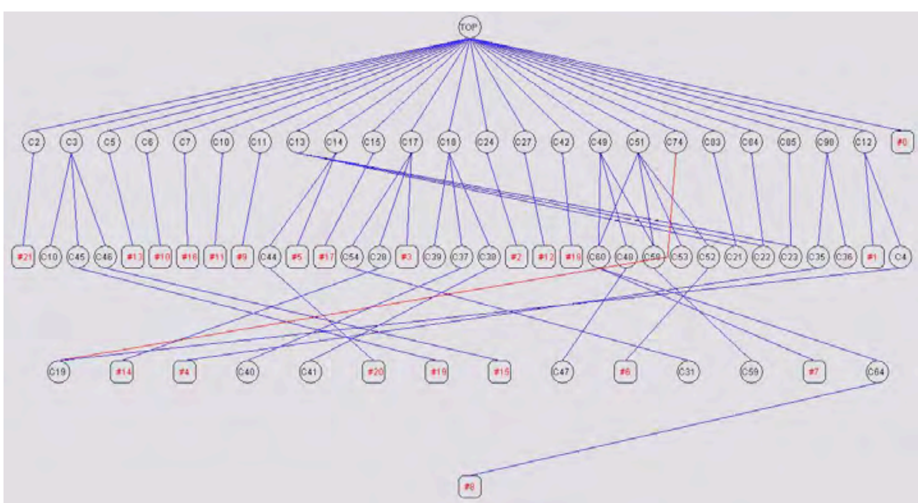
(((?DISNEYMOUSE Minnie) (?NIECE Millicent)) ((?DISNEYMOUSE Minnie) (?NIECE Melody)))

2D Visualization of Subsumption Hierarchy (1)

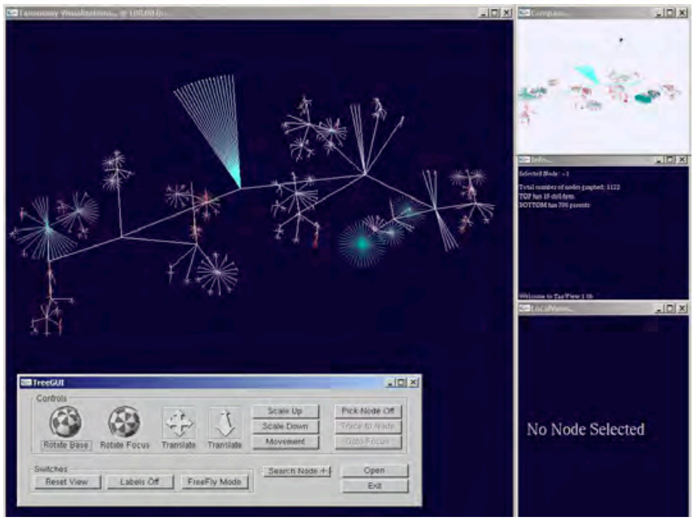


Developed by A. Zarrad, 2004

2D Visualization of Subsumption Hierarchy (2)

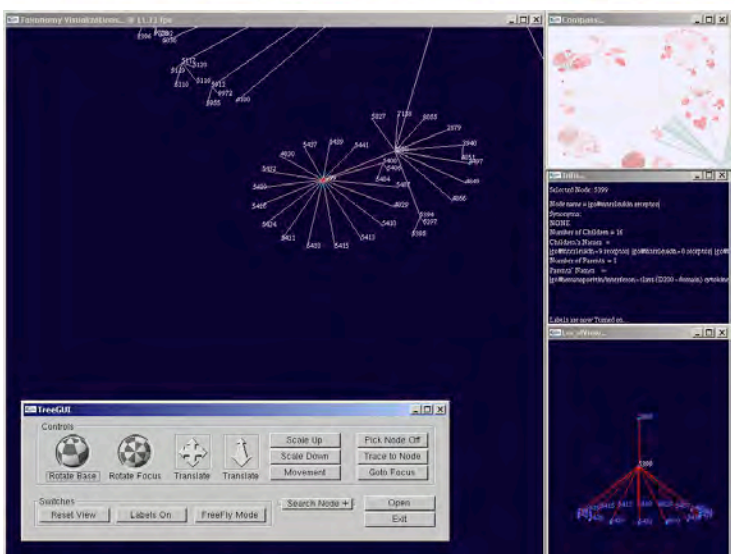


3D Visualization of Subsumption Hierarchy (1)

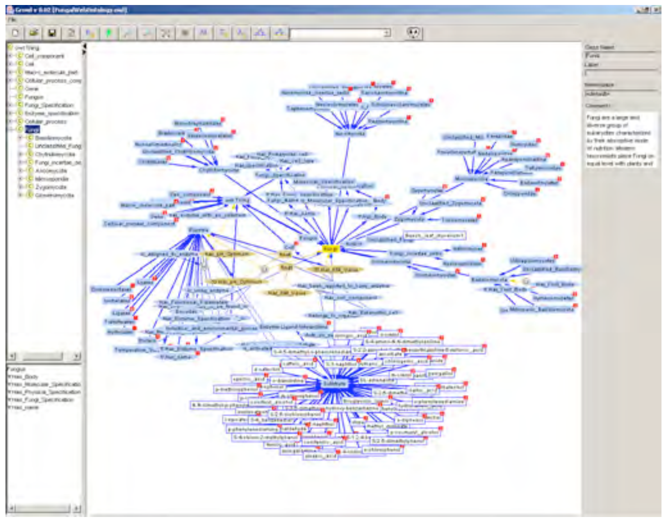


Developed by P. Eid, 2005

3D Visualization of Subsumption Hierarchy (2)

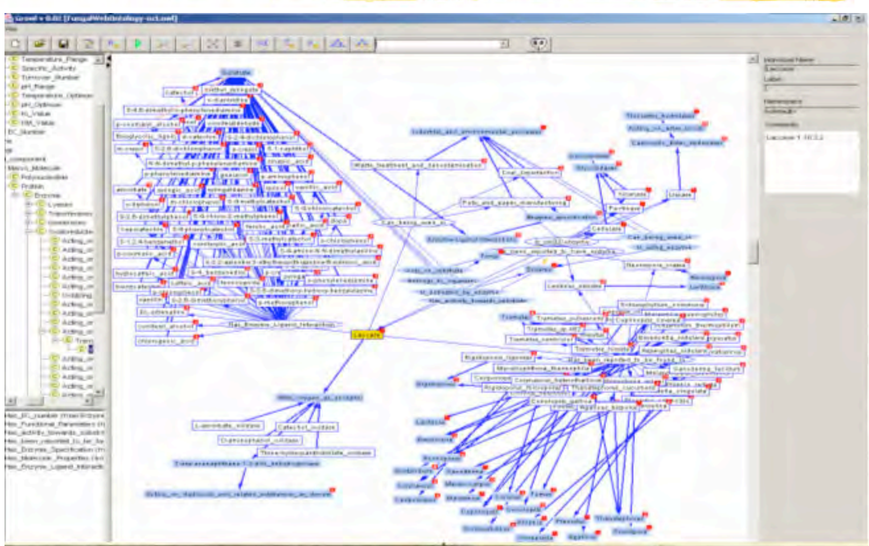


Genomics: FungalWeb Ontology (1)



Developed by A. Shaban-Nejad

Genomics: FungalWeb Ontology (2)



Inference Services Based on Satisfiability

- All concept inference services can be reduced to concept satisfiability
- We assume service sat(C, $\mathcal{T}$), C a concept, $\mathcal{T}$ a TBox
- subsumes(C, D, $\mathcal{T}$) $\equiv \neg$ sat($\neg C \sqcap D$, $\mathcal{T}$)
  - $C \sqsupseteq D$ holds $\leftrightarrow$ $\neg(C \sqcup \neg D)$ unsatisfiable $\leftrightarrow$ $\neg C \sqcap D$ unsatisfiable
- equivalence(C, D, $\mathcal{T}$) $\equiv$ subsumes(C, D, $\mathcal{T}$) $\wedge$
  subsumes(D, C, $\mathcal{T}$)
- disjoint(C, D, $\mathcal{T}$) $\equiv \neg$ sat($C \sqcap D$, $\mathcal{T}$)

World Description or ABox

- 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
  - $a^{\mathcal{I}} \neq b^{\mathcal{I}}$

ABox Inference Services (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 $\mathcal{T}$

ABox Inference Services (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?

Completion Rules for the Logic *ALC*

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

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. $\{a:C, a:D\} \cap \mathcal{A} = \emptyset$
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 $\mathcal{A}' = \mathcal{A} \cup \{b:C\}$

Clash detection

- After each rule application an ABox $\mathcal{A}$ is checked for a clash involving concept names
- No other clashes can occur
- Can be generalized to arbitrary concept expressions
  - A is not necessarily only a name
- Rule expansion stops if a clash is detected

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

Volker Haarslev
Department of Computer Science and Software Engineering
Concordia University, Montreal

Conjunction rule

- Decompose a binary concept conjunction into two separate parts that are added to the ABox
- Meaning of conditions
  - case 1 controls applicability
  - case 2 prevents cyclic rule application

Conjunction rule

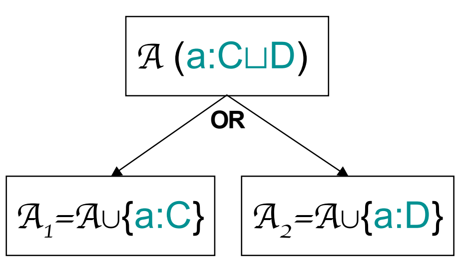
if 1. $a{:}C \sqcap D \in \mathcal{A}$, and

2. $\{a{:}C, a{:}D\} \not\subseteq \mathcal{A}$

then $\mathcal{A}' = \mathcal{A} \cup \{a{:}C, a{:}D\}$

Disjunction rule (non-deterministic)

- Non-deterministically add any of the disjuncts to the ABox
- Two alternative ABoxes are possibly explored



Disjunction rule

if 1. $a{:}C \sqcup D \in \mathcal{A}$, and

2. $\{a{:}C, a{:}D\} \cap \mathcal{A} = \emptyset$

then $\mathcal{A}' = \mathcal{A} \cup \{a{:}C\}$ **or**

$\mathcal{A}' = \mathcal{A} \cup \{a{:}D\}$

- Clashes eliminate branches in the OR tree

Volker Haarslev
Department of Computer Science and Software Engineering
Concordia University, Montreal

Maintain universal role restrictions

- Propagate role value restriction (C) to all applicable role (R) successors
- Only applicable if role successors can be found

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 $\mathcal{A}' = \mathcal{A} \cup \{b{:}C\}$

Create role successors

- Expand existential restrictions
  - create an appropriate role (R) successor (new individual)
  - assert the qualification (C) to the new successor
- $\mathcal{O}$ is the set of all possible individual names
- New individual (b) is considered as anonymous
  - not visible in original ABox
  - only needed for proof
  - part of a model
- Only rule that creates new individuals in an ABox

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 $\mathcal{A}$

Completion Rules for the Logic *ALC*

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

Conjunction rule
if 1. a:C⊓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⊔D $\in \mathcal{A}$, and
2. {a:C, a:D} $\cap\, \mathcal{A} = \emptyset$
then $\mathcal{A}' = \mathcal{A} \cup$ {a:C} **or**
$\mathcal{A}' = \mathcal{A} \cup$ {a:D}

Role exists restriction rule
if 1. a:∃R.C $\in \mathcal{A}$, and
2. ¬∃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:∀R.C $\in \mathcal{A}$, and
2. ∃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

Volker Haarslev
Department of Computer Science and Software Engineering
Concordia University, Montreal

Completion Rules for the Logic *ALC*

Clash trigger
{a:C, a:¬C} ⊆ $\mathcal{A}$

Conjunction rule
if 1. a:C⊓D ∈ $\mathcal{A}$, and
2. {a:C, a:D} ⊈ $\mathcal{A}$
then $\mathcal{A}$' = $\mathcal{A}$ ∪ {a:C, a:D}

Disjunction rule
if 1. a:C⊔D ∈ $\mathcal{A}$, and
2. {a:C, a:D} ∩ $\mathcal{A}$ = ∅
then $\mathcal{A}$' = $\mathcal{A}$ ∪ {a:C} **or**
$\mathcal{A}$' = $\mathcal{A}$ ∪ {a:D}

Role exists restriction rule
if 1. a:∃R.C ∈ $\mathcal{A}$, and
2. ¬∃b ∈ $\mathcal{O}$: {(a,b):R, b:C} ⊆ $\mathcal{A}$
then $\mathcal{A}$' = $\mathcal{A}$ ∪ {(a,b):R, b:C}
with b fresh in $\mathcal{A}$

Role value restriction rule
if 1. a:∀R.C ∈ $\mathcal{A}$, and
2. ∃b ∈ $\mathcal{O}$: (a,b):R ∈ $\mathcal{A}$, and
3. {b:C} ∉ $\mathcal{A}$
then $\mathcal{A}$' = $\mathcal{A}$ ∪ {b:C}

Proof for Concept Satisfiability

- Subsumes the concept ∃R.(A ⊓ B) the concept ∃R.A ⊓ ∃R.B ?
- Is the concept ¬∃R.(A ⊓ B) ⊓ ∃R.A ⊓ ∃R.B unsatisfiable?
- Application of completion rules
  - $\mathcal{A}_0$ = {a:∀R.(¬A⊔¬B) ⊓ ∃R.A ⊓ ∃R.B} (*conjunction rule*)
  - $\mathcal{A}_1$ = {a:∀R.(¬A⊔¬B), a:∃R.A, a:∃R.B} (*role exists restriction rule*)
  - $\mathcal{A}_2$ = {(a,x):R, x:A, (a,y):R, y:B, a:∀R.(¬A⊔¬B), ...} (*role value restriction rule*)
  - $\mathcal{A}_3$ = {x:¬A⊔¬B, y:¬A⊔¬B, (a,x):R, x:A, (a,y):R, y:B, ...} (*disjunction rule*)
  - $\mathcal{A}_4$ = {x:¬A, x:¬A⊔¬B, y:¬A⊔¬B, (a,x):R, x:A, (a,y):R, y:B, ...}
  - ↯ (clash between x:¬A and x:A detected)
  - $\mathcal{A}_3$ = {x:¬A⊔¬B, y:¬A⊔¬B, (a,x):R, x:A, (a,y):R, y:B, ...} (*disjunction rule*)
  - $\mathcal{A}_5$ = {x:¬B, x:¬A⊔¬B, y:¬A⊔¬B, (a,x):R, x:A, (a,y):R, y:B, ...} (*disjunction rule*)
  - $\mathcal{A}_6$ = {y:¬A, x:¬B, x:¬A⊔¬B, y:¬A⊔¬B, (a,x):R, x:A, (a,y):R, y:B, ...}
- The concept ¬∃R.(A ⊓ B) ⊓ ∃R.A ⊓ ∃R.B is satisfiable
- The concept ∃R.(A ⊓ B) does not subsume the concept ∃R.A ⊓ ∃R.B

Adding axioms from TBox

- Transform all axioms in TBox into normal form
  - $C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n$ gives
  - $\top \sqsubseteq \neg C_1 \sqcup D_1, \ldots, \top \sqsubseteq \neg C_n \sqcup D_n$
- Combine all normalized axioms into one axiom
  - $\top \sqsubseteq (\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$
- For each new individual a add $a{:}(\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$

Simple (Trigger) Rules

- Rules may have the form $C \Rightarrow D$
  - available in the Classic system
- Operational semantics
  - forward chaining of rules
  - if a:C holds, a:D is added
- Observe the difference to axioms
  - $C \sqsubseteq D$ implies the contrapositive $\neg D \sqsubseteq \neg C$
  - this is not the case for rules
    - if a:¬D holds, a:¬C is **NOT** added

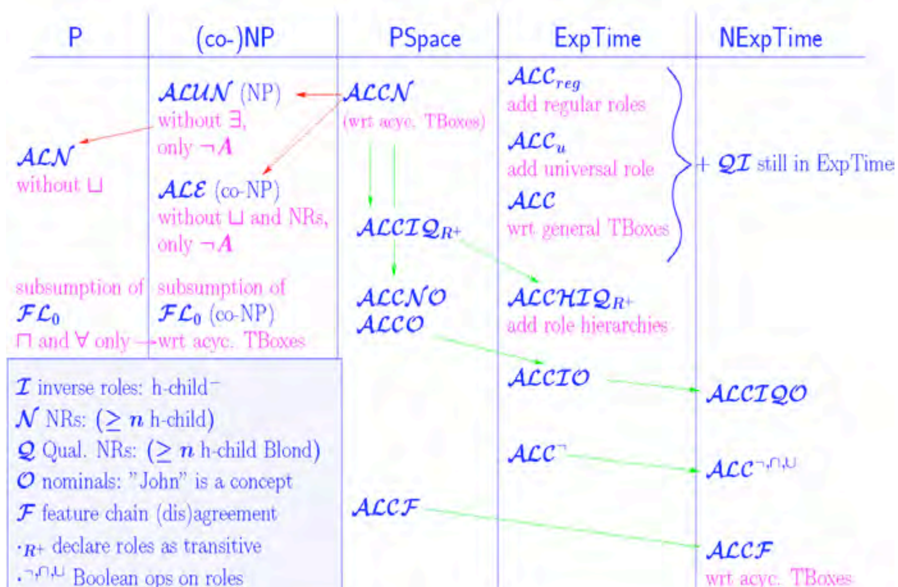
Two Reasoning views

- Traditional view from knowledge engineering
  - defined concept express domain knowledge
  - primitive concepts express only necessary conditions
  - axioms ensure global consistency criteria
  - inferences services
    - taxonomy
    - unsatisfiable concepts
- Theorem prover
  - domain knowledge is expressed as a set of arbitrary axioms
  - inference services
    - taxonomy gives no interesting information
    - unsatisfiable concepts
    - is a hypothesis implied by the set of axioms

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
  - (N)ExpTime
    - the logic $ALCQHIR_{+}(D^{-})$ supported by RACER
    - the OWL logic (OWL DL)
- Highly optimized reasoners required
  - average complexity usually much better
- RACER is still the only optimized reasoner for ABoxes

Complexity of Concept Consistency



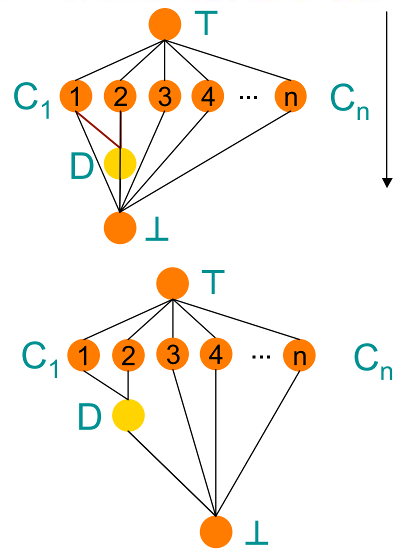
From: ***IJCAR Tutorial on Description Logics***, Ian Horrocks, Ulrike Sattler

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

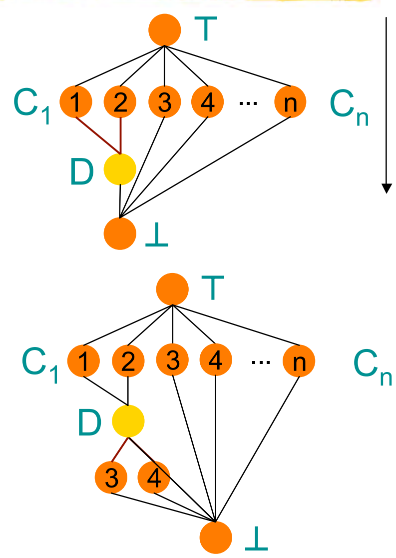
TBox Classification: Inserting a Concept

- 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
  - SAT(¬C_1⊓D), ..., SAT(¬C_n⊓D)
- 2. Bottom-search phase
  - traverse from bottom
  - determine children of D
    - C_3 and C_4
  - SAT(C_1⊓¬D), ..., SAT(C_n⊓¬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
  - SAT(¬C_1⊓D), ..., SAT(¬C_n⊓D)
- 2. Bottom-search phase
  - traverse from bottom
  - determine children of D
    - C_3 and C_4
  - SAT(C_1⊓¬D), ..., SAT(C_n⊓¬D)



Volker Haarslev
Department of Computer Science and Software Engineering
Concordia University, Montreal

Available Specifications: Primers

- RDF Primer
  - URI: http://www.w3.org/TR/rdf-primer/
- OWL Guide
  - URI: http://www.w3.org/TR/owl-guide/
- RDF Test Cases
  - URI: http://www.w3.org/TR/rdf-testcases/
- OWL Test Cases
  - URI: http://www.w3.org/TR/owl-test/