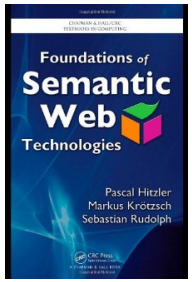
Semantic Web
5 - Reasoning, Logic and Rules

GEIST Research Group
http://geist.agh.edu.pl

AGH University of Science and Technology, POLAND

Using slides according to license from:

- P. Hitzler – "Knowledge Representation for the Semantic Web" *course based on*
- P. Hitzler, M. Krötzsch, S. Rudolph – Foundations of Semantic Web Technologies
- Ontology Modeling Languages course at ESSLLI 2009 in Bordeaux.



Outline

Outline

Why Rules?

OWL may not suffice for all applications

- There are statements that cannot be expressed in OWL (cf. Lecture 4)
- Modeling constructs of OWL not always adequate or most desirable
- First-order logic in general may be insufficient (e.g. if non-monotonic negation is desired)

→ "Rules" as an alternative paradigm for knowledge modeling

What is a Rule?

- Logical Rules (predicate logic implications)
  - "F → G" (equivalent to "¬F v G")
  - Logical extension of a knowledge base (**static**)
  - **Open World**
  - Declarative (descriptive)
- Procedural Rules (e.g. production rules)
  - "If X then Y else Z"
  - Executable machine directive (**dynamic**)
  - **Operational** (meaning = effect on execution)
- Logic Programming (e.g. Prolog, F-Logic)
  - "`man(x) :- person(X), not woman(X)`"
  - Approximating logical semantics with procedural aspects, built-ins possible
  - Typically **Closed World**
  - "**semi-declarative**"
- Deduction rules of a calculus (e.g. rules for RDF semantics, lecture 1)
  - Rules not part of the knowledge base, "meta-rules"

Which Rule Language?

Rule languages are hardly compatible with each other → important to chose adequate rule language

Possible criteria:

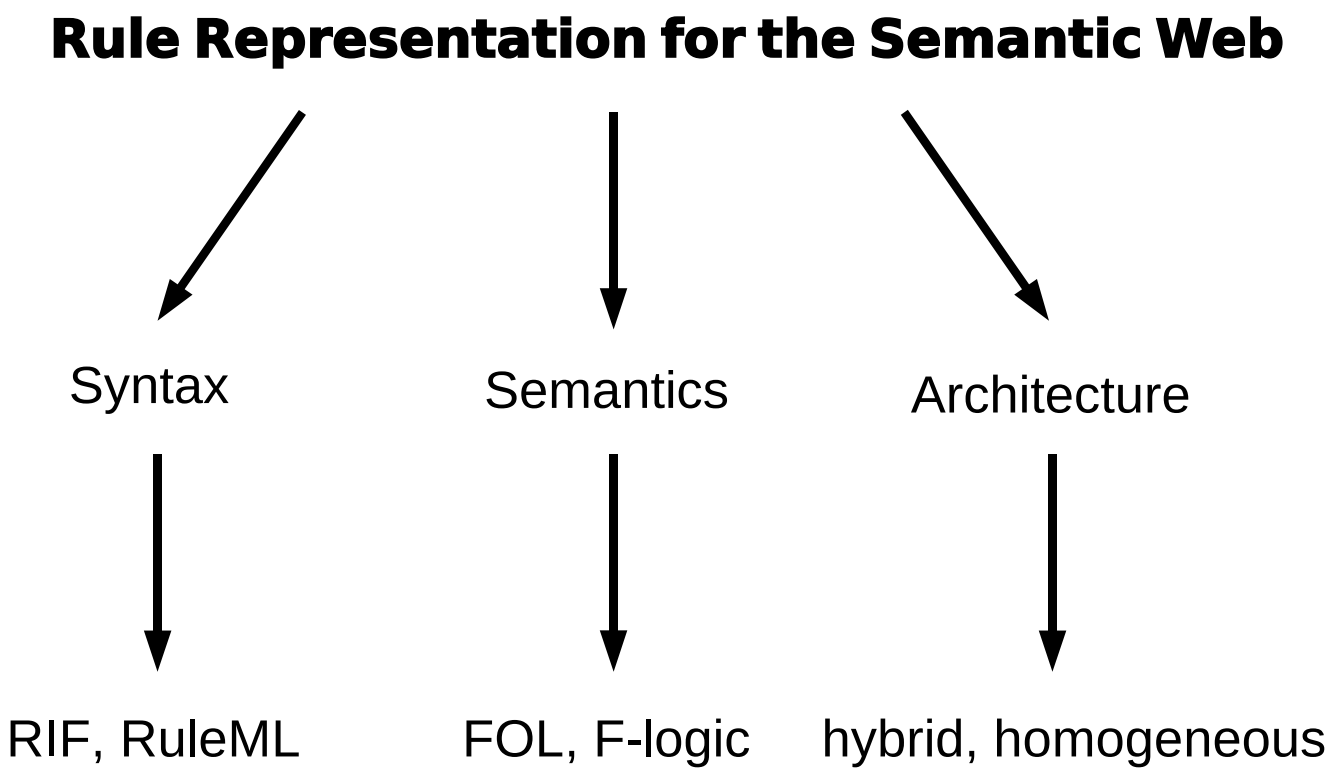
- Clear specification of syntax and semantics?
- Support by software tools?
- Which expressive features are needed?
- Complexity of implementation? Performance?
- Compatibility with other formats, e.g. OWL?
- Declarative (describing) or operational (programming)?
- …

Which Rule Language?

- **Logical Rules** (predicate logic implications)
  - Clearly defined, extensively researched, well understood
  - Very well compatible with OWL and RDF
  - Not decidable if unrestricted
- **Procedural Rules** (e.g. production rules)
  - Many independent approaches, vague definition
  - Used like programming languages, relation to RDF and OWL not clear
  - Efficient processing possible
- **Logic Programming** (e.g. Prolog, F-Logic)
  - Clearly defined, but many independent approaches
  - Partly compatible with OWL and RDF
  - Decidability/complexity depends very much on the chosen approach

→ In this lecture: predicate logic rules
(which are also the basis for logic programming)

Rule Representation



- e.g.: SWRL → OWL+RuleML syntax, FOL semantics, homogeneous

Syntax – Rule Markup

Objective

- rule encoding (markup)
- interoperability among application
- data processing, ontology mapping, ad hoc reasoning (RIF)

Markup Languages

- RuleML – Rule Markup Language
- RIF – Rule Interchange Format

Features

- XML-based syntax
- extensible
- interoperable with RDF/OWL

Semantics – Rule Meaning

Ideas from various rule systems

Logic Programming (LP), Production Rules Systems, Automatic Theorem Provers, Bussiness Rules, application-specific rules etc.

Challenges for rules and ontologies integration

- OWA in ontologies, CWA in LP and Rule-Based Systems
- Unique Name Assumption (UNA) in RBS
- Monotonic reasoning in ontology-based systems

Semantic for rules – most popular choices

- First Order Logic (FOL)
  - OWL ontologies based on Description Logic – subset of FOL
- Frame logic (F-logic)
- Stable Model Semantics, *no* fixed semantics

Hybrid/Homogeneous Architecture

Hybrid Approach

- loose integration, semantic separation between:
  - an ontology component – based on a DL variant
  - a rule component – usually a variant of Datalog
- unidirectional or bidirectional communication

Homogeneous Approach

- a single logical language
- no syntactic or semantic distinctions
- can be interpreted by a single reasoning engine
- union or intersection of the component languages

The union of the entire LP and DL fragments within First Order Logic is undecidable!

Semantic Web Rule Languages Zoo

1. Semantic Web Rule Language (SWRL)
   - expressive, yet undecidable, OWL and Horn rules **union**
2. DL-safe rules
   - SWRL rules applied to known individuals
3. Description Logic Programs (DLP)
   - decidable, limited expressivity, OWL and Horn rules **intersection**
4. OWL 2 DL
   - OWL 2 EL – simple terminology (TBox) reasoning
   - OWL 2 RL – reasoning about individuals (ABox)

Not only OWL!

5. Web Rule Language (WRL)
   - supports CWA and UNA, Well Founded/Perfect Model Semantics
6. TRIPLE language
   - Datalog-like rules cast on RDF data model, based on F-logic

Semantic Web Rule Languages Zoo

1. Semantic Web Rule Language (SWRL)
   - expressive, yet undecidable, OWL and Horn rules **union**
2. DL-safe rules
   - SWRL rules applied to known individuals
3. Description Logic Programs (DLP)
   - decidable, limited expressivity, OWL and Horn rules **intersection**
4. OWL 2 DL
   - OWL 2 EL – simple terminology (TBox) reasoning
   - OWL 2 RL – reasoning about individuals (ABox)

Not only OWL!

5. Web Rule Language (WRL)
   - supports CWA and UNA, Well Founded/Perfect Model Semantics
6. TRIPLE language
   - Datalog-like rules cast on RDF data model, based on F-logic

Outline

Outline

Predicate Logic as a Rule Language

- Rules as first-order logic implications (Horn clauses):

  $A1 \wedge A2 \wedge \ldots \wedge An \rightarrow H$ ("Body → Head")

  Example: "Man(x) ∧ happilyMarriedWith(x,y) → HappyHusband(x)"

- Constants, variables, function symbols can be used; but no negation
- Quantifiers are omitted: free variables considered universally quantified
- **Datalog:** rules without function symbols
  - Originally developed for deductive databases
  - Knowledge bases ("datalog programs") are sets of function-free Horn clauses
  - Decidable
  - efficiently implementable for large datasets (overall complexity ExpTime, i.e. "Draughts" or "Chess")

Outline

SWRL

How can datalog and OWL be combined?

SWRL – Semantic Web Rule Language [swirl]

- Proposal for a rule extension for OWL (W3C member submission)
- Idea: datalog rules referring to an OWL ontology
  → Symbols in rules can be OWL identifiers or new symbols
- Various further features and syntactic forms (not relevant here)

Semantics of SWRL

Combined semantics OWL DL + datalog?
→ use first-order mapping of OWL (lecture 2)

In effect:

- OWL individuals are datalog constants
- OWL classes are unary datalog predicates
- OWL properties are binary datalog predicates

→ A first-order interpretation can at the same time be a model for an OWL ontology and a datalog program
→ Entailment over OWL+datalog (SWRL) well-defined

Example: SWRL knowledge base

For readability, we abbreviate URIs strongly here; the datalog syntax does not follow a formal specification (SWRL XML is too verbose here)

```
(1)  Vegetarian(x) ∧ Fishproduct(y)  → dislikes(x,y)
(2)    ordered(x,y) ∧ dislikes(x,y)  → Unhappy(x)
(3)                   ordered(x,y)  → Dish(y)
(4) dislikes(x,z) ∧ Dish(y) ∧ contains(y,z) → dislikes(x,y)
(5)                                 → Vegetarian(markus)
(6)          Happy(x) ∧ Unhappy(x)  →

(7) markus rdf:type [
       rdf:type           owl:Restriction;
       owl:onProperty     ordered;
       owl:someValuesFrom ThaiCurry          ]
(8) ThaiCurry rdfs:subClassOf [
       rdf:type           owl:Restriction;
       owl:onProperty     contains;
       owl:someValuesFrom Fishproduct        ]
```

Example: SWRL knowledge base

OWL DL can be translated to first-order logic:

$$
\begin{array}{ll}
(1) & \mathtt{Vegetarian(x)} \land \mathtt{Fishproduct(y)} \rightarrow \mathtt{dislikes(x,y)} \\
(2) & \mathtt{ordered(x,y)} \land \mathtt{dislikes(x,y)} \rightarrow \mathtt{Unhappy(x)} \\
(3) & \mathtt{ordered(x,y)} \rightarrow \mathtt{Dish(y)} \\
(4) & \mathtt{dislikes(x,z)} \land \mathtt{Dish(y)} \land \mathtt{contains(y,z)} \rightarrow \mathtt{dislikes(x,y)} \\
(5) & \rightarrow \mathtt{Vegetarian(markus)} \\
(6) & \mathtt{Happy(x)} \land \mathtt{Unhappy(x)} \rightarrow \\
 & \\
(7) & \exists \mathtt{y.ordered(markus,y)} \land \mathtt{ThaiCurry(y)} \\
(8) & \forall \mathtt{x.ThaiCurry(x)} \rightarrow (\exists \mathtt{y.contains(x,y)} \land \mathtt{FishProduct(y)})
\end{array}
$$

→ Semantics completely defined
→ Expected conclusion: **`Unhappy(markus)`**

Note: empty rule heads correspond to “false” (rule body must never be true)
empty rule bodies correspond to “true” (rule head must always be true)

How Hard is SWRL?

- Deduction for OWL DL is NExpTime-complete
- Deduction for OWL 2 DL is N2ExpTime-complete
- Deduction in datalog is ExpTime-complete

→ How hard is deduction for SWRL?

Deduction for SWRL is undecidable

(for OWL and thus for OWL 2, even for OWL EL)

Markus Krötzsch, Sebastian Rudolph: Semantic Web Modeling Languages, ESSLLI 2009, Bordeaux semantic-web-book.org 12

Undecidability of SWRL

SWRL is undecidable:

There is no algorithm that can draw *all* logical conclusions from *all* SWRL knowledge bases, even with unlimited time and resources.

Practically possible:

- Algorithms that draw *all* conclusions for *some* SWRL knowledge bases
- Algorithms that draw *some* conclusions from *all* SWRL knowledge bases

→ Both trivial if "some" refers to very few things

Decidable Fragments of SWRL

Which classes of SWRL knowledge bases allow for complete inference algorithms?

- All SWRL knowledge bases consisting only of OWL (2) axioms
- All SWRL knowledge bases only consisting of datalog rules
- Every fixed finite class of SWRL knowledge bases

→ Which more interesting decidable fragments exist?

- **Description Logic Rules**
- **DL-safe Rules**

Markus Krötzsch, Sebastian Rudolph: Semantic Web Modeling Languages, ESSLLI 2009, Bordeaux semantic-web-book.org 14

Outline

Description Logic Rules

Observation:
Some SWRL-rules can already be expressed in OWL 2.

- Identifying all such Description Logic Rules leads to a decidable fragment
- Goal: Exploit "hidden" expressivity of OWL 2
- Implementation directly by OWL 2 tools

Markus Krötzsch, Sebastian Rudolph: Semantic Web Modeling Languages, ESSLLI 2009, Bordeaux semantic-web-book.org 15

Simple Rules in OWL 2: Examples

- Simple OWL 2 axioms correspond to rules:

```
Class1      rdfs:subClassOf      Class2 .
Property1   rdfs:subPropertyOf   Property2 .
```

correspond to

$$\begin{aligned} \mathtt{Class1}(x) &\rightarrow \mathtt{Class2}(x) \\ \mathtt{Property1}(x,y) &\rightarrow \mathtt{Property2}(x,y) \ . \end{aligned}$$

Markus Krötzsch, Sebastian Rudolph: Semantic Web Modeling Languages, ESSLLI 2009, Bordeaux semantic-web-book.org 16

Simple Rules in OWL 2: Examples

- Some classes can be decomposed in rules:

```
[ owl:intersectionOf ( Happy  Unhappy ) ]
                       rdfs:subClassOf      owl:Nothing .

[ rdf:type             owl:Restriction;
  owl:onProperty       livesIn;
  owl:someValuesFrom [ rdf:type  owl:Restriction;
                       owl:onProperty  locatedIn;
                       owl:someValuesFrom EUCountry ]
                     ]
] rdfs:subClassOf  EUCitizen .
```

correspond to

$$Happy(x) \wedge Unhappy(x) \rightarrow$$
$$livesIn(x,y) \wedge locatedIn(y,z) \wedge EUCountry(z) \rightarrow EUCitizen(x)$$

Markus Krötzsch, Sebastian Rudolph: Semantic Web Modeling Languages, ESSLLI 2009, Bordeaux semantic-web-book.org 17

Simple Rules in OWL 2: Examples

- Property chains provide further rule-like axioms:

```
hasUncle owl:PropertyChainAxiom ( hasParent  hasBrother ).
```

correspond to

$$hasParent(x,y) \wedge hasBrother(y,z) \rightarrow hasUncle(x,z)$$

→ In all examples, mapping can also be inverted (expressing rules in OWL 2)

Can all rules be expressed in OWL2?

- **No**, but significantly **more** than in SWRL based on OWL(1)
- some restrictions overcome by special OWL2 constructs and mechanisms
  - inverting properties,
  - replacing concepts with properties – `hasSelf`
- The is an algorithm for transforming rules into OWL2
- And a definition when an SWRL rule is DRL

Outline

DL-safe Rules

Observation: Datalog is decidable since rules can be applied in only finitely many ways: variables represent only constants.

- Variables in SWRL might represent arbitrarily many infered individuals
- Goal: Make rules "safe" by restricting possible variable assignments
- **DL-safe rules** as another decidable fragment of SWRL

DL-safe Rules: Definition

Rules now may also include non-OWL predicates:

- A **datalog atom** is an atom with a predicate symbol that does not occur as a class or property in any OWL axiom.

A SWRL rule is **DL-safe** if:

- Every variable in the rule head occurs in a datalog atom in the body.

→ Only constant symbols relevant when considering variable assignments in datalog atoms.

Enforcing DL-Safety

- Example:

$$\mathit{ordered}(x,y) \wedge \mathit{dislikes}(x,y) \rightarrow \mathit{Unhappy}(x)$$

→ not DL-safe if ***ordered*** or ***dislikes*** occur in OWL axioms

- Enforcing DL-safety by restricting rules to **named** individuals:

$$\mathit{ordered}(x,y) \wedge \mathit{dislikes}(x,y) \wedge O(x) \wedge O(y) \rightarrow \mathit{Unhappy}(x)$$

where a fact → $O(a)$ is added for all individuals ***a***.

→ Rule only applicable to **named** OWL individuals

DL-Safe Rules in Practice

- OWL 2 with DL-safe rules is decidable
- Naïve implementation: each rule expressible by finitely many DL rules where all variables are replaced by individual symbols in all possible ways (very inefficient)
- No increase in worst-case complexity

Implementations:

- Basic support in some reasoners (KAON2, Pellet)
- Implementation in tableau-based tools complicated

Summary: Rules

- SWRL ("OWL+ datalog") is undecidable
- Description Logic Rules:
  - SWRL fragment expressible in OWL 2
  - Supported indirectly by OWL 2 reasoners
  - Definition and translation based on dependency graph
- DL-safe rules:
  - SWRL fragment where variables can only assume concrete values
  - Support by some OWL reasoners
  - DL-safety can be enforced (also done implicitly in some tools)
- Combination OWL 2 + DL Rules + DL-safe rules possible

Rules for the Semantic Web?

- Standards and best-practices for rules still missing
- SWRL syntax most widely used in applications
- W3C RIF (Rule Interchange Format) Working Group
  - Standard for various rule languages, also SWRL-like rules
  - Various new features, e.g. syntax from Frame Logic
  - Official specification expected by end 2009
- Many studies on interfacing Logic Programming and OWL
- OWL 2 RL: a profile that can be translated to datalog rules
  (note: inverse direction of Description Logic Rules)
  → enables some interoperability OWL 2 ↔ RIF
- Operational "inference rules" or "production rules" supported by some RDF-stores (e.g. Jena)

Outline

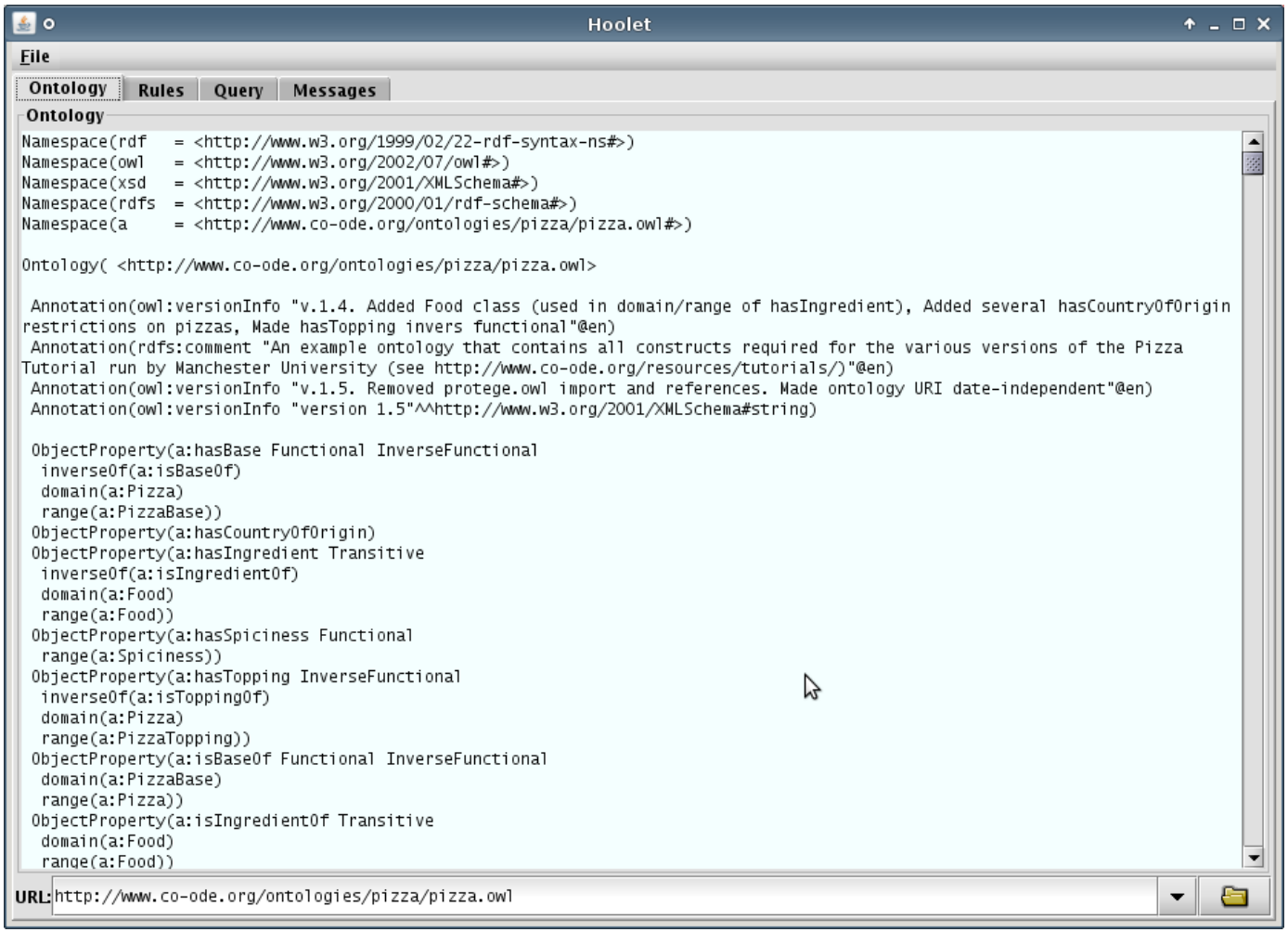
Outline

Bossam

- RETE algorithm
- OWL, SWRL, RuleML ontologies, RDF(S), Bossam documents
- standard ontology tasks
- no SPARQL support

Hoolet

- FOL reasoning for OWL DL and SWRL



Pellet

- tableaux algorithm
- OWL DL, all OWL 2 languages
- standard ontology tasks
- ontology analysis and repairing
- debugging
- incremental reasoning
- SPARQL-DL support
- datatype reasoning
- SWRL support

KAON2

- infrastructure for managing OWL, SWRL, F-logic ontologies
- API for ontology management
- SPARQL support
- DIG interface
- instance import from RDB

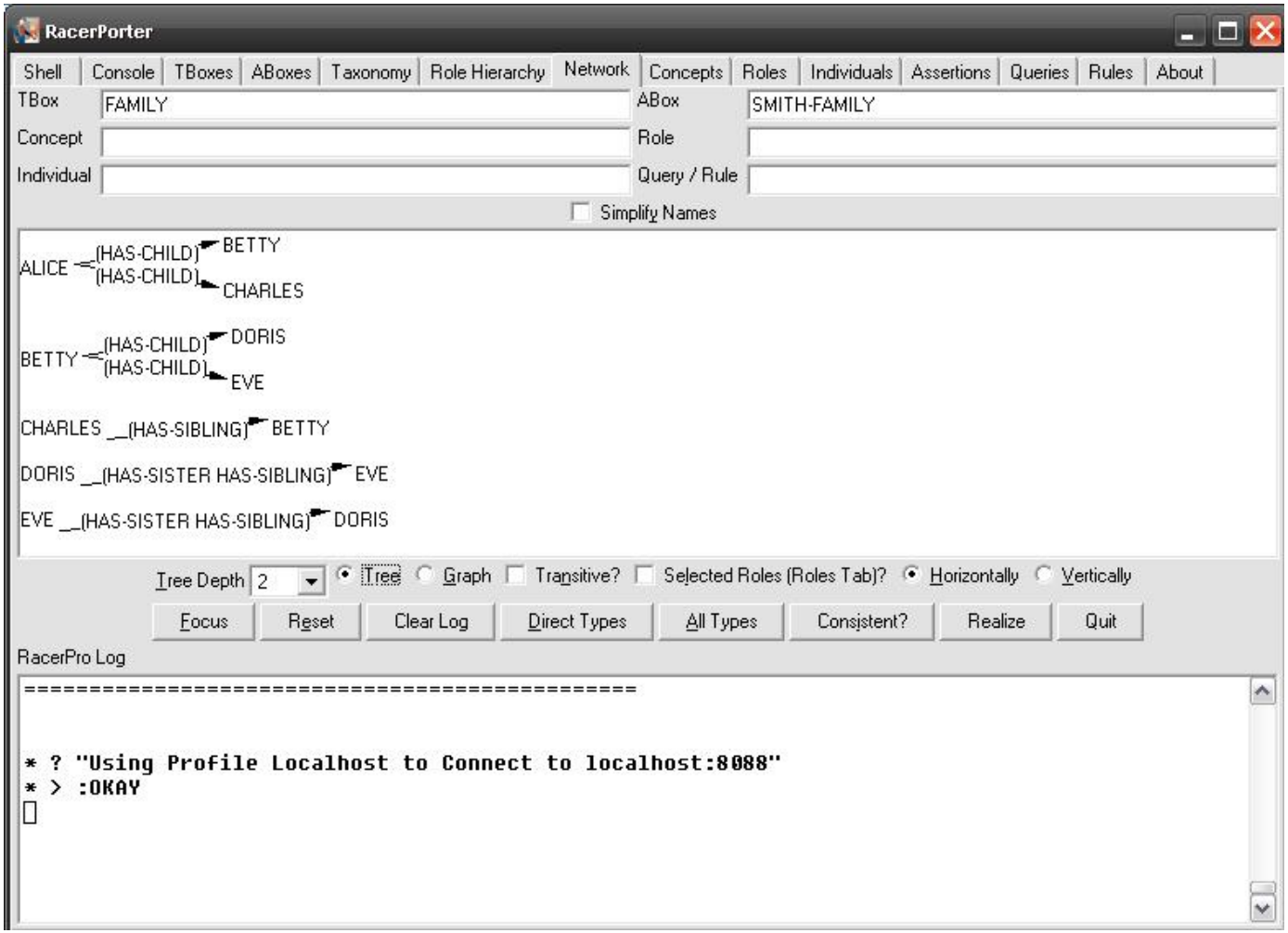
FaCT++

- tableau algorithms
- OWL Lite, OWL DL, partly OWL 2
- incremental and partial datatype reasoning
- DIG interface

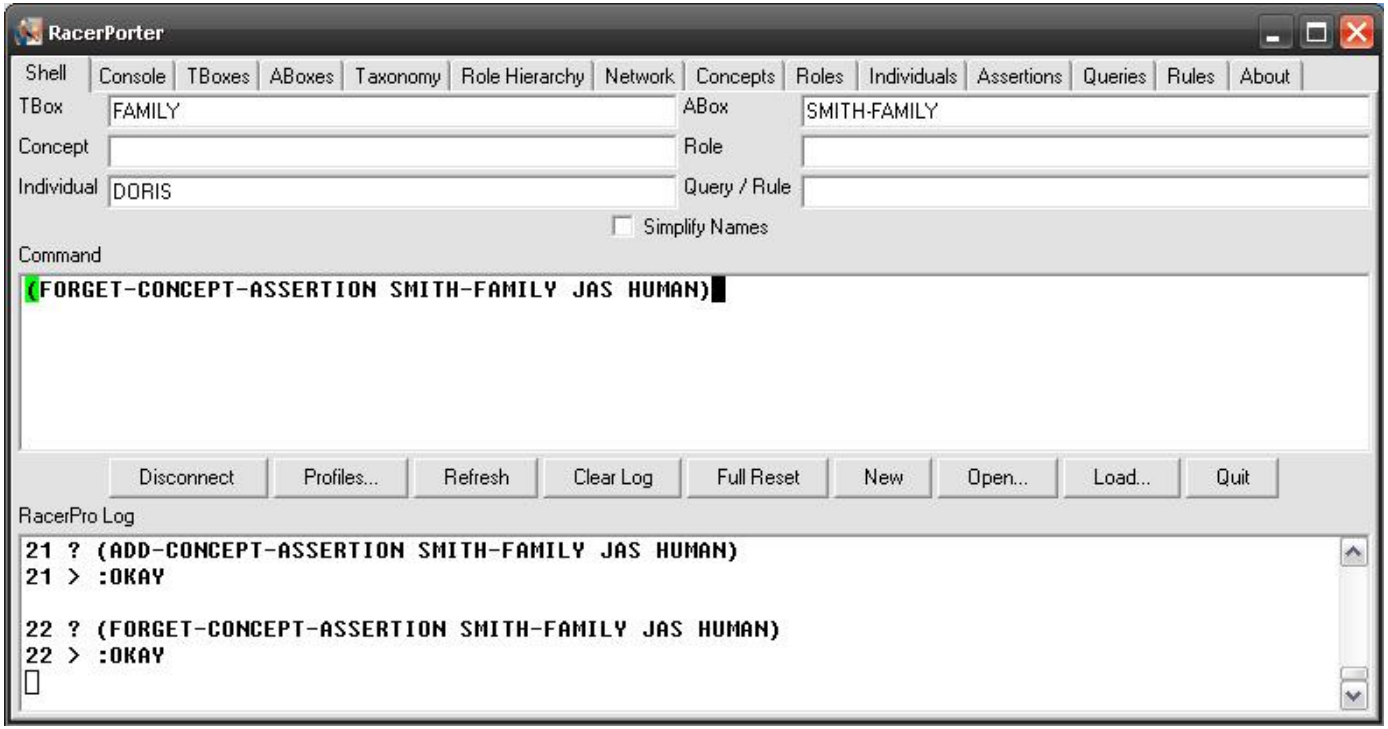
RACER (Pro)

- commercial, tableaux algorithm
- OWL, partly OWL2
- class relationships, incremental reasoning
- nRLQ engine
- interfaces: RacerPorter, JRacer, LRacer, DIG

RacerPorter

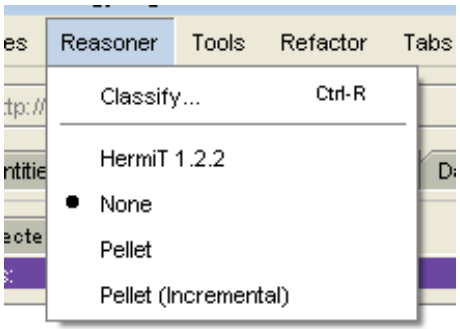


RACER: nRQL



HermiT

- novel hypertableau algorithm
- standard ontology tasks
- class relationships



CEL

- polynomial-time classifier for OWL 2 EL ontologies
- standard ontology tasks
- relationships among classes

Quonto

- OWL DL-Lite
- relationships among classes
- queries
- concepts, roles and attributes satisfiability

Owlgres

- fast scalable reasoner
- SPARQL-DL support
- transaction support
- cooperates with PostgreSQL

Tests – Conclusions

- **CEL** is best for OWL2EL ontologies
- **FaCT++ (Protege 4.02)** is usually the fastest
- **Pellet (Protege 4.02)** failed in 3 tests. However, for those tests it passed Pellet had the best average speed
- **HermiT (Protege 4.1)** – the slowest
- **Pellet(Incremental) (Protege 4.1)** – as the only one has passed *all* tests
- different tools are better for different ontologies (e.g. big TBox vs. big ABox)

See also:

- `http://ai.ia.agh.edu.pl/wiki/pl:dydaktyka:miw:2010:dltls:start`
- `http://ai.ia.agh.edu.pl/wiki/pl:dydaktyka:miw:2010:owl2tls:start`
- `http://www.cs.man.ac.uk/~sattler/reasoners.html`

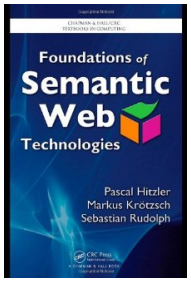
Questions

Any questions?

Thank you

Thank you for your attention!

http://geist.agh.edu.pl
GEIST Research Group



Powered by LaTeX