

The RacerPro Environment for Lisp-based Semantic Web Applications

Michael Wessel

Racer Systems

GmbH & Co. KG
Blumenau 50
22089 Hamburg

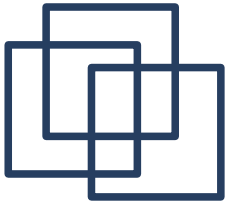
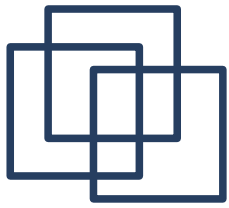
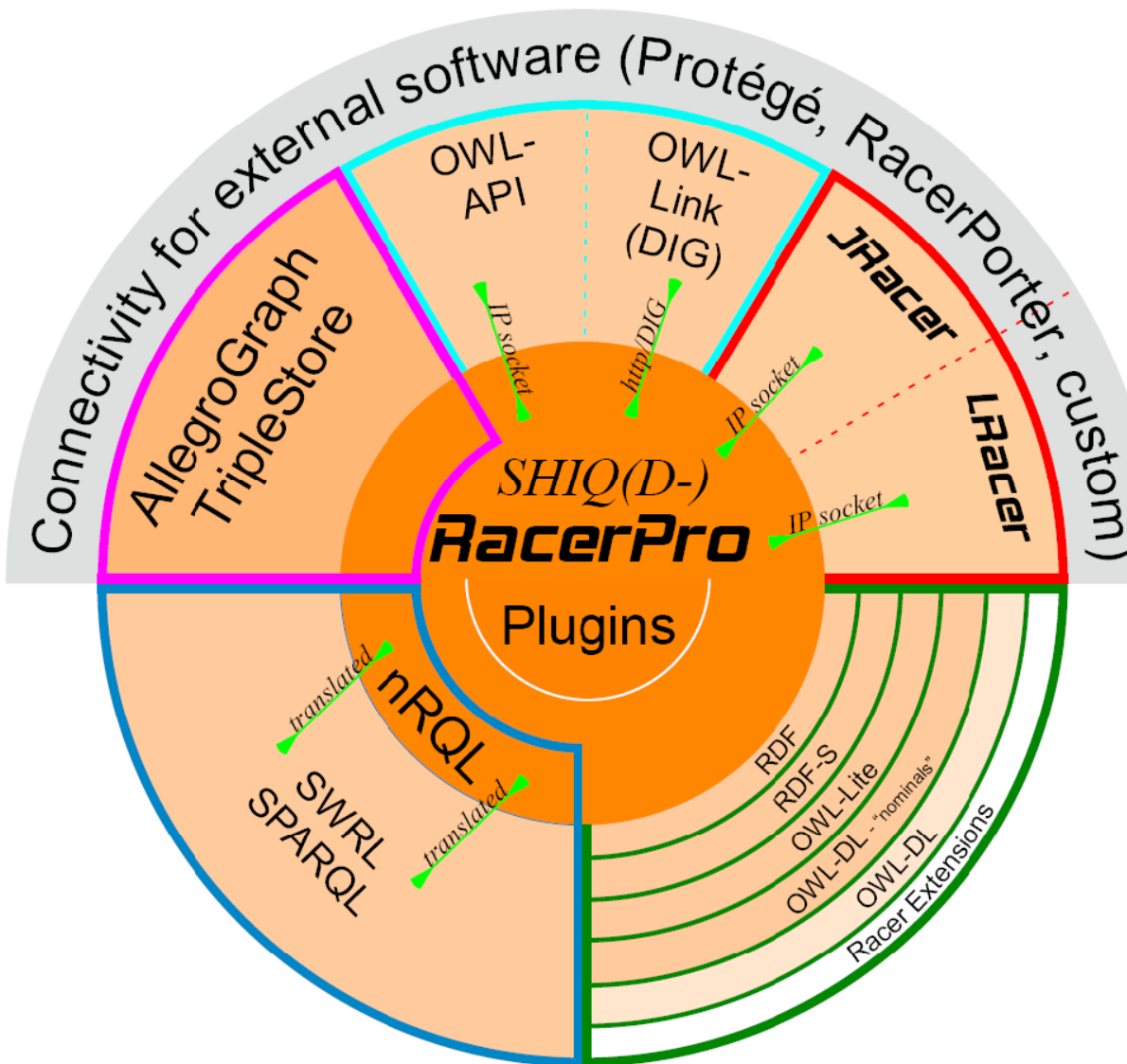


Table of Contents

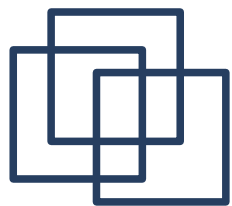
- History
 - Racer, Racer Systems, RacerPro
- Background
 - idea of the Semantic Web & logic-based Knowledge Representation
- Reasoning with formal ontologies
 - RacerPro & RacerPorter reasoning demo
 - W3C SemWeb „languages“ (OWL, RDFS, SPARQL, SWRL, ...)
- Semantic Web programming in the „RacerPro environment“
 - JRacer, LRacer, MiniLisp, extensibility, OWLAPI, OWLlink, ...
- The role of Lisp



RacerPro - Architecture & History



- Started as **Racer** at the University of Hamburg in **1998**, project of **Volker Haarslev & Ralf Möller**
- First description logic (DL) reasoner „of the new generation“ of highly optimized DL systems with **ABox (individuals, relations)**
- One of the **first OWL DL (-) systems (2002)**, DL *SHIQ(D)*
- Commercial offspring **RacerPro** by **Racer Systems (2004 - today)**
- Expressive query language **nRQL**
- First DL system that could give **complete answers** to the **LUBM Benchmark** queries (2004)
- First DL system with **inference-aware SWRL & SPARQL**
- Main memory-based
- Recently: Integrated **AllegroGraph**
- Some **special-purpose representations and reasoning**
- **Free for education & research**



Racer Systems – Partners & Friends

Prof.
Ralf Möller

Michael
Wessel

Prof.
Volker Haarsley

President
Kay Hidde



Research in
DLs, OWL,
Racer-projects

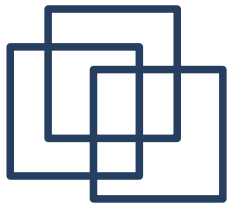
AllegroGraph
ACL
mutual
redistribution



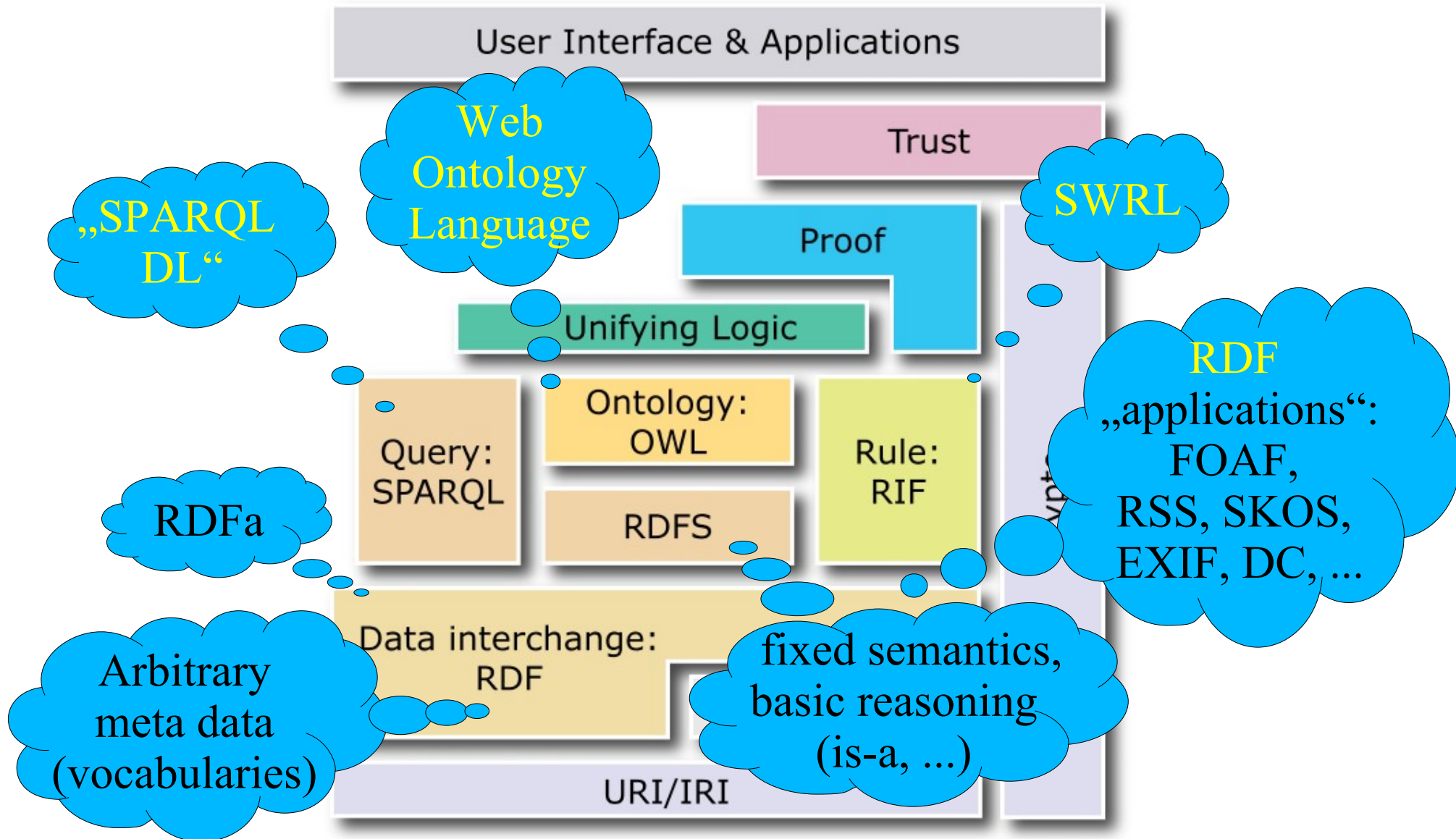


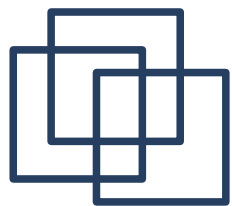
The Semantic Web - „A Web of Data“

- „The big database in the sky“
 - Web 1.0 – syntactic web, technical basis (HTTP, HTML, ...)
 - Web 2.0 – social / community web **for people**
(Wikis, Blogs, Boards, Flickr, Blogger, Twitter, ...)
 - folksonomies („(geo) tagging“)
 - Web 3.0 – Web 2.0 plus meta data **for machines**
 - meta data = page annotations, service descriptions, ...
provided in terms of **ontologies**
(provide explicit formal semantics for terms → reasoning)
 - annotations = logical propositions about resources
identified by URIs
 - SPARQL endpoints & RDFa (RDF in HTML)
- Technically, the SemWeb is not really a „database“ (see below)



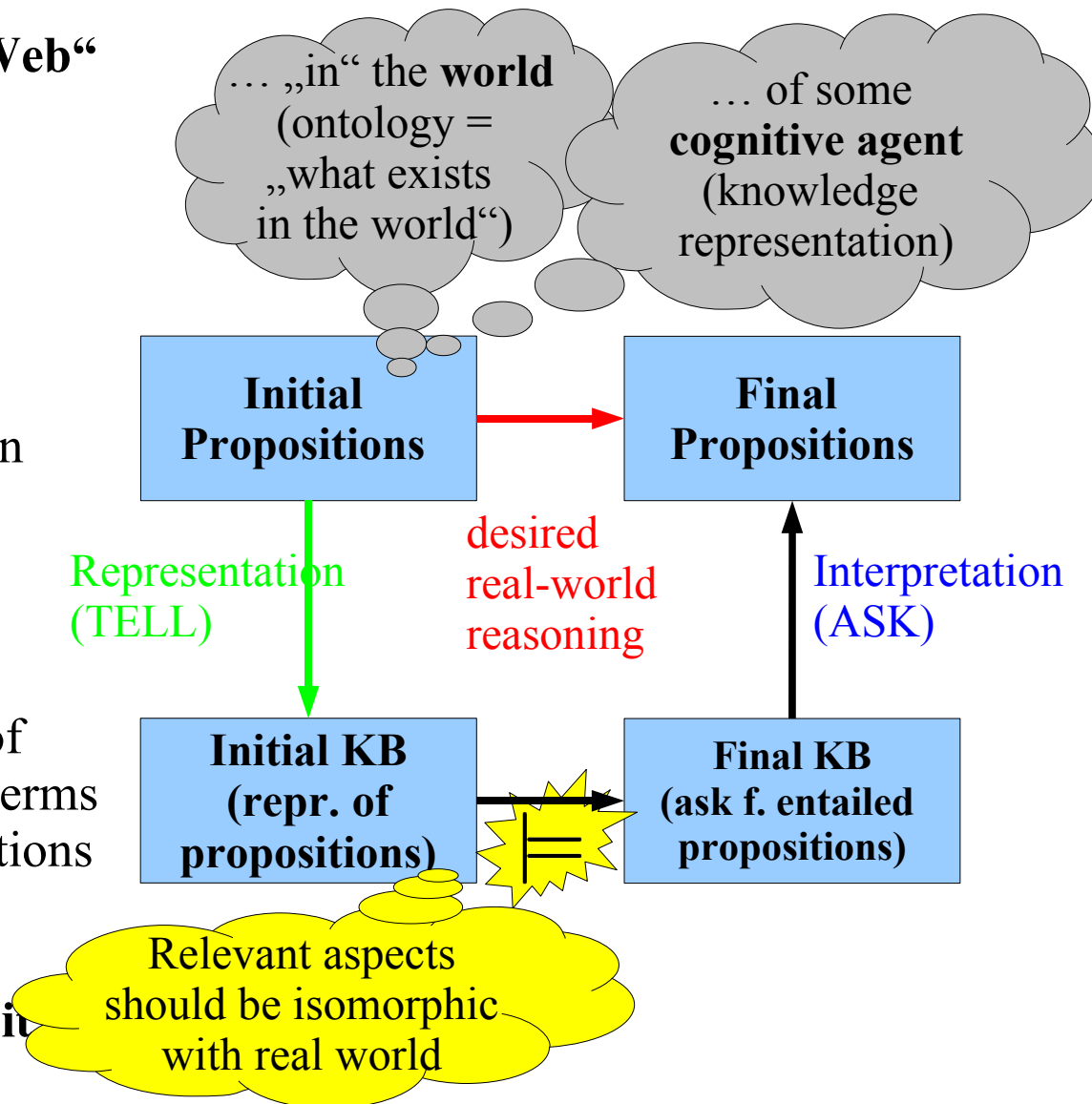
Semantic Web Stack (Layer Cake) © W3C

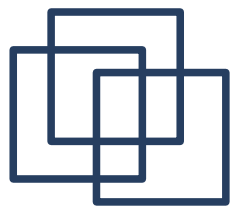




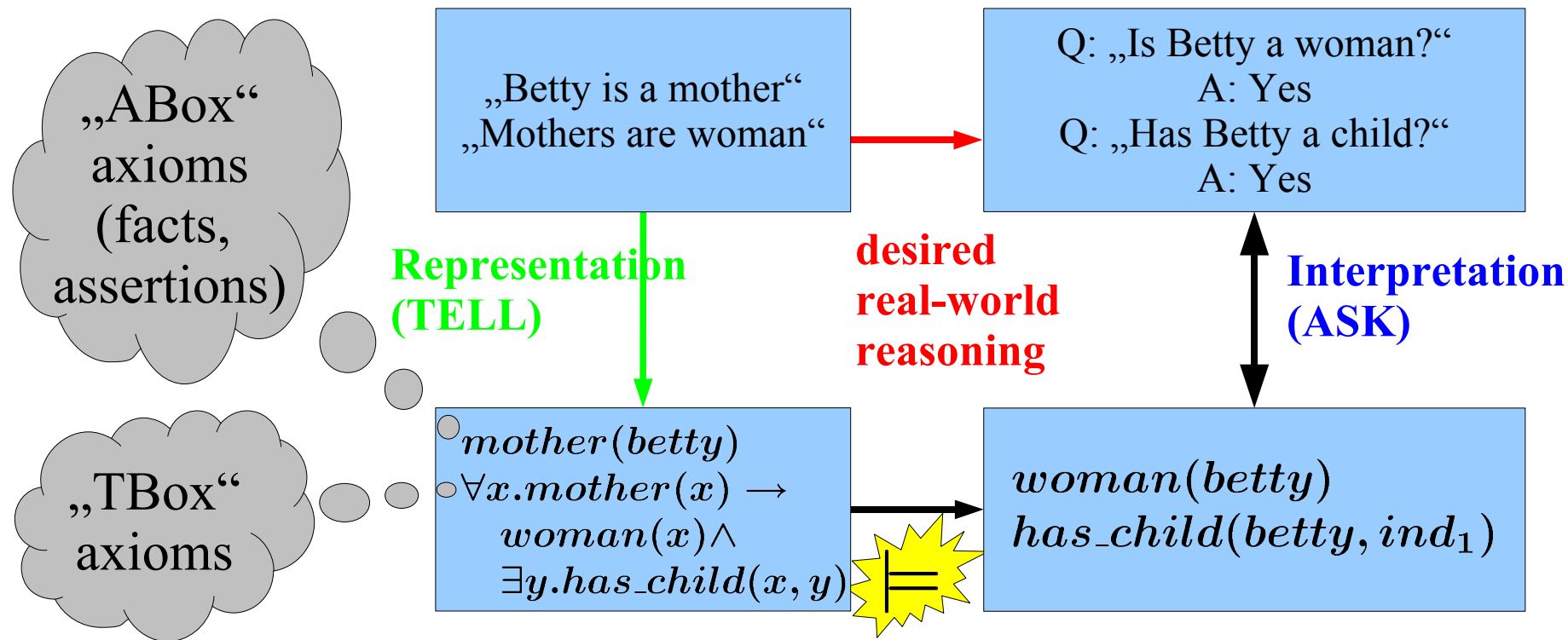
Logic-Based Knowledge Representation & Reasoning

- **SemWeb = „KR&R on the Web“**
- Replace real-world reasoning with computational operations performed in a model (\models)
- Model \sim representation \sim KB
- **Ontology: explicit specification of a conceptualization**
 - „formal account of what exists in the world“
 - logic-based definitions of concepts & relations in terms of other concepts & relations
 - automated reasoning
 - **inference makes implicit knowledge explicit**

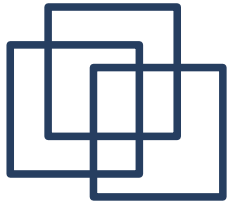




Logic-Based KR&R (2)

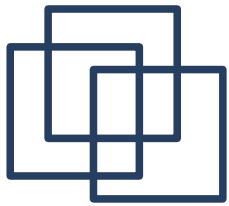


- First-Order Predicate Logic: \models undecidable
- Description Logics (OWL DL): decidable, but NEXPTIME complete



Reasoning with Formal Ontologies

- **Demo** of some standard inferences using RacerPro & RacerPorter
 - Basis: „People & Pets“ ontology
by **Sean Bechhofer (Univ. of Manchester)**
 - but will use KRSS / Racer Lisp syntax in this demo
 - show some OWL syntaxes later



RacerPorter – The Listener („Racer Shell“)

RacerPorter

Profiles | **Shell** | TBoxes | ABoxes | Concepts | Roles | Individuals | Assertions | Axioms | Taxonomy | Role Hierarchy | ABox Graph | Query IO | Queries + Rules | Def. Queries | Log | About

Active Profile: **4: Localhost / Big TBoxes, Big ABoxes**

Namespace (#!, *n*): **default**

TBox (*t*): **default**

Concept (*c*): 0

Individual (*i*): 0

Role (*r*): 0

Axiom (*ax*): 0

Definition (*def* = Name)

Ontology Container (*oo*)

Reasoner Container (*or*): **OWLAPI-KB**

Request: **20 : (get-namespace-prefixes)** Response: **20 : READY**

Classic Layout | < | < 2 / 2

[*] ? Cannot start RacerPro @ localhost:8088, server
[*] > :ERROR

[*] ? Automatically connected to RacerPro 2.0 running on localhost:8088
[*] > (:OKAY "RacerPro 2.0 running on localhost:8088")

[1] ? (get-racer-version)
[1] > "2.0"

[2] ? (owl-read-file []

Arguments of owl-read-file: FILENAME &KEY (VERBOSE *TBOX-VERBOSE*) (INIT T) (KB-NAME) (LOCATOR) RECURSIVE IGNORE-IMPOR

Function Doc | Complete Input

Sel. Concepts := Last Result | Sel. Roles := Last Result | Sel. Individuals := Last Result

Clear Selection | Clear Selection | Clear Selection

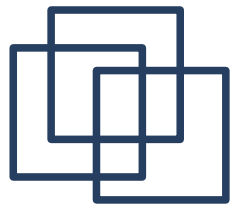
Show Manual | Save Shell... | Clear Shell | **Full Reset** | New Editor | Open in Editor... | Load... | Quit | Shutdown RacerPro & Quit

RacerPorter

- (owl-read-file
- FILENAME
- &KEY
- (VERBOSE *TBOX-VERBOSE*)
- (INIT T)
- (KB-NAME)
- (LOCATOR)
- RECURSIVE
- IGNORE-IMPORT
- IMPORT-META-ONTOLOGIES
- EXCLUDED-META-ONTOLOGIES
- FIRE-RULES
- MAINTAIN-OWLAPI-AXIOMS
- &ALLOW-OTHER-KEYS)

OK

Comfortable RacerPro listener with completion, function doc, history, pretty printing, ...



The Racer Editor with Some Example Queries

RacerEditor for knowledge base creation, expression evaluation, ... supports OWL RDF, KRSS, SPARQL

RacerPorter

Profiles	Shell	TBoxes	ABoxes	Concepts	Roles	Individuals	Assertions
Active Profile		4: Localhost / Big TBoxes, Big ABoxes					
TBox (**)		/home/mi.wessel/KBs/people+pets.owl					
Concept (*c*)							
Individual (*i*)							
Query / Rule (*qr*)							
Reasoner Container (*or*)		OWLAPI-KB					
Request		47: (get-namespace-prefixes)					

Classic Layout |< < 2 / 2

```
[*] ? Cannot start RacerPro @ localhost:8088, server is already running
[*] > :ERROR

[*] ? Automatically connected to RacerPro 2.0 running on localhost:8088
[*] > (:OKAY "RacerPro 2.0 running on localhost:8088 (case: preserve)")

[1] ? (get-racer-version)
[1] > "2.0"

[2] ? (full-reset)
[2] > :okay-full-reset

[3] ? (racer-read-file "z:/temp/people+pets.racer")
(in-tbox /home/mi.wessel/KBs/people+pets.owl size 124 role-size 26)
Duplicate definition (or animal (some part_of animal)) for Axiom_2
1/KBs/people+pets.owl --> /home/mi.wessel/KBs/people+pets.owl

[3] > :OKAY

[4] ? (retrieve (?x ?y) (and (?x cat_owner) (?x ?y has_pet) (?y cat)))

Concept (animal) causes a cycle in TBox /home/mi.wessel/KBs/people+pets.owl
Concept (plant) causes a cycle in TBox /home/mi.wessel/KBs/people+pets.owl
Concept (Axiom_2) causes a cycle in TBox /home/mi.wessel/KBs/people+pets.owl
Classifying TBox...
Concept (mad_cow) is incoherent in TBox /home/mi.wessel/KBs/people+pets.owl
[4] > (((?x Minnie) (?y Tom)) ((?x Fred) (?y Tibbs)))

[5] ?
```

Arguments of owl-read-file: FILENAME &KEY (VERBOSE *TBOX-VERBOSE*) (INIT T) (KB-NAME) (LOCATOR) RECURSIVE IGNORE-IMPOR

Function Doc Complete Input

Sel. Concepts := Last Result Sel. Roles := Last Result Sel. Individuals := Last Result

Clear Selection Clear Selection Clear Selection

Show Manual Save Shell... Clear Shell Full Reset New Editor Open in Editor... Load... Quit Shutdown RacerPro & Quit

RacerEditor

```
File Edit Buffer

;;; Old_lady not explicit

;;; ABox Graph : Minnie has_pet Tom
;;; Select Tom -> Describe -> Cat
;;; Assertions Tab: nothing was
;;; Is a cat by inference (see
;;; Interplay of definitions <

;;; The classes an individual is
;;; is computed from the definit
;;; multiple classification

;;; -----
;;;
;;; ABox Queries
;;; (Go to ABox Graph?)
;;;

(concept-instances cat_owner)

(retrieve (?x)
  (?x cat_owner))

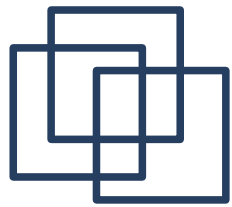
(retrieve (?x ?y)
  (and (?x cat_owner)
        (?x ?y has_pet)))

(retrieve (?x ?y)
  (and (?x cat_owner)
        (?x ?y has_pet)
        (?y cat)))

(retrieve (?x ?y (direct-types ?y))
  (and (?x pet_owner)
        (?x ?y has_pet)
        :dont-show-lambdas-p t))

;;;

Finished evaluating
RacerPro 2.0 running on localhost:8088 (case: preserve) ---- query
```



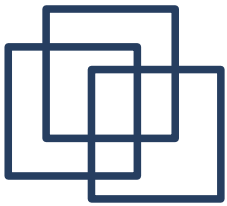
The Class (Concept) Hierarchy („Taxonomy“)

The screenshot shows the RacerPorter application interface. The top menu bar includes Profiles, Shell, TBoxes, ABoxes, Concepts, Roles, Individuals, Assertions, Axioms, and Taxonomy. The main window displays a concept hierarchy starting from (animal) and branching into (person), (grownup), (kid), and (pet_owner). The (person) node further branches into (cat), (dog), and (duck). The (grownup) node branches into (driver), (man), and (woman). The (pet_owner) node branches into (animal_lover), (cat_owner), and (dog_owner). The (cat_owner) node is highlighted with a red box and contains the concept (old_lady). The console window at the bottom shows the command `(describe-concept old_lady /home/mi.wessel/KBs/people+pets.owl)` and its output, including the primitive definition and synonyms.

```
(equivalent cat_liker
  (and person
    (some likes cat)))
(equivalent cat_owner
  (and person
    (some has_pet cat)))
(implies old_lady
  (and (all has_pet cat)
    (some has_pet animal)))
(define-primitive-role has_pet
:parents likes ...)
```

derived logical consequence („|=“):
cat owners are
cat likers,
old ladies are
cat owners!

```
[5] ? (describe-concept old_lady /home/mi.wessel/KBs/people+pets.owl)
[5] > (old_lady
      :old-primitive-definition
      (and
        (and (all has_pet cat) (some has_pet animal))
        (and person female elderly))
      :synonyms
      (old_lady)
      :parents
      ((elderly) (woman) (cat_owner))
      :children
```



The Relation (Role / Property) Hierarchy

RacerPorter

Profiles | Shell | TBoxes | ABoxes | Concepts | Roles | Individuals | Assertions | Axioms | Taxo...

Active Profile: 4: Localhost / Big TBoxes, Big ABoxes

TBox (*t*): /home/mi.wesselKBs/people+pets.owl

Concept (*c*): cat (4)

Individual (*i*): Tom (1)

Query / Rule (*qor*):

Reasoner Container (*or*): OWLAPI-KB

Request: 63: (describe-role has_pet /home/mi.wesselKBs/people+pet

Classic Layout | < | 13 / 13

(likes) — (has_pet)

(top-role) — (part_of), (eats), (drives), (works_for), (reads), (is_pet_of), (has_parent) — (has_father), (has_mother), (has_part), (has_child), (eaten_by), (service_number)

Relation hierarchy - „has pet“ is a subrelation of „likes“ (having a pet implies that you like it)

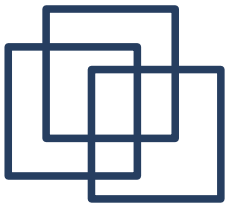
```
(equivalent cat_liker
  (and person
    (some likes cat)))
(equivalent cat_owner
  (and person
    (some has_pet cat)))
(implies old_lady
  (and (all has_pet cat)
    (some has_pet animal)))
(define-primitive-role has_pet
  :parents likes ...)
```

Print Graph

Parents Synonyms

Info

```
[07] ? (describe-role has_pet /home/mi.wesselKBs/...
[7] > (has_pet
  :synonyms
  (has_pet)
  :domain
  person
  :range
  animal
  :parents
  (likes)
  :children
```



Individuals & Relationships – ABox Graph

Another inference
 („reasoning about data“):
 Minnie is an old lady because
 she is a female elderly person.
 Old ladies are cat owners →
 Tom is a cat!

The screenshot shows the RacerPorter interface with the following elements:

- TBox (Top):** Contains logical rules such as:


```
(equivalent old_lady
  (and person female elderly))
(implies old_lady
  (and (all has_pet cat)
  (some has_pet animal)))
```
- ABox (Bottom):** Contains specific instances:

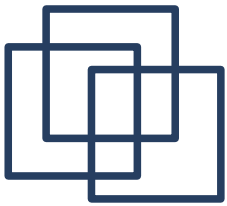

```
(instance Minnie elderly)
(instance Minnie female)
(related Minnie Tom has_pet)
(instance Tom top)
```
- Graph (Middle):** A graph showing relationships between individuals. A red box highlights the instance:


```
(Minnie) __ (has_pet likes) (Tom) __ (is_pet_of) (:BACK-TO Minnie)
```
- Info Panel (Bottom Left):** Shows a query result:


```
[6] ? {individual-direct-types Tom /home/mi.wessel/KBs/people+pets.owl}
[6] > ((cat) (pet))
```

TBox

ABox



Inspecting Class Assertions for Tom

RacerPorter

Profiles Shell TBoxes ABoxes Concepts Roles Individuals **Assertions** Axioms Taxonomy Role Hierarchy ABox Graph Query IO Queries + Rules Def. Queries Log About

Active Profile	4: Localhost / Big TBoxes, Big ABoxes			Namespace (#!, *n*)	
TBox (*t*)	/home/mi.wessel/KBs/people+pets.owl			ABox (*a*)	/home/mi.wessel/KBs/people+pets.owl
Concept (*c*)	cat			Role (*r*)	has_pet
Individual (*i*)	Tom			Axiom (*ax*)	1
Query / Rule (*qor*)				Definition (*def* = Name)	0
Reasoner Container (*or*)	OWLAPI-KB			Ontology Container (*oo*)	

Request: 68 : (all-concept-assertions | /home/mi.wessel/KBs/people+pe
Response: 68 : CACHE-HIT

Classic Layout |< < 18 / 18 > >| Delete Delete All Recover Simplify Sel. First Sel. Only Arg. Comp. Abort Racer Request

(Tom_top)

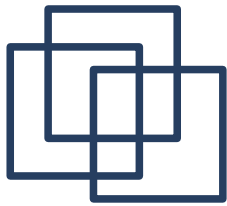
It is not asserted explicitly that Tom is a cat! („Class Assertion“) (top = thing concept)

All Concepts Cur. Concept Sel. Concepts All Inds. Cur. Ind. Sel. Inds. All Roles Cur. Role Sel. Roles
 Concept A Role A Same As A Different From A Attribute A Constraint A Annotation CA Annotation RA

Clear Selection Delete Selected

Info

```
[0] ? (describe-role has_pet /home/mi.wessel/KBs/people+pets.owl)
[7] > (has_pet
      :synonyms
      (has_pet)
      :domain
      person
      :range
      animal
      :parents
      (likes)
      :children
```

Relation („Role“) Assertions for Tom

The screenshot shows the RacerPorter application window. The 'Assertions' tab is active, displaying a table of assertions. The table has columns for 'Request', 'Response', and 'Count'. The first row shows a request for '69 : (all-role-assertions /home/mi.wessel/KBs/people+pets.o)' and a response of '69 : READY'. Below the table, there are navigation buttons and a search bar. The main display area shows the assertion '(Minnie Tom) has_pet.' with a blue thought bubble containing the text 'It is asserted that Tom is a pet of Minnie („Role assertion“)'. At the bottom, there are radio buttons for selecting different types of assertions and a console window showing the command '? (describe-role has_pet /home/mi.wessel/KBs/people+pets.owl)' and its output.

Request	Response	Count
69 : (all-role-assertions /home/mi.wessel/KBs/people+pets.o)	69 : READY	

Classic Layout | < | > | 19 / 19 | Delete | Delete All | Recover | Simplify | Sel. First | Sel. Only | Arg. Comp. | Abort Racer Request

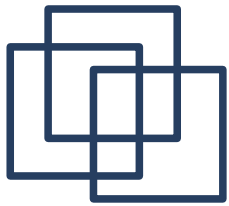
(Minnie Tom) has_pet.

It is asserted that Tom is a pet of Minnie („Role assertion“)

All Concepts | Cur. Concept | Sel. Concepts | All Inds. | Cur. Ind. | Sel. Inds. | All Roles | Cur. Role | Sel. Roles
 Concept A | Role A | Same As A | Different From A | Attribute A | Constraint A | Annotation CA | Annotation RA

Clear Selection | Delete Selected

```
Info
[[7] ? (describe-role has_pet /home/mi.wessel/KBs/people+pets.owl)
[[7] > (has_pet
:synonyms
(has_pet)
:domain
person
:range
animal
:parents
(likes)
:children
```



Syntaxes

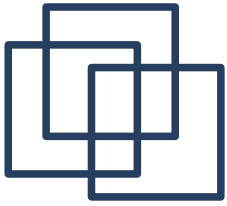
Racer can be used as a syntax converter

- Old lady concept in...
 - **KRSS / Racer native:**

```
(equivalent old_lady (and person female elderly))  
(implies old_lady (and (all has_pet cat)  
                        (some has_pet animal)))
```

- New: **OWL 2 Functional Syntax** (almost S-Expressions...)

```
EquivalentClasses(  
  old+lady  
  ObjectIntersectionOf(female person elderly))  
SubClassOf(old+lady  
  ObjectIntersectionOf(  
    ObjectAllValuesFrom(has_pet cat)  
    ObjectSomeValuesFrom(has_pet animal)))
```



Syntaxes (2)

- **Old lady** concept in **OWL RDF/XML**:

```
<owl:Class rdf:ID="old_lady">
```

(equivalent old_lady...

```
  <owl:equivalentClass>
```

```
    <owl:Class>
```

... (and ...

```
      <owl:intersectionOf
```

```
        rdf:parseType="Collection">
```

```
          <owl:Class rdf:about="#person"/>
```

```
          <owl:Class rdf:about="#female"/>
```

```
          <owl:Class rdf:about="#elderly"/>
```

```
        </owl:intersectionOf>
```

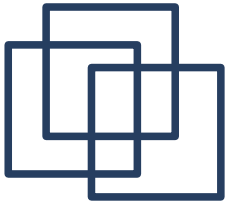
... person female elderly ...

```
      </owl:Class>
```

```
    </owl:equivalentClass>
```

...

...))



Syntaxes (3)

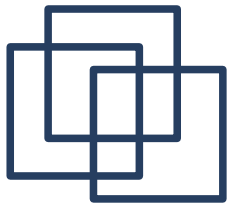
- **Old lady** concept in **OWL RDF/XML** continued

```
...  
<rdfs:subClassOf> ... (implies old_lady ...  
  <owl:Class>  
    <owl:intersectionOf rdf:parseType="Collection">  
      <owl:Restriction>  
        <owl:allValuesFrom rdf:resource="#cat"/>  
        <owl:onProperty>  
          <owl:ObjectProperty rdf:about="#has_pet"/>  
        </owl:onProperty>  
      </owl:Restriction>  
      <owl:Restriction>  
        <owl:onProperty>  
          <owl:ObjectProperty rdf:about="#has_pet"/>  
        </owl:onProperty>  
        <owl:someValuesFrom rdf:resource="#animal"/>  
      </owl:Restriction>  
    </owl:intersectionOf>  
  </owl:Class>  
</rdfs:subClassOf>  
</owl:Class>
```

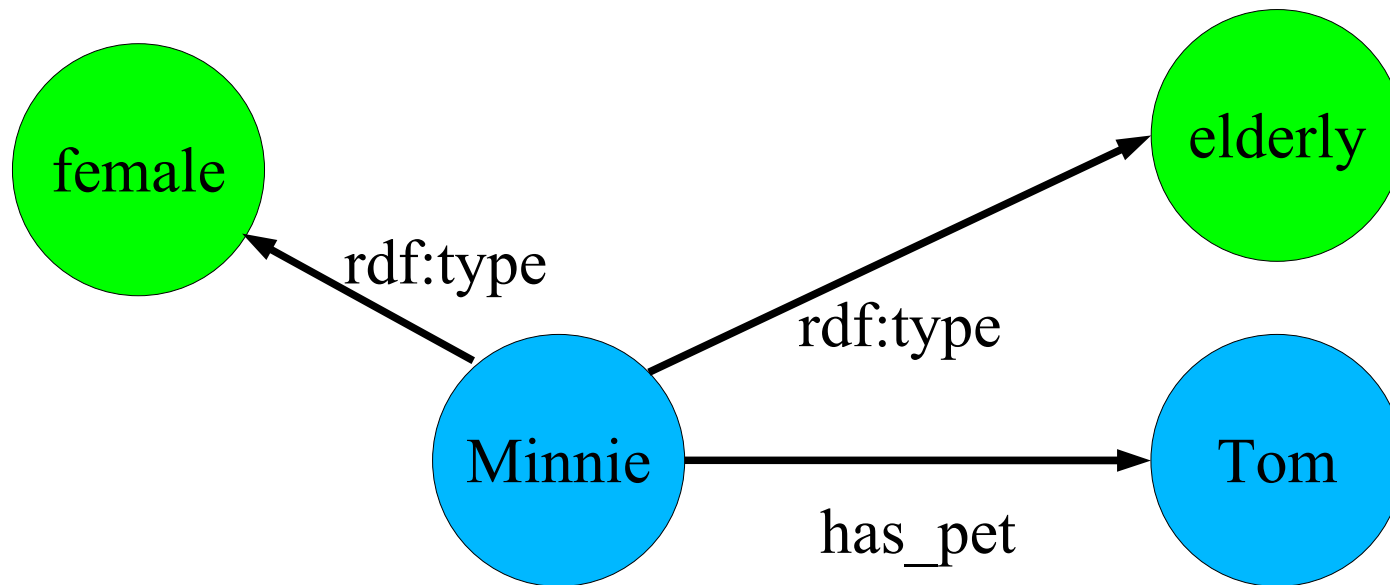
... (and ...

... (all has_pet cat) ...

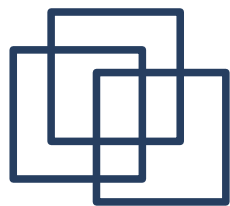
... (some has_pet cat) ...



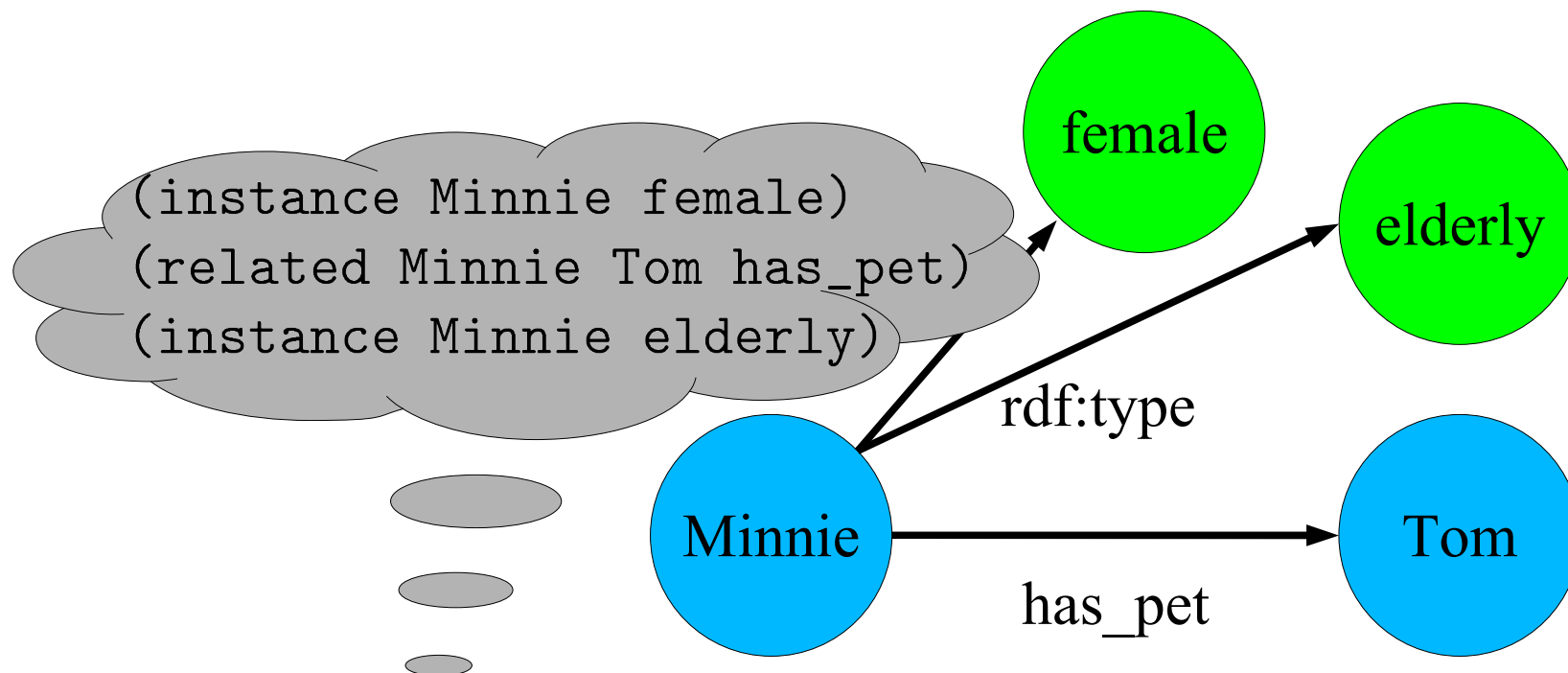
ABox Part of an OWL Ontology - RDF Graph



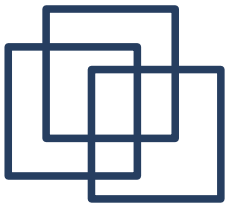
```
<rdf:Description rdf:ID="Minnie">  
  <has_pet rdf:resource="#Tom"/>  
  <rdf:type rdf:resource="#elderly"/>  
  <rdf:type rdf:resource="#female"/>  
</rdf:Description>
```



ABox Part of an OWL Ontology - RDF Graph



```
<female rdf:ID="Minnie">  
  <has_pet rdf:resource="#Tom"/>  
  <rdf:type rdf:resource="#elderly"/>  
</female>
```

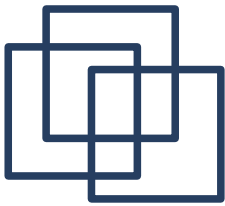


W3C RDF Query Language - SPARQL

```
prefix pets:
  <http://cohse.semanticweb.org/ontologies/people#>
select ?x ?y
where { ?x rdf:type pets:old_lady ;
        pets:has_pet ?y .
        ?y rdf:type pets:cat .
}
```

SQL-like
syntax

```
(define-prefix "pets"
  "http://cohse.semanticweb.org/ontologies/people#")
(retrieve (?x ?y)
  (and (?x #!pets:old_lady)
    (?x ?y #!pets:has_pet)
    (?y #!pets:cat)))
```



SPARQL with RacerPro Demo

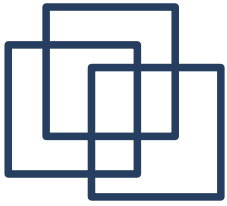
The screenshot shows the RacerPorter application window. The 'Query IO' tab is active, displaying a SPARQL query in the RacerEditor. The query is:

```
(sparql-answer-query "prefix pets: <http://cohse.semanticweb.org/ontologies/people#> select ?x ?y where { ?x rdf:type pets:old_lady ; pets:has_pet ?y . ?y rdf:type pets:cat . }")
```

The query is being evaluated, and the results are shown in a table at the bottom of the window:

No.	Variable	Binding
1	?x	#!:Minnie
1	?y	#!:Tom

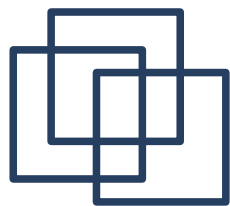
A blue thought bubble points to the query editor with the text: "SPARQL queries can be evaluated from the Listener or the editor, also in native Syntax".



Some Comments on SPARQL...

- SPARQL was not meant as an OWL query language
 - does it consider **inferred triples** (`rdf:type`? inferred properties?)
 - can't retrieve old ladies
 - **no negation as failure, no universal quantification, no aggregation**
 - most of our example queries cannot be formulated
 - as a rule language: has `construct`, but **cannot create new URIs**
- SPARQL in Racer, **2 modi**:
 - 1**: use AllegroGraph SPARQL processor (filled by Racer with triples)
 - scalable, secondary memory, ..., but only shallow inference
 - 2**: translated into nRQL query (uses AllegroGraph SPARQL parser)
 - full OWL reasoning, but not so scalable, SPARQL subset only

compromiss: let RacerPro **materialize** the inferred triples in AllegroGraph, then use mode 1 for SPARQL query answering

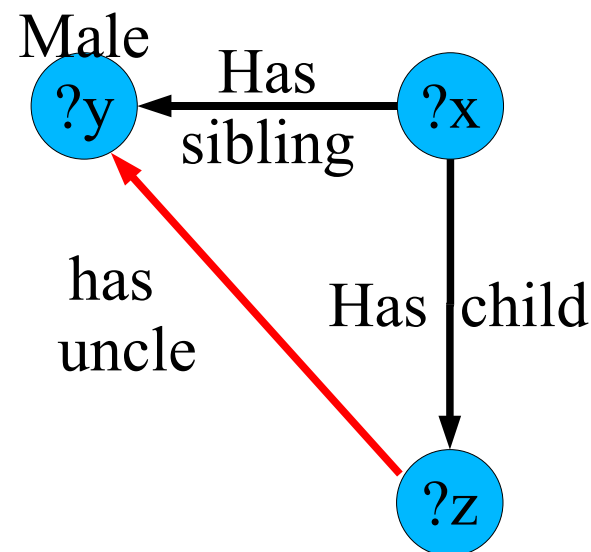


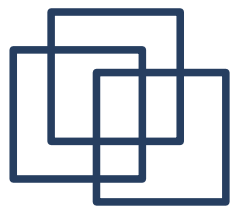
W3C Semantic Web Rule Language - SWRL

- Motivation: enhanced **relational expressivity**
(certain relational structures can't be encoded with concepts)

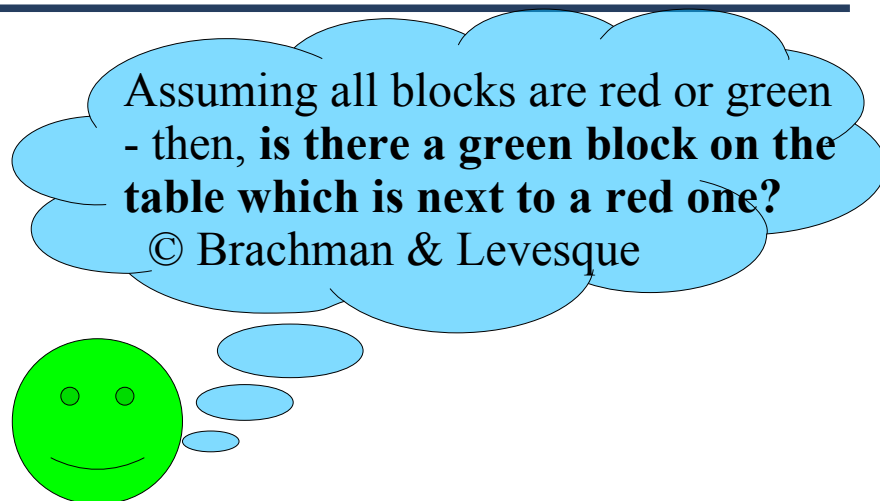
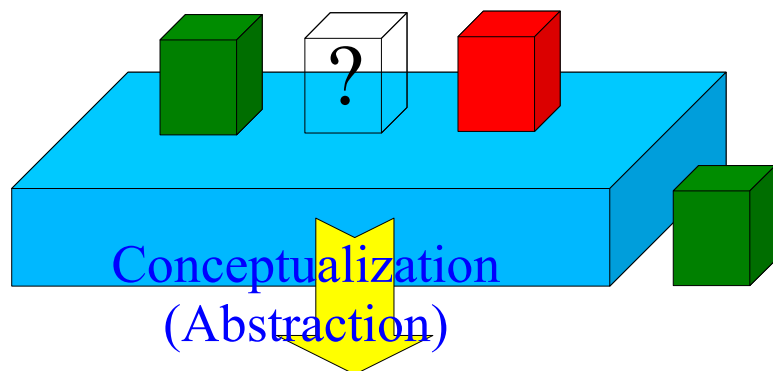
$$has_sibling(?x, ?y) \wedge male(?y) \wedge has_child(?x, ?z) \Rightarrow has_uncle(?z, ?y)$$

- Horn rules in RDF/XML syntax
 - Jess-based implementations
- undecidable, but decidable fragments
- Racer supports restricted subset of SWRL
 - translated into nRQL rules
 - nRQL rules need not be horn
 - and can construct new individuals
 - but have a non-logical semantics (similar to Jess)





ABox Queries & Indefinite Information



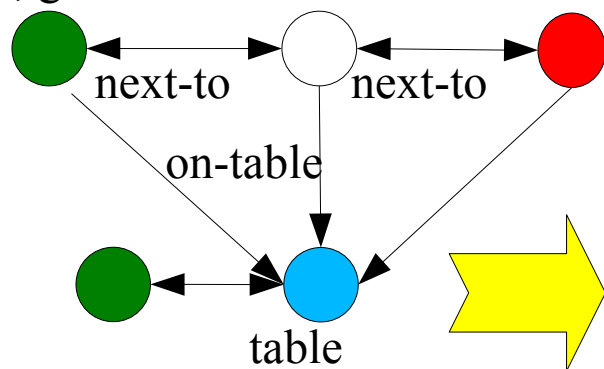
on-table, next-to, block, green, red, ...



$$\{ \text{block} \sqsubseteq \text{red} \sqcup \text{green}, \text{next_to} \doteq \text{next_to}^{-1} \}$$

block, green

$$\{ t : \text{table}, lb : \text{green} \sqcap \text{block}, rb : \text{red} \sqcap \text{block}, \\ mb : \text{block}, ob : \text{green} \sqcap \text{block}, \\ (lb, t) : \text{on_table}, (ml, t) : \text{on_table}, (rb, t) : \text{on_table}, \\ (gb, ob) : \text{next_to}, (ob, rb) : \text{next_to} \}$$

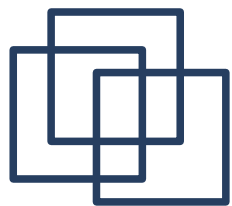


Problem Solving

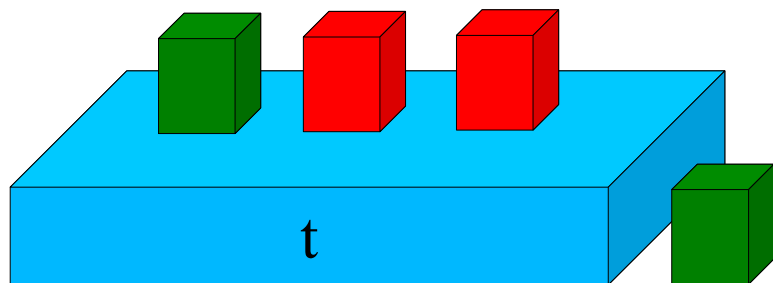
Ask for instances of the concept

$\text{table} \sqcap$

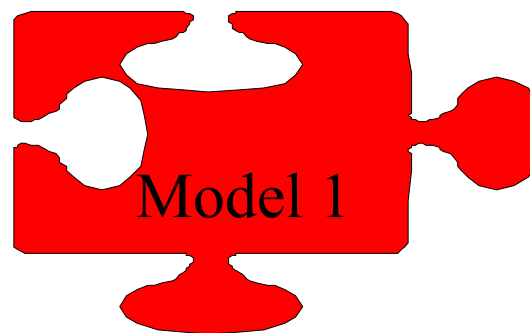
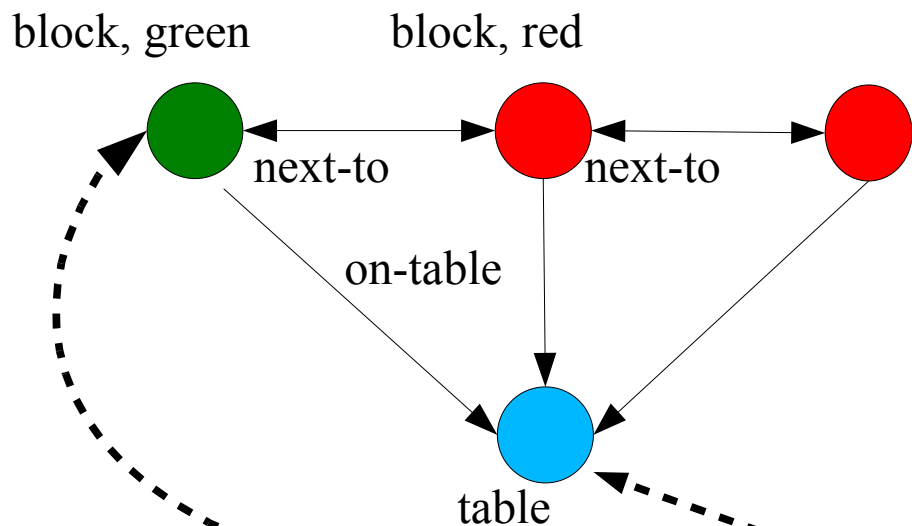
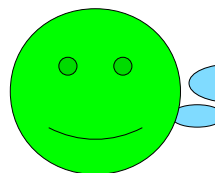
$$\exists \text{on_table}^{-1}. (\text{block} \sqcap \text{green} \sqcap \\ \exists \text{next_to}. (\text{red} \sqcap \text{block}))$$



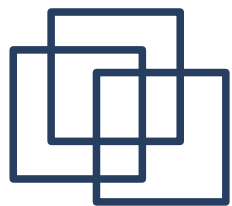
ABox Queries & Indefinite Information (2)



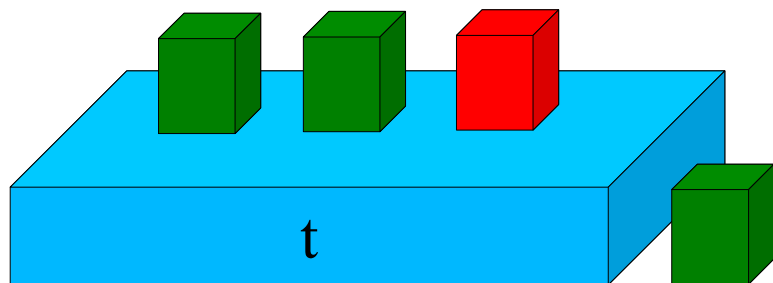
There are two possibilities.
If the middle block is red,
then the green left block is
next to a red one. But...



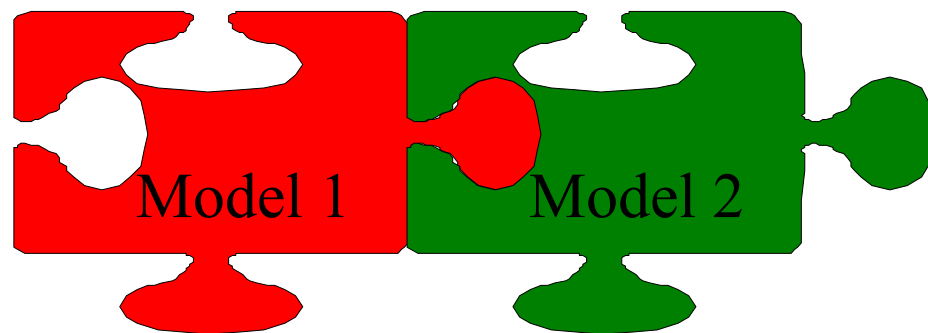
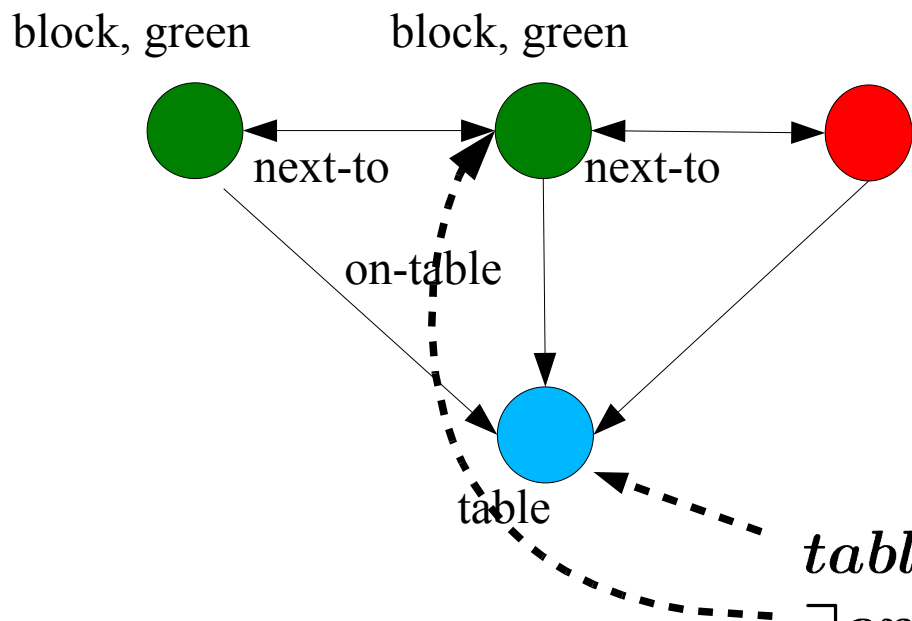
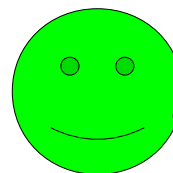
$table \sqcap$
 $\exists on_table^{-1}. (block \sqcap green \sqcap$
 $\exists next_to.(red \sqcap block))$



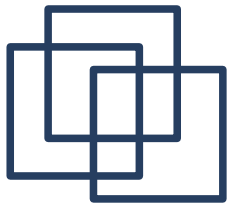
ABox Queries & Indefinite Information (3)



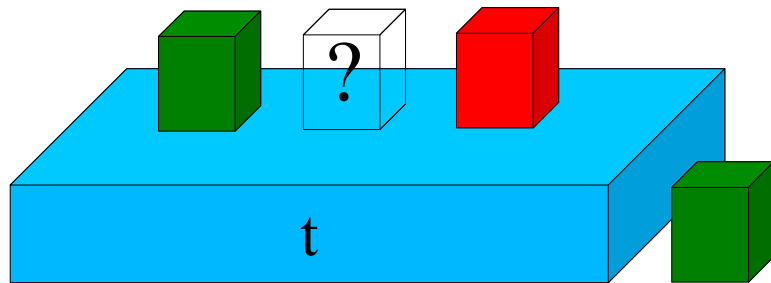
... if the middle block is green, then it is also next to the red block – so YES, there always is such a block!



$table \sqcap$
 $\exists on_table^{-1}. (block \sqcap green \sqcap$
 $\exists next_to.(red \sqcap block))$



ABox Queries vs. Database Queries

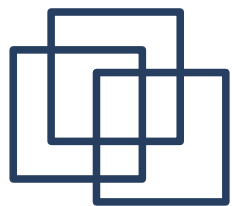


$\text{concept_instances}(\text{table} \sqcap \exists \text{on_table}^{-1}. (\text{block} \sqcap \text{green} \sqcap \exists \text{next_to}.(\text{red} \sqcap \text{block}))) = \{t\}$

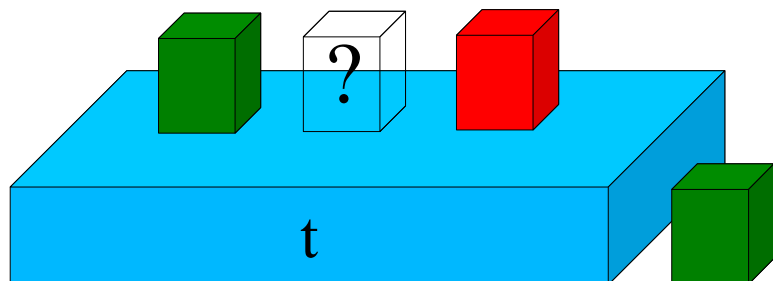
However:

$\text{concept_instances}(\text{block} \sqcap \text{green} \sqcap \exists \text{next_to}.(\text{red} \sqcap \text{block}))) = \{ \}$

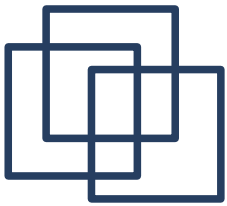
- Unlike DB queries, instance retrieval queries can cope with
 - incomplete information (have to perform **case analysis**)
 - have to **consider ALL models**, not only one („model = DB“)
 - only the **existence** of such a block is entailed



More Expressive Queries: Conjunctive ABox Queries


$$\mathit{ans}(\?x) \leftarrow \mathit{table}(\?x), \mathit{on_table}(\?y, \?x), \\ \mathit{block}(\?y), \mathit{green}(\?y), \\ \mathit{next_to}(\?y, \?z), \mathit{red}(\?z), \mathit{block}(\?z).$$

- Answer should be: $\?x = t$
 - most DL systems nowadays **return no answer**
 - decidability open until recently
- You can't retrieve $\?y$ because its binding can't be fixed
 - answer (head) variables & other variables



Solving Problems with Reasoning - Sudoku

Create a KB whose logical models represent all possible Sudoku solutions. A good Sudoku has only ONE solution \rightarrow entailed facts = solution!

$$\text{pairwise_disjoint}(C_1, C_2, C_3, C_4)$$

$$\top \sqsubseteq (C_1 \sqcup C_2 \sqcup C_3 \sqcup C_4) \sqcap$$

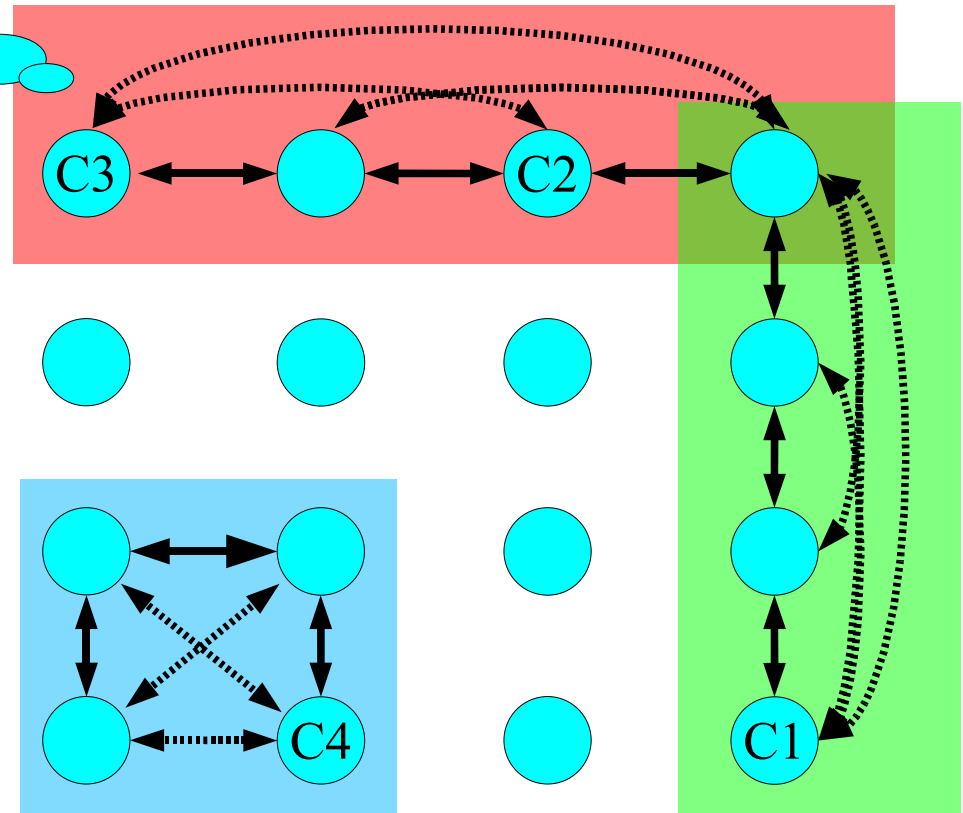
$$(C_1 \rightarrow \forall R. \neg C_1) \sqcap (C_2 \rightarrow \forall R. \neg C_2) \sqcap$$

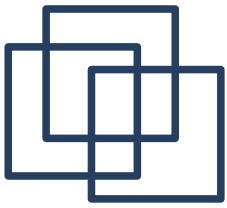
$$(C_3 \rightarrow \forall R. \neg C_3) \sqcap (C_4 \rightarrow \forall R. \neg C_4) \sqcap$$

$$\dots$$

3		2	
	4		1

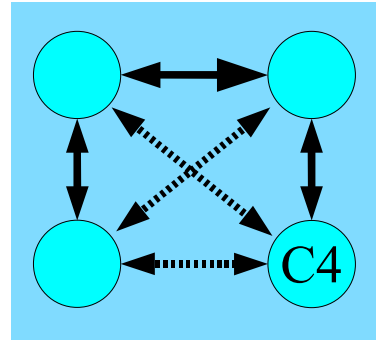
3	2		
			1
	4		



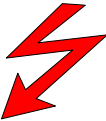


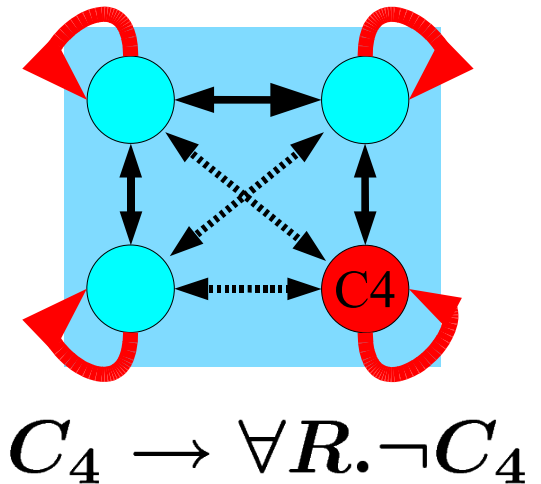
Sudoku – ABox Construction

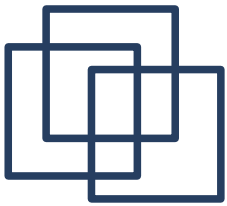
3		2	
	4		1



ABox construction

- by hand? OK for 4x4, but for 9x9?
→ create the structure programmatically (MiniLisp)
- transitive & symmetric property → 
 - use different backward property instead of a symmetric property
 - quantification over common parent property





Sudoku – Relational Structure

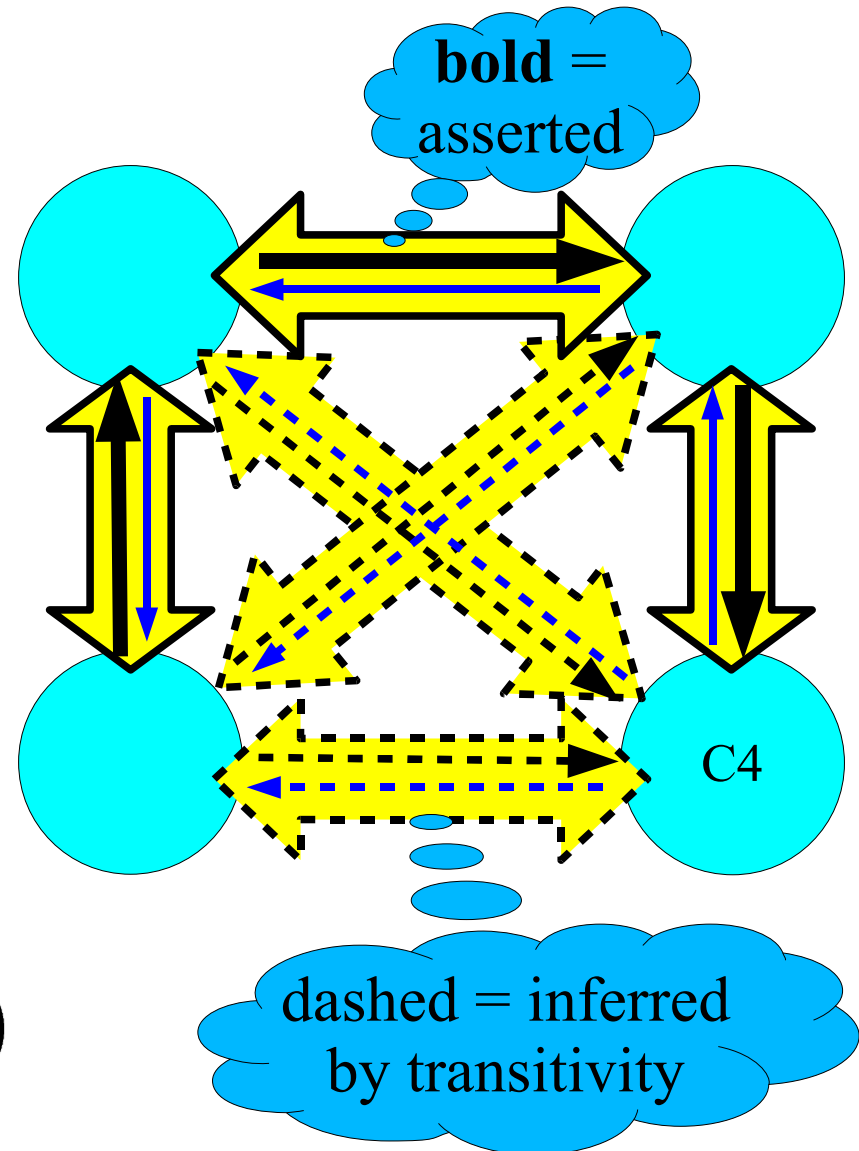
3		2	
	4		1

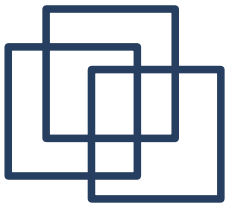
— $Q_1 \dot{\subseteq} R$ —
 — $Q_2 \dot{\subseteq} R$ —

transitive(Q_1)

transitive(Q_2)

$$Q_1(x, y) \leftrightarrow Q_2(y, x)$$





Solving Sudokus with Racer Reasoning!

MiniLisp
for programmatic KB
(here: ABox) creation, and
output generation.
New „ad hoc“ server
functions can be defined in
MiniLisp.

The screenshot shows the RacerPro 2.0 interface. The main window is titled "RacerPorter" and has a menu bar with "Profiles", "Shell", "TBoxes", "Axioms", "Taxonomy", "Role Hierarchy", "ABox Graph", "Query IO", "Queries + Rules", "Def. Queries", "Log", and "About". The "Shell" tab is active, showing a MiniLisp prompt. The prompt shows a query for a Sudoku solution, followed by a grid of numbers and a message indicating the evaluation took 0.2820 seconds. The result is "NIL".

The "RacerEditor" window is open, showing a MiniLisp script. The script defines two server functions: "sudoku-web" and "sudoku". The "sudoku" function is used to solve a Sudoku puzzle. The script also includes comments in red: "Register new server functions...", "... and use them!".

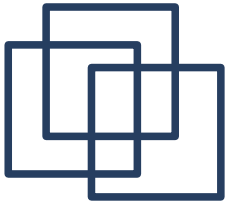
```
(terpri)
(format t "~%Sorry, no solution"))
(t (format t "~%Bad Sudoku.))))

;;;
;;; Register new server functions...
;;;
(server-function sudoku-web)
(server-function sudoku)

;;;
;;; ... and use them!
;;;

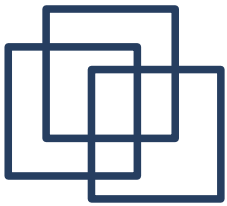
time
(SUDOKU ((0 6 0 5 0 3 2 0 8)
(1 0 5 0 0 8 0 0 3)
(8 0 0 0 0 6 4 1 0)
(9 0 0 0 0 1 0 4 0)
(0 7 0 3 0 4 0 8 0)
(0 8 0 7 0 0 0 0 9)
(0 1 8 4 0 0 0 0 6)
(2 0 0 8 0 0 7 0 1)
(7 0 6 1 0 5 0 3 0))))

Finished evaluating
RacerPro 2.0 running on localhost:8088 (case: preserve) ---- s
```

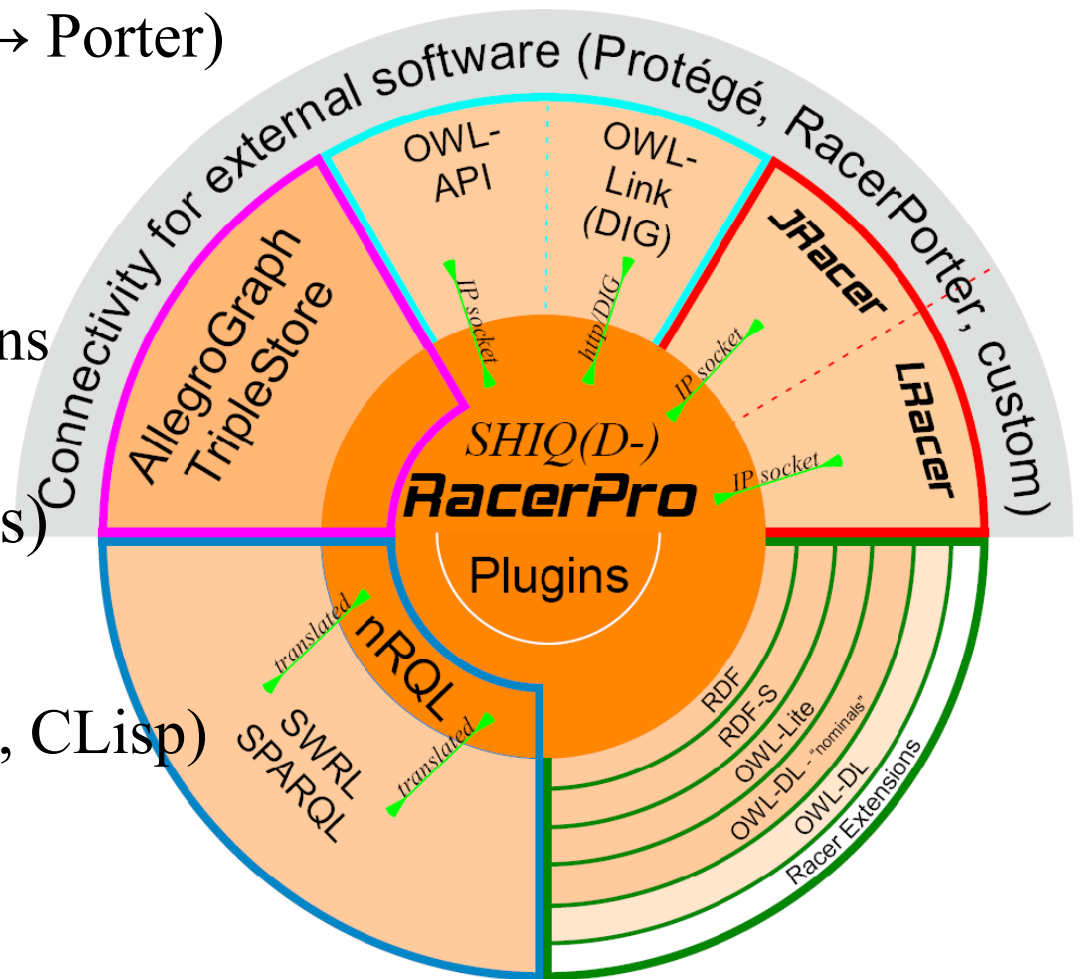
End of Presented Material...

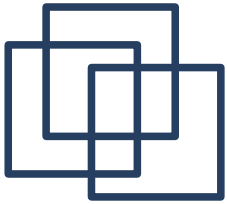
... due to a lack of time I couldn't present the remaining material but I am including the slides here anyway.



„SemWeb“ Development with Racer

- RacerPro is a server: 2 sockets / ports
 - 8088 TCP Lisp syntax (→ Porter)
 - 8080 HTTP XML (DIG)
 - file IO
 - approx 1000 API functions
- RacerPro remote access libraries (sets of stubs)
 - **LRacer** for Lispers (ACL, Lispworks, SBCL, CLisp)
 - **JRacer** for Java
 - unicode (UTF8)
- DIG, OWLlink, OWLAPI

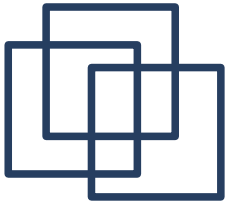




LRacer for Lispers

```
(enable-lracer-read-macros)
(full-reset)
(define-prefix "people"
  "http://cohse.semanticweb.org/o
(owl-read-file "people+pets.owl")
(abox-consistent?)
(taxonomy)
(concept-synonyms #!people:bottom)
(retrieve (?x ?y)
  (and (?x #!people:person)
    (?x ?y #!people:has_pet)))
(instance #!people:betsy #!people:ma
(abox-consistent?)
```

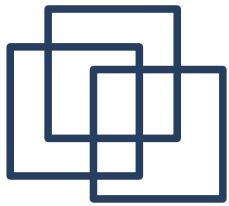
- Size: > 1000 API functions / macros
 - HAS to be generated automatically
- Some problems with UTF8 socket streams on different Lisps
- ACL „modern Lisp“
 - Racer is case sensitive „mlisp“
 - LRacer: maybe „alisp“
 - NIL ↔ nil
 - conversion required! ... but for which symbols? depends on packages!
- with-macros...



JRacer for Java Developers

```
RacerClient racer = new RacerClient(ip,port);
try {
    racer.openConnection();
    racer.fullReset$();
    racer.owlReadFile$(peopleAndPets);
    boolean consistent = racer.aboxConsistentP();
    RacerResult res2 = (RacerResult)
        racer.racerAnswerQuery$("(?x ?y)",
            "(and (?x #!:person) (?x ?y #!:has_pet))");
    if (res2 instanceof RacerSymbol) {
        System.out.println("No instances!");
    } else {
        for (RacerList<RacerList<RacerSymbol>> bindings :
            (RacerList<RacerList<RacerList<RacerSymbol>>>)res2) {
            for (RacerList<RacerSymbol> binding : bindings) {
                for (RacerSymbol varval : binding) {
                    System.out.println(varval);
                }
            }
        }
    }
}
```

- Automatically generated
- Strings or ArrayLists for S-Expressions → generics, structure iteration
- Typecasts and runtime checks not avoidable...
- UTF8
- Java ellipsis for &rest, &key
- overloaded methods for &optional a b ...
- with-... macros
- > 3000 Java methods

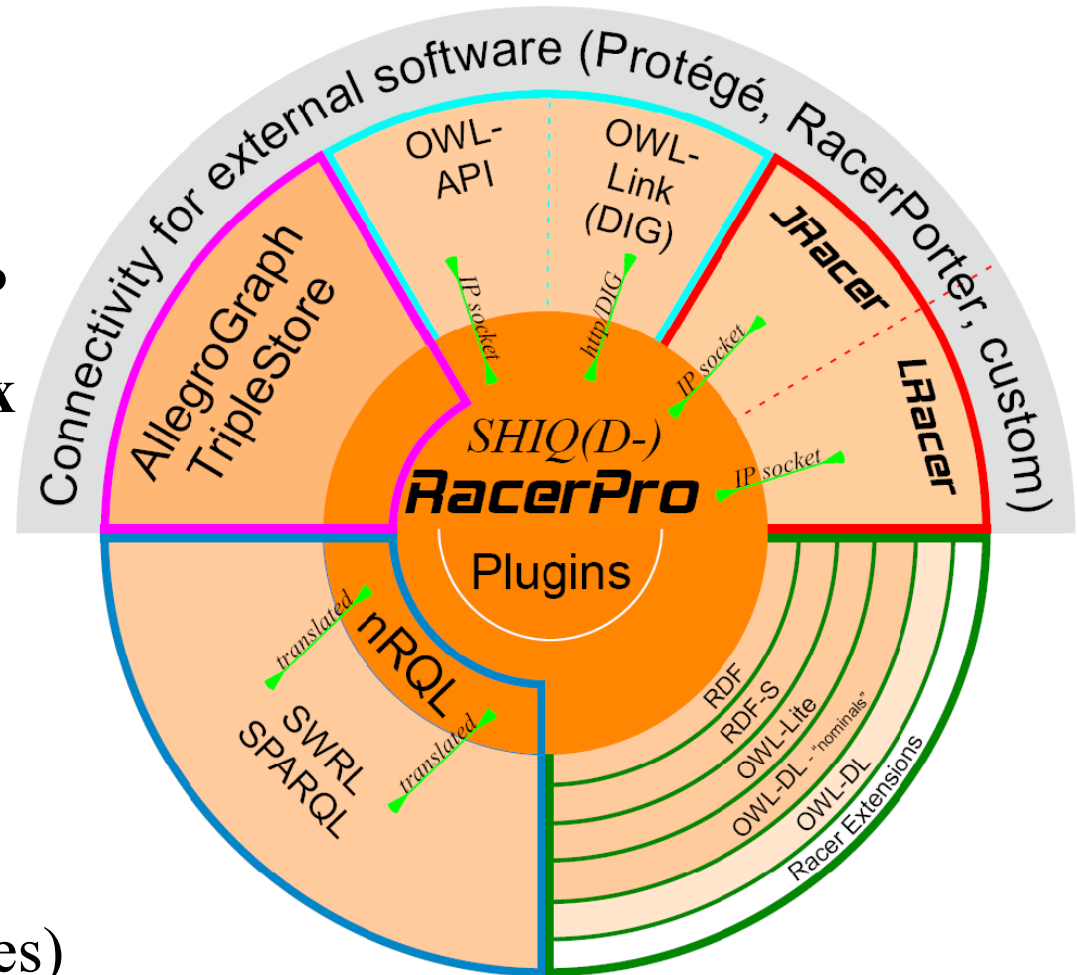


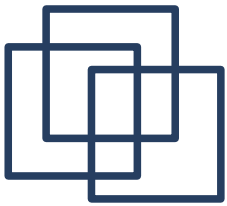
DIG & OWLlink – XML-over-HTTP APIs

- XML over HTTP-based
 - 8080 port of RacerPro
 - AllegroServe / CL-HTTP
 - DIG used by **Protégé 3.x** ontology editor

- OWLlink

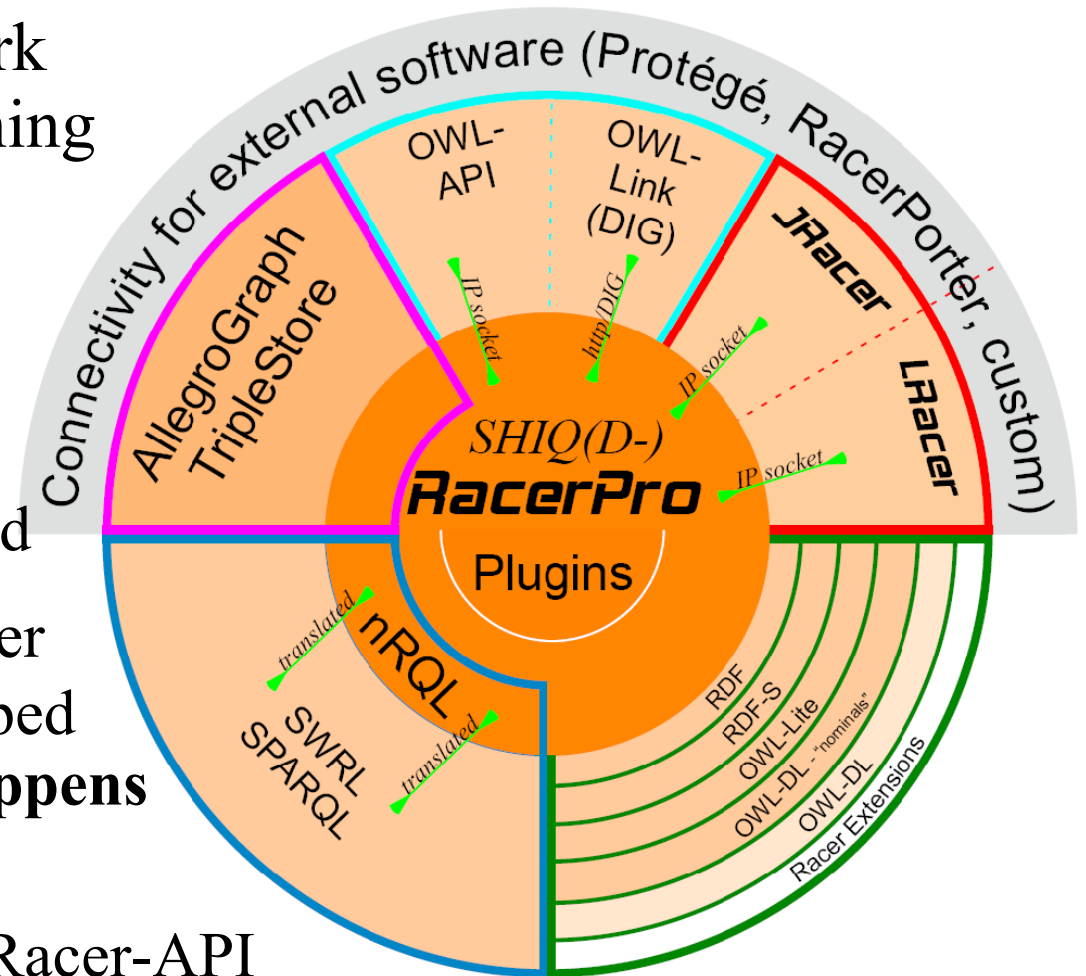
- successor of DIG
- we are developing an S-Expression over HTTP (instead of XML messages) binding for the protocol (idea: turn OWL functional syntax into S-Expressions)

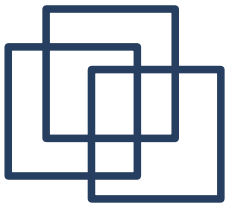




The OWLAPI Java „SemWeb“ Framework

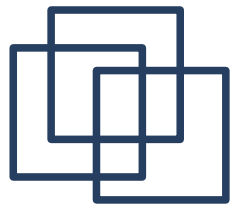
- An important Java framework for Semantic Web programming (similar to Jena for RDF, ...)
- Basis of **Protégé 4.x**
 - handles reasoner access
 - RacerAdapter required
 - RacerReasoner adapter & Protégé plugin developed and provided by **Olaf Noppens from Ulm University**
 - required an entirely new Racer-API (Racer-OWLAPI) in order to make the adapter work (have to support the core OWLAPI abstractions)





Graphical OWL2 Modeling with Protégé 4

The screenshot displays the Protégé 4 graphical OWL2 modeling interface. The main window shows the 'people' ontology. The left pane displays the 'Inferred class hierarchy' with the 'old_lady' class selected. The right pane shows the 'Class Annotations' for 'old_lady', including a 'comment' annotation and a 'label' annotation. Below this, the 'Rules' pane is empty. The 'Description: old_lady' pane shows the class description, including equivalent classes (elderly and female and person), superclasses (has_pet some animal and has_pet only cat), and inferred anonymous superclasses (eats some Thing).



Graphical OWL2 Modeling with RacerPorter

The screenshot displays the RacerPorter application interface. The main window has a menu bar with options: Profiles, Shell, TBoxes, ABoxes, Concepts, Roles, Individuals, Assertions, Axioms, Taxonomy, Role Hierarchy, ABox Graph, Query IO, Queries + Rules, and Def. Querie. The 'Axioms' menu is currently selected.

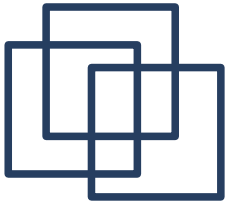
The main workspace shows a list of axioms with columns for ID, Axiom Type & Attribute Type, and a description. A table of axioms is visible:

ID	Axiom Type & Attribute Type
191	SubClassAxiom
191	SUB-CLASS
191	SUPER-CLASS
192	DisjointClassesAxiom
192	DESCRIPTIONS
196	EquivalentClassesAxiom
196	DESCRIPTIONS
>>> 197 <<<	SubClassAxiom
>>> 197 <<<	SUB-CLASS
>>> 197 <<<	SUPER-CLASS
199	EquivalentClassesAxiom
199	DESCRIPTIONS

An 'Edit SubClassAxiom (ID: 197)' dialog box is open, showing the configuration for a sub-class axiom. The dialog includes fields for Concept Name (set to '#!:old_lady'), Role (set to '#!:drives'), and Concept Constructors (set to 'and'). The Concept Description for the Attribute SUPER-CLASS of Axiom 197 is '(and (all #!:has_pet #!:cat) (some #!:has_pet #!:animal))'. The dialog also shows the Concept Name 'bottom' and Role '#!:drives' for the super-class.

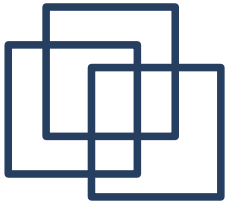
The bottom status bar shows the following information:

```
[1] ? (owl-read-file "z:/KBs/people+pets.owl" :maintain-owlapi-axioms t)
Reading ontology z:/KBs/people+pets.owl...
Duplicate definition (not (or (or (http://cohse.semanticweb.org/ontologies/people#part_of|http://cohse.semanticweb.org/ontologies/people#plant)))) for (http://cohse.semanticweb.org/ontologies/people#part_of)
Reading ontology z:/KBs/people+pets.owl done.
[1] > z:/KBs/people+pets.owl
```



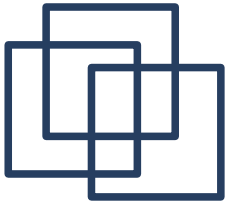
RacerPro Extensibility

- MiniLisp
 - ad-hoc server extensions (API function missing?)
 - executed on the server (no communication latency → fast)
 - user-defined query predicates
 - report generation, programmatic knowledge base creation („Sudoku Grid“), „Racer scripting“
- Plugin mechanism
 - create a FASL file with AllegroExpress, convert it into a plugin
 - server extensions possible (hook mechanism)
 - „at our own risk“, full access to Racer internals,
 - faster & more general than MiniLisp, but not „ad hoc“



MiniLisp in a Nutshell

- A simple „**Lisp 1 Lisp in Lisp**“ (own evaluator)
- Motivation: termination safe & simple (important if used in queries!)
 - **total recursive functions**
- Basic datatypes and operations (both borrowed from CL) for
 - lists, numbers, symbols, characters, booleans, basic IO streams
 - no cyclic lists
- Control structures (mostly borrowed from CL)
 - **bounded loops**, structure mapping (no cyclic lists)
`dotimes`, `dolist`, `maplist`, `maptree`, ...
 - `if`, `when`, `unless`, `cond`, ...
- `defun` and `defpar`, `defcon`
 - recursion always aborted at runtime (stack inspection)

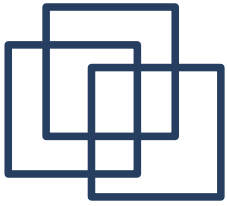


MiniLisp in a Nutshell (2)

- No macros
- All RacerPro API functions / macros callable (macros treated as functions)
- `setq` (`incf`, `decf`), but no generalized variables (prevent cyclic lists)
- No closures
 - impossible

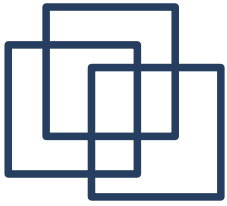
```
((lambda (x x) (x x)) (lambda (x x) (x x)))
```
 - (built-in) higher order functions have to be special forms

```
(maplist (lambda (x) (1+ x)) '(1 2 3))
```
- `evaluate`
 - but no `(evaluate ... (evaluate ...) ...)`
- Quote, backquote, ...
- **Claim: covers 99,99% of the typical „Racer programming“ cases!**



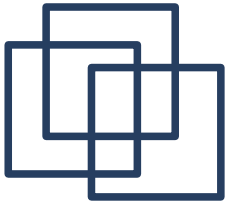
Benefits of Lisp

- Racer could have been implemented in another language...
 - ... but some Lisp-features are especially valuable here
 - „standard arguments“ (GC, closures, ...) apply to many languages nowadays (Haskell, Python, Ruby, JavaScript, F#, ...)
- Merits of functional programming
 - Racer tableaux prover (= system core / kernel) was implemented in a functional style (seems natural)
 - good for debugging
 - but problems with stack size someday
 - switched to closures for representation of backtracking context („continuation-passing style“)
 - implementation didn't break although this was a drastic change in the system architecture → flexibility of Lisp



Benefits of Lisp (2)

- Ability to concisely represent and conveniently manipulate complex expression
 - **structured literals**
 - you don't want
`expr.add(new this („a“).add(new that („b“))) ;`
 - `ArrayLists : [„a“, „b“]`
 - S-Expressions were invented for symbolic computation
→ perfect
 - S-Expressions for the **front-end syntax**
(things get encoded later)
 - ... **LOTS of operations deal with front-end syntax only**

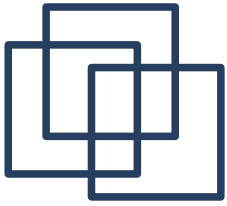


Benefits of Lisp (3)

- **Abstraction**

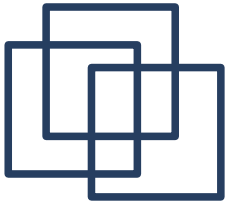
- W3C standards such as OWL2 are still a moving target
 - a very flexible basis is needed / prototyping
 - decouple implementation from standardization ;-)
 - transform OWL (SPARQL, SWRL, ...) into that representation (but keep the original representation)
- Lisp allows you to **defer decisions**
 - no static typing (no extensive „type“ or class hierarchy refactorings)
 - no tight data structure (class hierarchy) / operations coupling
 - operations can be combined in a more flexible way
 - macros can save you a LOT of refactoring time (change the macro, keep the code!)
 - „open“ method / function signatures / delegation chains:

```
(defun f1 (... &rest args &allow-other-keys)
  ... (apply #'f2 args)))
```



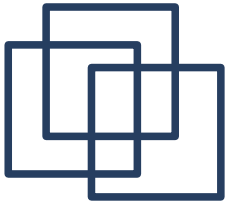
Benefits of Lisp (4)

- **Reflexive / introspective qualities** of Lisp
 - meta-information is always therewithin the SAME environment (→ synergy effect), e.g.
 - `racer-defun` does many things in one place:
 - registers the function for the server listener
 - creates LRacer & JRacer stubs based on lambda list
 - creates code for RacerPorter to support completion, ...
 - `defowlaxiomclass`
 - creates the axiom editor CAPI dialogs for Porter by „inspecting“ its slots, conjoining appropriate CAPI code for the different attributes
 - „data driven“ meta programming
- **Lisp allows a very small company (us!) to survive W3C Semantic Web standards ;-)**



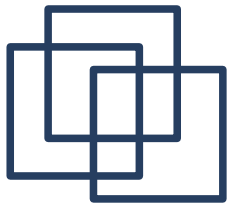
Drawbacks of Lisp

- Lack of „(quasi) standardized“ frameworks / solutions
 - and quality of the existing frameworks probably not as good as in the Java world (MUCH less developers are using them on a daily basis)
 - Java ↔ Lisp „in memory“ integration still very hard
 - e.g., very nice graph layouters in Java
 - Java developers get much more for free
 - Lispers have to work harder: more hand-crafted solutions
- Language too old (?)
 - unicode sockets, custom streams (e.g., gzipped streams), ...
- GC is a big plus, but very hard to control sometimes (for large KBs)



Exploited Lisp Frameworks

- CL-HTTP (John Mallery) / AllegroServe
 - owl-read-document (HTTP client)
 - owl:Import (downloads an ontology from the web)
 - DIG / OWLlink server
- Wilbur (Ora Lassila)
 - basic RDF processing
- AllegroGraph
 - SPARQL parser & triple store for RacerPro
- Lispworks CAPI and Lispworks editor
 - for RacerPorter / RacerEditor
 - thanks to Martin Simmons for great CAPI support



How do I get RacerPro ?

- www.racer-systems.com
there is the 2.0 preview version
 - no license required
 - to be finalized soon
- A recent research project which uses RacerPro:
www.boemie.org

Thanks !

The screenshot shows a web browser window displaying the Racer Systems website. The URL is <http://www.racer-systems.com/de/index.phtml?lang>. The page features the Racer logo and a navigation menu with links for Produkte, Downloads, Über uns, Aktuelles, Kontakt, and Impressum. A sidebar menu includes Home, Produkte, Services, Technologie, Unternehmen, and Website. The main content area prominently displays "RacerPro 2.0 pre-release version available now!". Below this, it states "RacerPro, der OWL Reasoner und Inference Server für das Semantic Web, ist jetzt als Vorschau-Version 2.0 verfügbar!". The text explains that the preview version allows users to see new functions planned for the official version, such as support for W3C standards and improved processing speed. It invites customers and users to test the preview version. A note mentions that the current official version is 1.9.0, but version 2.0 is planned for summer 2009. A search bar at the bottom contains the text "qual".