Racer Systems

Design Principles & Realization Techniques for User Friendly, Interactive and Scalable Ontology Browsing and Inspection Tools

Ralf Möller, Michael Wessel

Racer Systems GmbH & Co. KG

OWLED 07, June 6, 2007

Motivation – DLS OBITs

- Many ontology tools primarily focus on authoring or visualization
- OBIT $\neq$ Editor $\neq$ Ontology Displayer
  - different requirements per se, but
  - browsing and inspection (also) requires (graphical) visualization and querying
  - authoring functionality is a nice add on
- (W3C) standards are a necessity, but proprietary DL system functionality must be offered by an OBIT as well
- -> Reusable ideas behind RacerPorter

Motivation – Criticism (1)

- Today, most ontology tools ...
  - ***focus on XML syntax*** (which was invented for machines, not people)
    - hard to read, (almost) impossible to write
    - visualization and visual editing becomes unavoidable, but visual editing has drawbacks
  - don't offer textual ***interactive communication*** with reasoners
    - ad hoc queries and commands are needed
    - problematic due to XML again
  - > interactions mostly widget-based
    - either not general enough or too complicated
    - -> textual interactions needed

Motivation – Criticism (2)

- Plugin architectures are fine, but ...
  - plugins often don't know of each other
    - no coherent perspective and usage
    - no or bad ***information flow between plugins***
    - for complex ontology inspection tasks, ***results of several queries have to be combined!***
- Editors: too much emphasis on visual editing (caused by XML)
  - low bandwidth (experienced KRSS users are ***much*** faster textually, abstract OWL?)
  - no interactive and rapid editing possible
- Tools have scalability problems

OBIT Requirements (1)

- Based on the analysis / criticism
- To achieve high bandwidth textual interaction with a reasoner ...
  - add a ***shell*** with command and argument completion, command history, redo, ...
  - > enables complex, semantic ad hoc KRSS (and SPARQL) queries
- Visual ontology browsing & navigation
  - different visualizations (tree vs. graph, depth limit, graph/tree roots)
  - widget- / gadget-based interactions

OBIT Requirements (2)

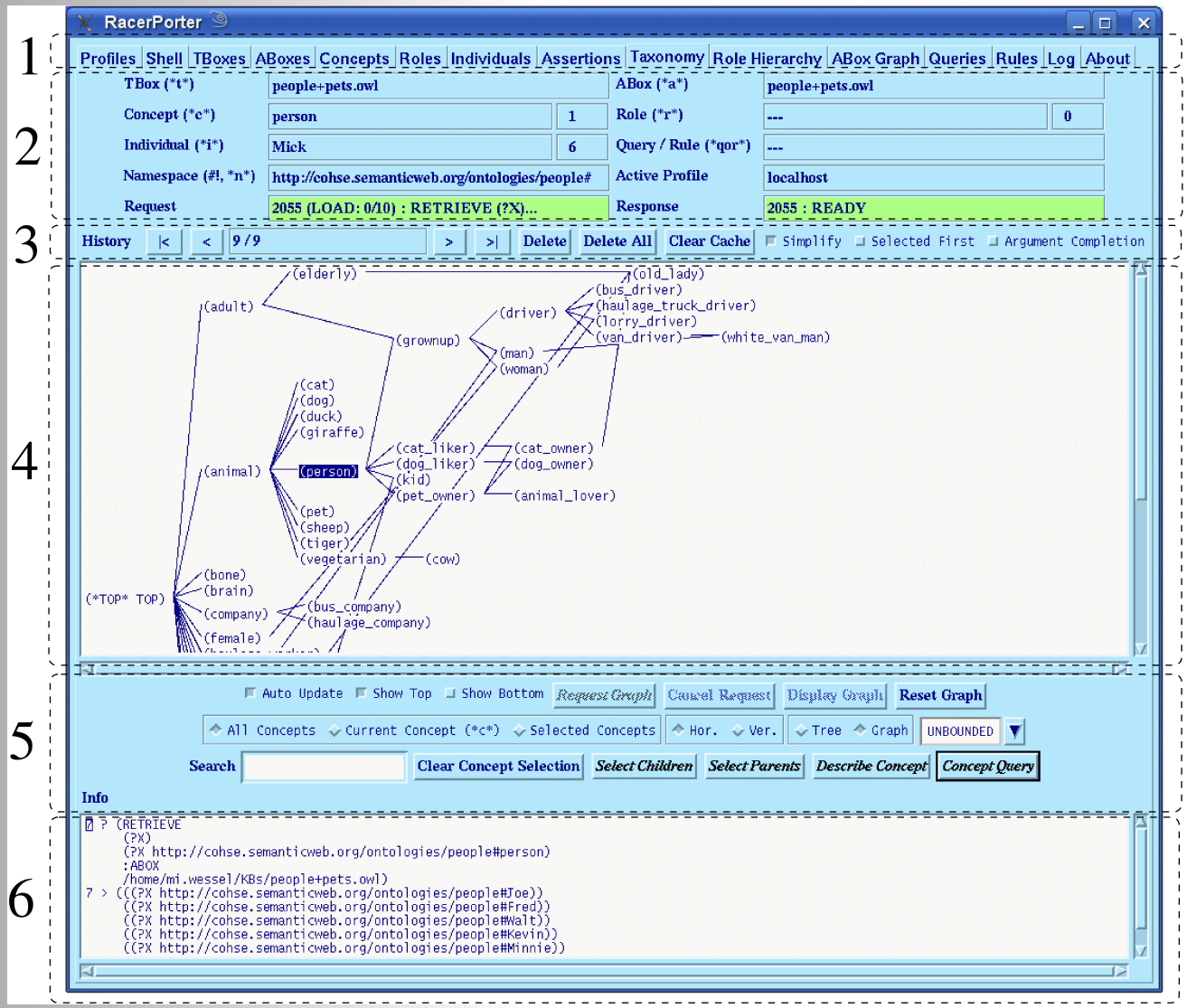
- Visualize different aspects of a DLS
  - Tbox, Abox, role hierarchy, queries, ...
  - ***different aspects*** shall be visualized using ***different views*** or perspectives, but ***interrelated and coherently***
  - how to realize the ***information flow?***
  - *how to incorporate the shell* and widget-based interactions and results produced by them ***into the information flow?***
- DL system specific functionality
  - RacerPro: nRQL query managment, server persistency facility, ...

RacerPorter

- Influenced by RICE © Ronald Cornet
- First released with RacerPro 1.8.0 in July 2005, has many users
- Designed according to requirements
- Tabbed interface
  - different tabs represent different aspects,
  - or the same aspects, but with different visualization modalities
- Revised extensively for next release
  - to solve scalability problems (cyc.owl)
  - many new features (SPARQL evaluation)

RacerPorter - GUI

Racer Systems

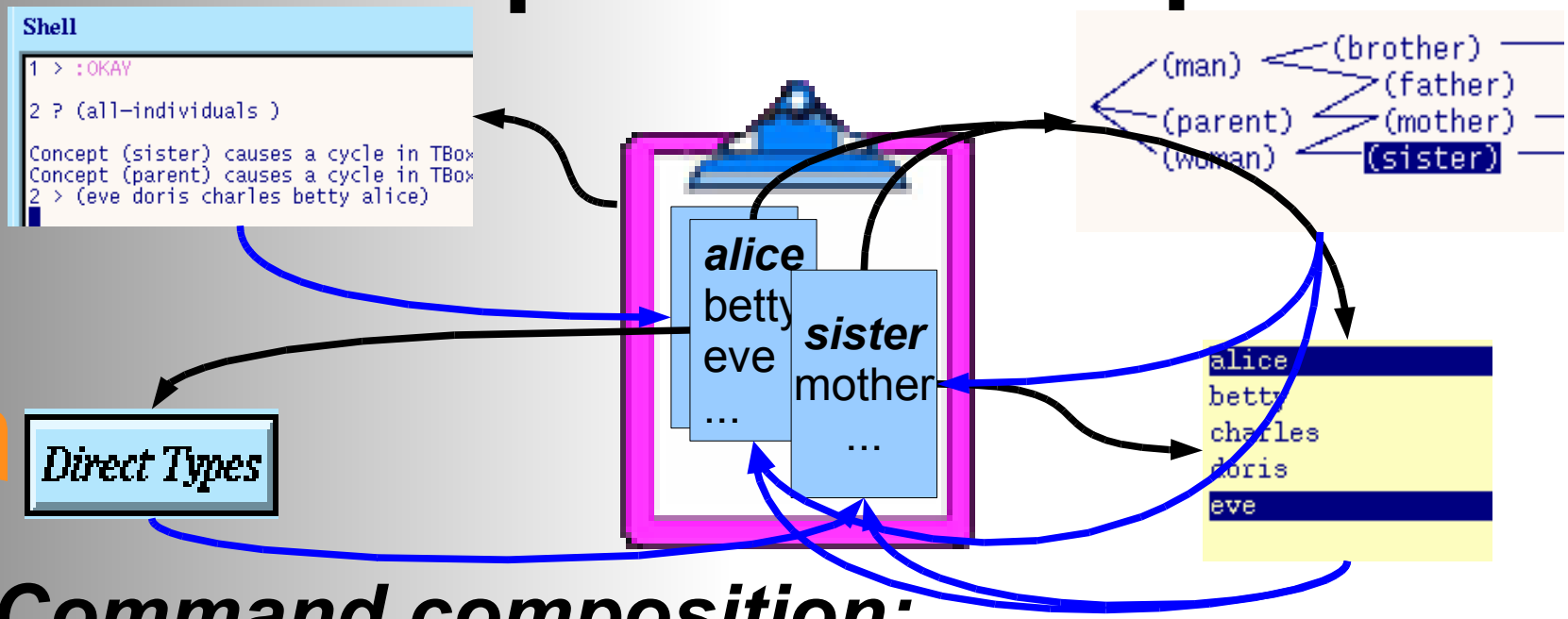


Information Flow in Porter

- Tabs show ***interrelated*** information
  - e.g., the ***taxonomy tab*** can only show the descendants of the concepts which have been selected in the ***list of concepts tab***
  - notion of current objects and state required
- ***(KRSS) commands*** can be executed
  - with the push of a button (-> current object)
  - via a mouse gesture (browse and click)
  - typed into the shell
- Commands ***require arguments and produce results***

Racer Systems

The Clipboard Metaphor



Command composition:

- `sel.-inds:=`**`all-individuals`**`(cur-abox)`
- `show-list(sel.-inds)`
- `cur.-ind:=select-w-mouse(sel.-inds)`
- `sel.-concepts:=`**`direct-types`**`(cur.-ind)`
- `show-taxonomy-fr-roots(sel.-concepts)`

Racer Systems

Focus Control & Navigation

- The clipboard is also for focus control
  - in general, there is one focus per tab
  - focus on ***current*** or ***selected objects***
  - navigation history, VCR navigation buttons
    - reestablish previous focus effortless
    - -> very complex navigation history required
- "Drill down"-like browsing
  - if mouse click changes `cur.-concept` and `show-taxonomy-fr-roots(cur.-concept)` is requested and redrawn automatically
  - automatic redrawing can be problematic

Other Features

- ***Emacs-compatible editor*** with buffer and expression evaluation mechanism
  - also linked with the shell
  - KRSS, OWL, SPARQL
- Other new features:
  - query result inspector
  - support for controlling (starting, stopping, setting options of) RacerPro servers
  - multiple sessions in parallel
  - much better OWL support (abbreviates XML namespaces using the `#!` prefix)
  - mostly asynchronous (non blocking) GUI

Lessons Learned

- Use uniform and system wide metaphors and mechanisms
- A good metaphor can address more than one problem
  - e.g., information flow and focus control
- Expect that your graph drawers will fail
  - ***Cancel & Retry*** mechanisms are needed, e.g., focus on certain nodes and retry with different display and/or focus options
- Expect large results (don't put 1.000.000 individuals in the shell without asking the user, ...)

Lessons Learned (2)

- Socket-based communication has problems
  - strings can become too long
  - heavyweight caches are needed
- Don't block the interface if possible
  - avoid the looks like dead syndrome
  - use threads (+ cancel becomes possible)
- Check your data structures for scalabilty
- Give control (regarding display focus and display update options) to the user

Future Work

- Internalization issues
  - unicode / japanese characters in KRSS
- Explanation facilities
- Abortable individual RacerPro requests
  - maintain “process browser“-like list view of currently active requests
- Better / different Abox visualizations
  - currently, ***unraveling*** is used
  - no cycles can be displayed
- ***Some*** graphical authoring?

Racer Systems

Thank You!

If you are interested - see our demo in the Posters & Demos session!