PROGRAM ANALYSIS AND VERIFICATION

POINTER PROGRAMS IN C

L.GEORGIEVA

HERIOT WATT UNIVERSISTY

ANALYSIS OF POINTER PROGRAMS

Pointer programs	• Common in legacy code.
Context	• Any program that manipulates linked data structures: e.g. linked lists, trees.
Why should this problem be solved?	• Error prone. Costly refactoring.
How will I know this problem has been solved?	• Verification versus testing.

BACKGROUND INFORMATION

Research into modelling linked data structures in relational logics

- Description logics
- Purposefully designed languages
- Separation logic

Experiments with tools

- Racer
- Spass
- Bohne
- Otter

WORKABLE SOLUTIONS

Modelling in DL

- DL reasoners are fast.
- The language is intuitive
- The most serious issue: reachability. DL languages are not expressive enough to handle this automatically.

First order logic

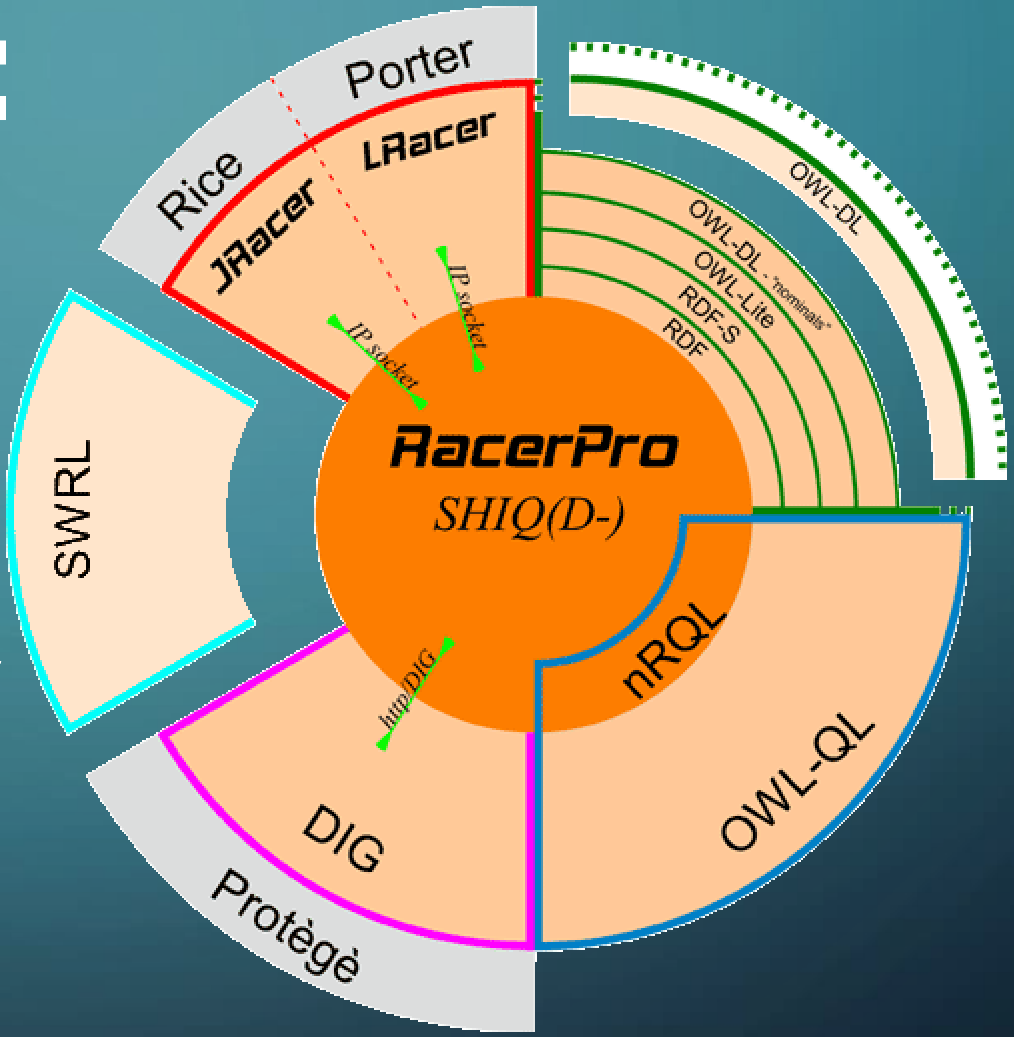
- First order logic reasoners are powerful.
- Steep learning curve.
- Issues with decidability: one approach is to restrict the language to fixed number of variables only.

Separation Logic

- Intuitive approach.
- Relates the heap to the data structure directly.
- Limited automated reasoning support.

THE PROTOTYPE

- We used RACER in a case study.
- Allows for automated reasoning with number restrictions and reachability
- Not all interesting features are expressible (i.e. role reversal)



OUR EFFORTS

- Modelling of linked data structures.
- Limited support of sharing (graphs, trees).
- Case studies: we considered basic linked lists, cyclic linked lists, skipped linked list, binary tree.
- Red black tree: issues with complexity.

EXTENDING THE WORK

- Currently looking at alternative reasoners
  - Support for SHOIQ
  - Support for more expressive languages (e.g. FOL).
  - Different data structures and programs with different sizes
- Sharing is a challenge.

PROGRESS

- Limited features captured.
  - Standard linked data structures modelled successfully
    - Fully automated support for linked lists like data structures.
- Automated reasoner all the necessary features (support for transitive closure) does not exist.
- Alternative frameworks.