LARA Tutorial

4. Weaving MATLAB

Tiago Carvalho, Pedro Pinto, João Bispo, Ricardo Nobre, Luís Reis, and João M.P. Cardoso

University of Porto, FEUP, Porto, Portugal

April 20th, 2016

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

ANTAREX 10^{18}

SPeCS Special-Purpose Computing Systems, languages and tools

Objectives

- Introduction to MATISSE

- Using LARA to analyse and modify MATLAB programs

What is MATISSE?

- MATISSE: Framework to analyse and compile MATLAB code;
  - MATLAB: High-level programming language based on matrices
  - Used in scientific, engineering and finance domains.
- MATISSE has two main features:
  - MATLAB Weaver: LARA for MATLAB
  - Compilation to C and OpenCL code
- We'll focus on LARA for now

LARA for MATLAB

- We can apply LARA aspects to MATLAB files.
- All we have to do is switch weavers
- Note that MATLAB and C are different, so the joinpoints sometimes differ.

http://specs.fe.up.pt/tools/matisse/

4.1 Target Language Report

- Goal: Report information regarding the target language
  - Join point: points of interest in the code
  - Attribute: information concerning a join point
  - Action: target code transformation

- Strategy
  - Use *Weaver* object
  - Show available join points and root
  - Iterate join points
    - List attributes, selects and actions

Weaver object members:

```
Weaver.root
      .joinpoints
      .selectsOf( jpName )
      .actionsOf( jpName )
      .attributesOf( jpName )
```

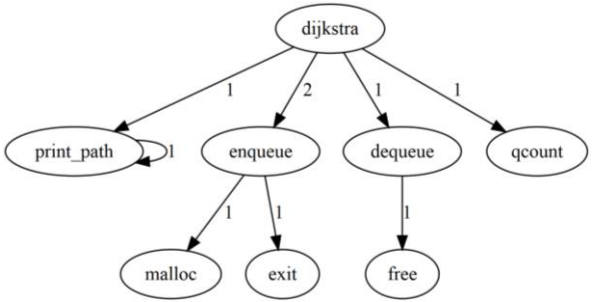
4.2 Static Call Graph

- Goal: Print a static call graph in dot format
- Identical to the MANET version
- File:
  - Any
- Strategy
  - Select pairs function->call
  - Count occurences of each pair
  - Iterate LaraObject and print

Example dot and resulting graph:

```
digraph static_call_graph {

            print_path->print_path [label="1"];
            enqueue->malloc [label="1"];
            enqueue->exit [label="1"];
            dequeue->free [label="1"];
            dijkstra->enqueue [label="2"];
            dijkstra->qcount [label="1"];
            dijkstra->dequeue [label="1"];
            dijkstra->print_path [label="1"];
}
```



4.3 Parfor To For

- Goal:
  - Identify Parfor loops and convert them to sequential for loops
- File:
  - subband.m (Modified)
- Strategy
  - Find all parfor loops
  - Replace the “header” of the loop with the “for” equivalent.

4.4 Validate Naming Conventions

- Goal:
  - Find violations of naming conventions
- File:
  - test.m
- Strategy
  - Find functions or variables
  - Check if their name follows the naming conventions

4.5 Find Common Typos

- Goal:
  - Find typos in code bases (e.g. “retrun” instead of “return”)
- File:
  - typo.m
- Strategy
  - Find implicit or script calls
  - See if their name is on a list of “common typos”.

4.6 Detect Missing Preallocations

- Goal:
  - Detect inefficient MATLAB idiom where matrices are resized in loops
- File:
  - latnrm.m
- Strategy
  - Find variables that are properly initialized
  - Find array sets in loops
    - See if they were properly initialized
    - Complain if they were not

Takeaway Points

- MATISSE: Framework to analyse, modify and compile MATLAB
- LARA can handle multiple languages
- We can use LARA to detect certain idioms