CS512 LABORATORY – WEEK 2 – Winter 2010

Prof. Kerstin Voigt

This lab session will have you implement basic graph-search in Common Lisp. Our implementations will start out very close to Paul Graham, ANSI Common Lisp, pp. 51.

Exercise 1. Open a file `bfs1.lisp` and copy the code in today's handout with same name. As you copy the lines of code, be mindful of the indentation style which greatly helps with the readability of Lisp code. Also, alert the instructor to any Lisp constructs that may be new and unfamiliar to you. E.g., expect to learn about

- `null`, `eql`, `reverse`, `append`
- `assoc`
- `mapcar` and `lambda`
- others ...

Load your file into `alisp` and test your function by typing

```
(gsearch 'a 'g graph)
```

Call `gsearch` for any other two nodes in the graph for which you want to test connectivity from the first to the second. Add nodes to the graph that are not reachable from node `'a`. Run again.

Discuss what you find useful or lacking in this implementation of breadth-first search.

Exercise 2. Create a copy called `bfs2.lisp` of your working `bfs1.lisp`. Make all modifications necessary to reflect the Lisp code of your second handout. This version of graph-search will compute a path of through the graph from a starting node to a goal node (if such path exists). Since the program is searching breadth-first, any solution path is guaranteed to be the **shortest** path. Understand the significance of this.

Test your program as in Exercise 1. Then compare both implementations.

Exercise 3. A small change to the code can convert your breadth-first searching program into one that conducts **depth-first** search. What is that change? Where do you apply it? Make your modifications in a copy of file `bfs2.lisp`, e.g.,`dfs.lisp`. Load your new program and test. Compare with `bfs`.