CS 3100 – Models of Computation – Fall 2011 – Handout dated 11/3/11

Project Proposal Due 11/8 – Appprovals/Comments Given 11/10 – Project Report Due 12/8

Below are some standard projects that you may do. You may also do a project of your own choosing, after seeking approval from me. My requirements are also listed below.

Project Requirements

- Must involve theory/practice taught
- Can involve coding Python, a thorough exploration of all features of JFLAP, or study of a topic relevant to the course but not covered
- You will be graded on the basis of your project report (accompanying webpages for code, illustrations are encouraged)
- You may work in groups of size 1-4. With higher group sizes, there should be more overall work presented. Each project member must document what their role was, and what the roles of all other project members were, and have a few sentences describing how much (what percentage), in their opinion, each project member contributed to the overall work.

Standard Projects

- Write Python routines to help experiment with non-deterministic push-down automata (we will call them "PDA" for short). *Recall that we are studying NPDAs; NPDA and DPDA* **are not equivalent** *in the sense of language recognition.* You must provide a file/module of function definitions (or use Python classes) that maintain PDA configurations. Users must be able to input PDAs, and an input string. Your program should allow users to pursue specific configurations and move forward expanding the chosen configuration. At any point, the user should be able to generate a PDF of their PDA with the configuration shown below it. You can imitate functionality found in JFLAP *with due credits given in your report.* All coding should be original work.
- Write Python routines to help experiment with Deterministic Turing Machines (DTM). *Recall that NDTMs and DTMs* **are equivalent** *in computational power.* You must provide a file/module of function definitions (or use Python classes) that maintain TM configurations. Users must be able to input TMs, and an input string. Your program should allow users to pursue the TM execution step by step, also seeing the tape. At any point, the user should be able to generate a PDF of their TM, along with the configuration of the finite control and the tape. You can imitate functionality found in JFLAP *with due credits given in your report.* All coding should be original work.
- Write a Post Correspondence Problem (PCP) solver. Given a collection of dominoes, it must search for solutions, and find and report. You can look up details at `http://en.wikipedia.org/wiki/Post_correspondence_problem`. Ling Zhao (U of Alberta) had a nice PCP tool and an MS thesis that goes with it.

  It is customary to present PCP instances in terms of "tiles" (or "dominoes"), with each tile at the respective index portrayed thus:

  ```
  Index  0    1    2    3
  ```

```
Tile  [01] [01] [01] [  1]
      [ 1] [  ] [ 0] [101]
```

The question is: is there an arrangement of one or more of the above tiles, with repetitions allowed, so that the top and bottom rows read the same? Here is such an arrangement:

```
[  1] [01] [01] [1  ] [01] --> This row reads 10101101
[101] [ 0] [ 1] [101] [  ] --> This row reads 10101101
```

In obtaining this solution, the tiles were picked, *with repetition*, according to the sequence given by the indices `3,2,0,3,1`:

```
Index  3     2    0    3     1    --> Solution is
                                      3,2,0,3,1
       [  1] [01] [01] [1  ] [01] --> reads 10101101
       [101] [ 0] [ 1] [101] [  ] --> reads 10101101
```

Given a PCP instance S of length N ($N = 4$ in our example), a *solution* is a sequence of numbers $i_1, i_2, \ldots, i_k$ where $k \geq 1$ and each $i_j \in \{0 \ldots N-1\}$ for $j \in \{1 \ldots k\}$ such that $S[i_1]S[i_2] \ldots S[i_k]$ has the property of the top and bottom rows reading the same. By the term "solution" we will mean either the above sequence of integers or a sequential arrangement of the corresponding tiles.

• The PCP instance below has an optimal solution of length 206, showing how quickly the search-space grows!

```
[1000] [01] [1  ] [00 ]
[   0] [ 0] [101] [001]
```

Your Projects

You may choose a project of similar magnitude as above. You may code in interesting languages such as Ocaml, Haskell, C#, Java, or Prolog, besides Python (these are amenable to quick prototyping and the use of higher order functions). It should adhere to material already taught or closely related material. If I don't approve your project idea, I will require you to choose one of the standard projects (in the interest of time). Therefore, you are encouraged to email teach-cs3100 anytime from now till 11/8/11 asking whether your idea has a chance of being approved. This may give you a head-start on project selection.

Grading Rubric

10 course points are allotted for the project. The project report ought to be equivalent to about 6 typeset PDF pages (**single-space**) of 11 point Times Roman font type. You must use Latex or Word to generate your document. Code and illustrations are extra and don't count in the page limit above. URLs may be used or figures/code may be provided as appendices. You must write precisely and use mathematical concepts wherever appropriate.

- Project report – clarity, organization, degree to which formal/mathematical ideas related to this course are used: 70%
- Novelty of idea, creativeness of project group members: 20%
- Difficulty of coding and/or intellectual taskS: 10%