6.001 Announcements and Notes

Gerald Dalley

February 17, 2005

This file contains extra notes, hints, and updates. Most of this information is in response to questions that have arisen either via email or during tutorials. Make sure you read this *entire* document.

1 Tutorials next week

Things are still up-in-the air a bit for what's happening next week. It looks like we will be having the regular Monday schedule. If this is indeed the case, I'll coordinate with the Tuesday people and squeeze them into the Monday tutorial sections as much as possible.

2 Project 1

2.1 Formatting, etc.

For project 1, I expect you to use *basebot.scm* as a starting point. Fill in the code that you need to write, adding any additional helper procedures as you go. You should succinctly document the code that you write, but don't write essays. You must also create your own intelligent test cases. For the tests, make sure it's clear what you're testing. Then, paste the results from the scheme interpreter into basebot.scm as comments, e.g.:

```
;; ... comments in template file ...

; Multiplies x by y by adding x to itself
; y times (this comment only needed if
; it tells something extra that's not
; told by the template-comments)
(define (slow-mul x y)
  (if (= y 0)
      0
      (+ x (slow-mul x (- y 1)))))

;; Test cases

; Try some normal multiplications
(slow-mul 10 10) ; -> 100
;Value:100
(slow-mul 10 5) ; -> 50
;Value:50

; Test boundary conditions when x
; and/or y are 0/1
(slow-mul 100 0) ; -> 0
;Value: 0
(slow-mul 0 100) ; -> 0
;Value: 0
(slow-mul 1 1) ; -> 1
;Value: 1
```

where all of the *;Value:xyz* comments should be pasted from the *scheme* or *transcript* buffer.

I will only accept a single file for grading. This means that if you submit multiple files, I will only look at the last file you send. All comments, code, test cases, and results must be embedded together in a single text file. I should be able to load the text file into 6.001 Scheme and hit M-o and have it give me the same results you claim, without any errors.

2.2 Critical Correction

It has come to our attention that there is still a typo in Project 1, problem 6. The line in *basebot.scm* that says:

```
;; dv = - (1/m beta v + g) dt
```

should say

```
;; dv = - (1/m speed beta v + g) dt
```

2.3 Using *let*

In project 1, you might find it handy to read section 1.3 in the text. The *let* special form can be quite useful for a few of the later problems. Even if you don't use *let* itself, understanding how to transform problems to embed temporary variables into lambdas is useful. If you would like, it is okay to use *let* in your project (assuming you do it correctly).

3 Complexity

3.1 Θ equivalences

Note that $\log_2 n = \Theta(\log_3 n)$ and $\log_3 n = \Theta(\log_2 n)$, but $3^n \neq \Theta(2^n)$.

3.2 Exponential Space (or not)

In one of the tutorial sessions, a student asked for an example of an algorithm that uses exponential space. It turns out that there is a very good example, if you take into consideration the new constructs we learned in lecture today. *make-tree* actually constructs an exponentially-large tree and is $\Theta\left(2^d\right)$ in space:

```
(define (make-tree d)
  (if (= d 0)
      nil
      (cons (make-tree (- d 1))
            (make-tree (- d 1)))))
```

In class, I attempted to come up with an example on-the-fly:

```
(define (foo x)
  (if (= x 0)
      0
      (+ (* 2 (foo (- x 1)))
         (* 2 (foo (- x 1))))))
```

It turns out that the space complexity of this procedure is linear ($\Theta(x)$). Reread section 1.1.5, especially the "Applicative order versus normal order" subsection. The key is that although we are generating an execution tree, only one path will be held in memory at a time, and each root-to-leaf path of the tree is of length x.

We can see that the time complexity of *foo* is $\Theta(2^x)$:

$$\begin{aligned}
t_x &= 2t_{x-1} + 6 \\
&= 2(2t_{x-2} + 6) + 6 \\
&= 2(2(2t_{x-3} + 6) + 6) + 6 \\
&... \\
&= 2^x + 2^{x-1}6 + 2^{x-2}6 + ... + 6 \\
&= \left(6\sum_{i=0}^{x} 2^i\right) - 5 \cdot 2^x \\
&= 6\frac{1-2^{x+1}}{1-2} - 5 \cdot 2^x \\
&= \Theta(2^x)
\end{aligned}$$

if we assume that $=$, $+$, $-$, and $*$ all cost unit time and that everything else is free.