CPSC 201: Introduction to Computer Science
Carsten Schürmann
Date: February 4, 2003

Homework 3

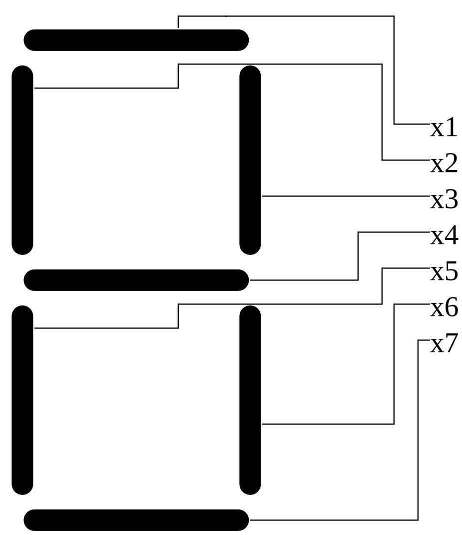
Due: Wednesday, February 12, 2003.

Guidelines

While we acknowledge that beauty is in the eye of the beholder, you should nonetheless strive for elegance in your code. Not every program which runs deserves full credit. Make sure to state invariants in comments which are sometimes implicit in the informal presentation of an exercise. If auxiliary functions are required, describe concisely what they implement. Do not reinvent wheels, and try to make your functions small and easy to understand. Use tasteful layout and avoid long winded and contorted code. None of the problems requires more than a few lines of SML code.

Exercise 1 This exercise asks you to implement the circuitry for a digital display using only NAND gates. To do this exercise, copy the file `blue3.sml` from the course directory `/c/cs201/lib/code/ass3` into your personal directory. Do not edit the signature or structure `Circuit`. It contains all you need to finish this problem.

The goal of this problem is to convert a number between `0` and `7`, given in bit representation *abc* into the control logic of a digital display unit which has seven connectors.



which is represented as `DISPLAY x1 x2 x3 x4 x5 x6 x7` in ML.

1. Write a function in ML

```
circuit :  Circuit.Gate * Circuit.Gate * Circuit.Gate
           -> Circuit.Gate
```

that corresponds to a circuit that controls the digit display above. For example

```
Circuit.simulate (circuit (Circuit.SIGNAL true,
                           Circuit.SIGNAL true,
                           Circuit.SIGNAL false));
```

should display the digit `6` as the following.

```
      ------
     /
    /
    ------
  /     /
 /     /
 ------
```

You probably want to write some auxiliary functions, that helps you construct the circuit. Be inventive!

2. Write a function

```
count :  Circuit.Gate -> int
```

that counts the number of NAND gates used. Just to give you a ballpark number what to expect: my circuit contained `7209534` gates.

Exercise 2 In this exercise, you are not asked to write a program but to prove an invariant. Prove it carefully. It is not difficult. Go step by step. State carefully what assumption you want to do induction on. First, we define the `map` function that applies a function `f` to each element of a list and returns a new list.

```
fun map f nil =  nil
  | map f (h :: t) = f h :: map f t
```

Second, we write `f o g` for `fn x => f (g x)` and third, we consider the function that computes the length of a list.

```
fun length nil = 0
  | length (h :: t) = 1 + length t
```

1. Prove for all lists `L` and functions `f`, `g` that

$$\texttt{map f (map g L)} \equiv \texttt{map (f o g) L}$$

where $e_1 \equiv e_2$ means that $e_1 \stackrel{*}{\Longrightarrow} v$ and $e_2 \stackrel{*}{\Longrightarrow} v$.

2. Prove for all lists `L` and functions `f` that

$$\texttt{length L} \equiv \texttt{length (map f L)}.$$