Assignment 1

COS30023 - Languages in Software Development

Daniel Parker - 971328X

August 17, 2014

1. Proof Tree

$$\frac{\text{father(X,luke)} \quad \text{father(X,leia)} \quad \text{mother(Y,luke)} \quad \text{mother(Y,leia)}}{\text{sibling(luke, leia)}}$$

2. Bubble Sort

2.1. bubble/2

In planning this predicate, I rationalised the result I wanted as being the sorted result of the head of the current list and the head of the tail of the current list. Then I would keep the smaller of the two, and recursively bubble the bigger part as the head of the remaining tail.

2.2. bubble_sort/2

The bubble_sort predicate makes use of three other predicates; bubble/2, reverse/2, and remove/3. This predicated is again recursive in nature. A sorted list is created by bubbling, reversing the result, removing the top most element (head), reversing it back again and calling the bubble sort on the remaining tail.

2.3. Source

```
% Bubble list from left to right
%
% LOGIC:
%    Check the head against the head of the tail
%    swapping the heads if the first head is bigger than the second head.
%    recursively bubble the remaining tail with the correct swapped head.
%
bubble([],[]).
bubble([X],[X]).
```

```
bubble([X,Y|T], [Y|Z]) :- X > Y, bubble([X|T], Z).
bubble([X,Y|T], [X|Z]) :- X =< Y, bubble([Y|T], Z).

% Separate the head and the tail of the list
remove([H|T],H,T).

% Bubble sort the whole list
%
% LOGIC:
%    Bubble the list, reverse it and remove the head.
%    Bubble the tail of the resulting list. Repeat until all parts are sorted.
bubble_sort([],[]).
bubble_sort([X], [X]).
bubble_sort(X,[H|L]) :- bubble(X, Y),
                        reverse(Y, Z),
                        remove(Z,H,T),
                        reverse(T,C),
                        bubble_sort(C, L).
```

3. Logical Circuits

3.1. NAND

```
% Nand gate specification
c_nand(t,t,f).
c_nand(t,f,t).
c_nand(f,t,t).
c_nand(f,f,t).
```

3.2. NAND Logic Gates

3.2.1. AND

```
% Bitwise AND
c_and(X,Y,Z) :-     c_nand(X,Y,R),
                    c_nand(R,R,Z).
```

3.2.2. OR

```
% Bitwise OR
c_or(X,Y,Z) :-      c_nand(X,X,X1),
                    c_nand(Y,Y,Y1),
                    c_nand(X1,Y1,Z).
```

3.2.3. NOR

```
% Bitwise NOR
c_nor(X,Y,Z) :-     c_nand(X,X,X1),
                    c_nand(Y,Y,Y1),
                    c_nand(X1,Y1,R),
                    c_nand(R,R,Z).
```

3.2.4. XOR

```
% Bitwise XOR
c_xor(X,Y,Z) :-     c_nand(X,Y,R1),
                    c_nand(X,R1,A),
                    c_nand(Y,R1,B),
                    c_nand(A,B,Z).
```

3.2.5. XNOR

```
% Bitwise XNOR
c_xnor(X,Y,Z) :-    c_nand(X,Y,R1),
                    c_nand(X,R1,A),
                    c_nand(Y,R1,B),
                    c_nand(A,B,C),
                    c_nand(C,C,Z).
```