UNIVERSITY OF EDINBURGH

FACULTY OF SCIENCE AND ENGINEERING

MASTER OF SCIENCE IN COMPUTER SCIENCE

COMPUTER-AIDED FORMAL REASONING

Tuesday 2 May 2000

9:30 to 11:00

Examiners:

I. Watson (External)

R.N. Ibbett (Chairman)

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

1. **Theorem proving**

   (a) Give two different motivations for formalising proofs [*3 marks*]

   (b) Name three different tasks of a proof assistant [*3 marks*]

   (c) Describe in your own words the idea of *prototype verification* in PVS and its relationship to soundness of the prover. [*3 marks*]

   (d) The following is a generic sequent in sequent calculus.

   $$\Gamma \Longrightarrow_{\mathcal{L}} \Delta$$

   Explain what the symbols $\Gamma, \Delta, \mathcal{L}$ stand for. Explain how the semantics (meaning) of such a sequent is defined. [*6 marks*]

   (e) Which *logical rule* of the sequent calculus ends with

   $$\Gamma, \forall x{:}\tau.\phi \Longrightarrow_{\mathcal{L}} \Delta$$

   and what is its premise? Which PVS command corresponds to this rule? Without making any assumptions on the form of Γ, Δ, ϕ do there exist other rules in the sequent calculus having that same conclusion? [*5 marks*]

   (f) Give a proof in sequent calculus of the formula $\forall x{:}\tau.(\phi \Rightarrow \exists x{:}\tau.\phi)$. Give a proof of the same formula in PVS. [*7 marks*]

2. **Hardware verification**

(a) A *JK-flipflop* is a device with two inputs J, K (as well as a clock which we leave implicit) and one output Q. The value Q' of Q after the clock (at time $t+1$) is obtained from the values of J, K, Q before the clock (at time t) according to the following table:

J	K	Q'
F	F	Q
F	T	F
T	F	T
T	T	$\neg Q$

That is to say, if J, K are both false then Q stays as it was; if J, K are different then J determines the next value of Q and if J, K are both true then Q changes its value.

We make the simplifying assumption that at time $t = 0$ (when power is switched on) we have $Q = F$.

Complete the two boxes in the following PVS theory so that the definition of `flipflop` adequately reflects the above description. Upon completion you should copy the contents of the boxes to your answer booklet. [*4 marks*]

```
flipflop :  THEORY BEGIN
  IMPORTING bitvectors@bit
  flipflop(J,K:[nat -> bit])(t:nat) :  RECURSIVE bit =
    IF t=0 THEN FALSE ELSIF
      NOT J(t-1) AND NOT K(t-1) THEN flipflop(J,K)(t-1)
    ELSIF
```

$\boxed{}$

```
  MEASURE [ ]
END flipflop
```

QUESTION CONTINUES OVERLEAF

(b) A synchronous N-bit counter can be built up from N JK-flipflops (with inputs $J_0, K_0, \ldots, J_{N-1}, K_{N-1}$ and outputs $Q_0, \ldots Q_{N-1}$) by setting

- $J_0 = K_0 = T$ (so Q_0 changes with every clock)
- $J_{n+1} = K_{n+1} = Q_0 \wedge Q_1 \wedge \ldots \wedge Q_n$ (so Q_{n+1} changes if all the Q_i with $i = 0, \ldots, n$ were T)

The idea is that at time t the outputs $Q_{N-1}, \ldots, Q_0$ are precisely $t \bmod 2^N$ in binary.

Complete the two boxes in the following PVS theory so that the definition of `counter` adequately reflects the above description. Upon completion you should copy the contents of the boxes to your answer booklet. [*4 marks*]

```
counter[N:nat] :  THEORY
  BEGIN
  IMPORTING flipflop

  bigand(n:nat)(b:[below(n)->bit]):RECURSIVE bit =
    IF n=0 THEN TRUE
    ELSE
      b(n-1) AND bigand(n-1)(LAMBDA(j:below(n-1)):b(j))
    ENDIF
  MEASURE n

  counter(j:upto(N)):RECURSIVE [nat->bit] =
    LET H(t:nat) =
┌──────────────────────────────────────────────────┐
│                                                  │
│                                                  │
└──────────────────────────────────────────────────┘
    IN flipflop(H,H)
  MEASURE [ ]
END counter
```

Hint: Here `counter(j,t)` is meant to be Q_j at time t. The signal H stands for the (identical) values J_j and K_j. You have to make sure that it becomes the "and" of the Q_i's with $i < j$.

QUESTION CONTINUES OVERLEAF

(c) We see that the definition of `counter` uses `flipflop` only with both inputs identified. Complete the following lemma which describes this situation. Upon completion you should copy the contents of the box to your answer booklet. [*3 marks*]

```
flipflop_lemma_diag :  LEMMA
  FORALL (H:[nat->bit]):FORALL(t:nat):
    flipflop(H,H)(t+1)=
```

(d) Formulate a specification of `counter` which describes its intended behaviour. Recall that `bitvectors@bv_nat` contains the following function which translates binary representations into integers.

```
bv2nat:[bvec[N]->nat]
```

where `bvec[N]=[upto(N)->bit]` [*6 marks*]

(e) You are provided with the following theory

```
succ[N:nat]:THEORY
BEGIN
  IMPORTING counter[N],bitvectors@bv_core
  succ(bv:bvec[N]):bvec[N] = LAMBDA(j:below(N)):
  IF bigand(j)(LAMBDA(i:below(j)):bv(i))
    THEN NOT bv(j) ELSE bv(j)
  ENDIF
  succ_char :  LEMMA
    FORALL(bv:bvec[N]):
    bv2nat(succ(bv))=mod(bv2nat(bv)+1,exp2(N))
END succ
```

Explain informally how you would prove your specification of `counter` building on that theory. Please state clearly whether and where you plan to use induction and what extra lemmas you would introduce. You may assume the PVS functions `mod:[nat,nat->nat]` and `exp2 :  [nat->nat]` as well as arithmetical truths describing these. [*8 marks*]

3. **Lego and Type Theory**

(a) What is a *proof term*? Why are proof terms important? [*4 marks*]

(b) Formalise the following formulas

$$\phi \equiv \forall x, y, z{:}\tau. R(x, y) \Rightarrow R(y, x) \Rightarrow R(x, z)$$
$$\psi \equiv \forall x, y{:}\tau. R(x, y) \Rightarrow R(y, x)$$

in Lego (you need to declare τ and R appropriately.) . [*4 marks*]

(c) Give a Lego proof of the formula

$$\phi \Rightarrow \psi \Rightarrow \forall x, y{:}R(x, y) \Rightarrow R(x, x)$$

. [*7 marks*]

(d) Suppose that the following declarations have been made in Lego:

`[A:Type(0)][B:A->Type(1)]`

What is the type of `{x:A}B x` and what is the intuitive meaning of that expression? [*3 marks*]

(e) State the formation rule SUM-TYPE for Sigma types. [*3 marks*]

(f) Name three different applications of Sigma types. [*4 marks*]