Final

Important Notes

This is an open-book exam. You may consult any book, paper, note, or on-line resource, but discussion with others (in person or via a network) is strictly forbidden.

Problems 2 and 4 require electronic submission. Please pack all files for the two problems in one single .zip file and email it to the instructor (`tsay@ntu.edu.tw`).

Problems

1. (20 %) Prove, using *Natural Deduction* (in the sequent form), the validity of the following sequents.

   (a) $\neg(p \wedge q) \vdash \neg p \vee \neg q$

   (b) $\exists x(\forall y A) \vee \exists x(\forall y B) \vdash \exists x(\forall y(A \vee B))$

2. (20 %) Consider the Coq formalization of group theory in one of our homework assignments. Your task here is to prove two lemmas: one states that $(a^-)^- = a$ for every a, and the other states that $(a \cdot b)^- = b^- \cdot a^-$ for every a and b. You may introduce and prove other lemmas to better organize your proofs, but should not use highly automated proof commands/tactics. Please write down the proof scripts on the exam paper and also include the corresponding self-contained .v file in the single .zip file for the instructor.

```
Section Group.

Parameter G : Set.
Parameter op : G -> G -> G.
Parameter e : G.
Parameter inv : G -> G.

Infix "o" := op (at level 35, right associativity).
Notation "a -" := (inv a) (at level 25, left associativity).
Axiom assoc : forall a b c : G, a o (b o c) = (a o b) o c.
Axiom unit_l : forall a : G, e o a = a.
Axiom unit_r : forall a : G, a o e = a.
Axiom inverse_l : forall a : G, a- o a = e.
Axiom inverse_r : forall a : G, a o a- = e.

Lemma inv_inv :
```

```
  forall a : G, a-- = a.

Lemma inv_op :
  forall a b : G, (a o b)- = b- o a-.

End Group.
```

3. (10 %) Prove that $\models wlp(S_1; S_2, q) \leftrightarrow wlp(S_1, wlp(S_2, q))$ which we claimed when proving the completeness of System PD (for the validity of a Hoare triple with partial correctness semantics).

   Here, assuming a sufficiently expressive assertion language, $wlp(S, q)$ denotes the assertion p such that $[\![p]\!] = wlp(S, [\![q]\!])$, where $[\![p]\!]$ is defined as $\{\sigma \in \Sigma \mid \sigma \models p\}$ (i.e., the set of states where p holds) and $wlp(S, \Phi)$ as $\{\sigma \in \Sigma \mid \mathcal{M}[\![S]\!](\sigma) \subseteq \Phi\}$. Recall that, for $\sigma \in \Sigma$, $\mathcal{M}[\![S]\!](\sigma) = \{\tau \in \Sigma \mid \langle S, \sigma\rangle \rightarrow^* \langle E, \tau\rangle\}$, $\mathcal{M}[\![S]\!](\bot) = \emptyset$, and, for $X \subseteq \Sigma \cup \{\bot\}$, $\mathcal{M}[\![S]\!](X) = \bigcup_{\sigma \in X} \mathcal{M}[\![S]\!](\sigma)$.

4. (20 %) The following simple C function concatenates one array of m integers and another of n integers to obtain a third array of $m + n$ integers. Annotate the code to show that, if the first two arrays are sorted and the first element of the second array is larger than or equal to the last element of the first array, then the third array is sorted as well, and prove correctness of your annotation using Frama-C. Please write down the annotations on the exam paper and include the corresponding self-contained .c file in the single .zip file for the instructor.

```
void concateArrays(int* a, int m, int* b, int n, int* c)
{ int i;

  for (i=0; i<m; i++)
    c[i] = a[i];

  for (i=0; i<n; i++)
    c[m+i] = b[i];

}
```

5. (20 %) Prove the partial correctness of the following program using the Owicki-Gries method. Variables T_0, T_1, t_0, and t_1 are Boolean;

$$\{acc = 0\}$$

$$
\left[\begin{array}{lcl}
t_0 := T_0; & & t_1 := T_1; \\
T_1 := t_0; & & T_0 := \neg t_1; \\
\textbf{await } T_0 \neq t_0; & & \textbf{await } T_1 \neq t_1; \\
s_0 := acc; & \| & s_1 := acc; \\
acc := s_0 + 1; & & acc := s_1 + 1; \\
t_0 := T_0; & & t_1 := T_1; \\
T_1 := t_0 & & T_0 := \neg t_1
\end{array}\right]
$$
$$\{acc = 2\}$$

6. (10 %) In the temporal verification framework of Manna and Pnueli, one uses the following rule to prove that a state predicate q holds in all states of every computation of a program with initial condition Θ and a set of actions $\mathcal{T}$.

$$\frac{\begin{array}{l}\Theta \rightarrow \varphi \\ \{\varphi\}\ \mathcal{T}\ \{\varphi\} \\ \varphi \rightarrow q\end{array}}{\Box q}$$

The first two premises correspond respectively to "φ initially " and "φ stable", which taken together means "φ invariant", in the UNITY logic. However, UNITY does not have a direct way to express $\Box q$. Nonetheless (with a potential risk of introducing contradiction to the inductive definition of an invariant), through the Substitution Axiom, which states that "an invariant may be replaced by *true*, and vice versa, in any property of a program", one may simply use "q invariant" to express the desired property $\Box q$ for a program. The relevant (derived) rule is the following:

$$\frac{\begin{array}{l}\varphi \text{ invariant} \\ \varphi \rightarrow q\end{array}}{q \text{ invariant}}$$

(Note: to differentiate invariants such derived from those that are truly inductive, some researchers have suggested to write in the form of "q invariant$_\varphi$", where φ is the inductive invariant used to derive "q invariant".)

Please show the validity of the above derived rule under the Substitution Axiom.