TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2010

END-OF-YEAR

SWEN 224
Formal Foundations of Programming

Time Allowed: 3 Hours

Instructions:
- Answer all seven questions.
- The exam will be marked out of one hundred and eighty (180).
- Calculators ARE NOT ALLOWED.
- Non-electronic Foreign language dictionaries are allowed.
- No other reference material is allowed.

Question 1. Static Alloy Modelling [20 marks]

Consider the following Alloy specification, which models family relations.

```
abstract sig Person {
  father: lone Man,
  mother: lone Woman
}

sig Man extends Person {
  wife: lone Woman
}

sig Woman extends Person {
  husband: lone Man
}

fun parent: Person -> Person {
  mother + father + father.wife + mother.husband
}

fun grandparent: Person -> Person {
  parent.parent
}

pred p {
  some p: Person | p in p.grandparent
}
```

(a) Understanding an Instance

Consider the following instance of this model:

$$
\begin{aligned}
\texttt{Man} &= \{Adam, Bob\} \\
\texttt{Woman} &= \{Ana, Jane\} \\
\texttt{husband} &= \begin{array}{|c|c|}\hline Ana & Bob \\ \hline Jane & Adam \\ \hline\end{array} \\
\texttt{wife} &= \begin{array}{|c|c|}\hline Bob & Ana \\ \hline Adam & Jane \\ \hline\end{array} \\
\texttt{father} &= \begin{array}{|c|c|}\hline Adam & Bob \\ \hline\end{array} \\
\texttt{mother} &= \begin{array}{|c|c|}\hline Ana & Jane \\ \hline\end{array}
\end{aligned}
$$

(i) [1 mark] Compute `Adam.wife`

(ii) [1 mark] Compute `~wife`

(iii) [2 marks] Compute `parent`

(iv) [2 marks] Compute `Adam.grandparent`

(v) [2 marks] In your own words, describe what predicate p expresses.

(vi) [2 marks] Is the predicate `p` true for this instance? Briefly explain why or why not.

(b) Writing Alloy

(i) [1 mark] Provide a run command that shows instances with at least one woman.

(ii) [2 marks] Provide a run command that shows instances with at least one woman who is the wife of someone.

(iii) [2 marks] Write a function called `spouse` that takes a person as argument and returns the spouse (that is, the wife if the person is male or the husband if the person is female) of the given person.

(iv) [3 marks] Write a check command to check whether every person who is a spouse of someone has a spouse. Provide a counter-example that could have been generated if your check command was added to the above model and executed.

(v) [2 marks] Add a fact to the specification to ensure that every person with a spouse is the spouse of his or her spouse.

Question 2. Dynamic Alloy Modelling [25 marks]

The following Alloy provides a dynamic model for the marriage relation.

```
abstract sig Person {}
sig Man, Woman extends Person {}

sig State {
  husband: Woman -> Man,
  wife: Man -> Woman
}

pred divorce(s,s': State, m: Man, w: Woman) {
  s'.husband = (Woman-w) <: s.husband
  s'.wife = (Man-m) <: s.wife
}
```

(a) Understanding an Instance

Consider the following instance of this model:

$$\texttt{Man} = \{Adam, Bob\}$$

$$\texttt{Woman} = \{Ana, Jane\}$$

$$\texttt{State} = \{S0, S1, S2\}$$

$$\texttt{husband} =$$

$S1$	Ana	$Adam$
$S2$	Ana	$Adam$
$S2$	Ana	Bob

$$\texttt{wife} =$$

$S2$	$Adam$	Ana
$S2$	Bob	Ana

(i) [3 marks] For each of the states $S0$, $S1$, and $S2$ explain who is husband or wife to whom.

(ii) [3 marks] Compute `(Man-Bob) <: S2.wife`

(iii) [3 marks] Explain the role of the `divorce` predicate and its arguments.

(iv) [2 marks] Can the `divorce` predicate be satisfied in the given instance? If it can, identify the arguments for which the `divorce` predicate is true.

(b) Extending the Alloy Model

(i) [4 marks] Provide a predicate called `inv` that takes a state as argument and is true if, for the given state, every man has at most one wife and every woman has at most one husband.

(ii) [6 marks] Provide an operation `marry` that models a man and a woman getting married and preserves the invariant.

(iii) [4 marks] Write an Alloy command to check whether the `marry` operation you provided in **(ii)** preserves invariant `inv`.

Question 3. JML [25 marks]

(a) Understanding JML

Do the following methods correctly implement their specification? Give a brief explanation why you think they do or do not.

(i) [2 marks]

```
//@ requires true;
//@ ensures 0 <= \result && \result <= 50;
int magicNumber1() { return 42; }
```

(ii) [2 marks]

```
//@ requires x == 0;
//@ ensures \result == 41 || \result == 42;
int magicNumber2(int x) { return 42; }
```

(iii) [2 marks]

```
//@ requires false;
//@ ensures false;
int magicNumber3() { return 42; }
```

(iv) [2 marks]

```
//@ requires a != null;
//@ ensures 0 <= \result && \result < a.length;
//@ ensures a[\result] == value;
int find1(int[] a, int value) {
  a[0] = value;
  return 0;
}
```

(v) [2 marks]

```
//@ requires a != null;
//@ ensures 0 <= \result && \result <= a.length;
//@ ensures a[\result] == value;
int find2(int[] a, int value) {
  for (int i = 0; i < a.length; i++) {
    if (a[i] == value) return i;
  }
  return a.length;
}
```

(b) Writing JML

Write a JML specification for each of the following methods.

(i) [3 marks] `int max(int x, int y)`

Method `max` returns the maximum of the two given integers.

(ii) [5 marks] `int[] replace(int[] a, int x, int y)`

Given integers `x` and `y`, and a non-null integer array `a`, the array returned by method `replace` is the same as `a` with all occurrences of `x` replaced by `y`.

(c) Class Invariants

Consider the following Java class to represent a bank account.

```
public class Account
{
  //@ invariant amount >= 0;
  private int amount;

  public void deposit(int value) {
    amount += value;
  }

  public void withdraw(int value) {
    amount -= value;
  }
}
```

(i) [3 marks] Explain what the JML invariant clause means.

(ii) [2 marks] Does the `deposit` method preserve the given JML invariant? Give a brief explanation why you think it does or does not.

(iii) [2 marks] The `withdraw` method does not preserve the given JML invariant. Provide a JML annotation that ensures that the `withdraw` method preserves the given JML invariant.

Question 4. Loop Invariants and Variants [25 marks]

Consider the following Java method:

```
boolean test(String s) {
  int k = 0;
  int n = s.length()/2;
  while ( k < n && s.charAt(k) == s.charAt(n+k) ) {
    k = k+1;
  }
  return ( k == n );
}
```

Given a string, `s`, of even length, this method determines whether `s` is the concatenation of two copies of the same string. For example, `test` will return `true` if `s` is `"abcabc"` or `"xx"` and false if `s` is `"abba"`.

(a) [5 marks] Give a JML specification (precondition and postcondition) that formalises the above description of the `test` method.

(b) [12 marks] Give a loop invariant and use it to give an argument (informal proof) that the method satisfies its specification.

(c) [8 marks] Give a loop variant and an argument (informal proof) to show that the method terminates.

Question 5. Properties of Strings and Languages [30 marks]

(a) [4 marks] A string s is a *prefix* of another string t, written $s \preceq t$, if t is the concatentation of s and some other string, say u; i.e. $s \preceq t \equiv \exists u : t = s \frown u$.

Show that if x is a prefix of y and y is a prefix of z, then x is a prefix of z (i.e. $x \preceq y$ and $y \preceq z$ implies $x \preceq z$).

(b) [10 marks] Give a proof for each of the following equalities, where X and Y are languages over some alphabet A, and X^R is the *reflection* of X, i.e. $X^R = \{\alpha^R \mid \alpha \in X\}$:

(i) $(X \cup Y)^R = X^R \cup Y^R$

(ii) $(X \frown Y)^R = Y^R \frown X^R$

(iii) $(X^*)^R = (X^R)^*$

In these proofs, you should explain each step and state the properties of sets and strings that they rely on, but you do not need to prove these properties.

(c) [5 marks] Explain how the results in part **(b)** above can be used to show that the class of regular languages is closed under reflection.

(d) [6 marks] Explain how the set of strings represented by a *trie* can be obained by solving a set of equations. Illustrate this result using a small example.

(e) [5 marks] How is the result in part **(d)** above affected if we represent the set of strings by an NFA rather than a trie? Illustrate your answer using a small example.

Question 6. Finite Acceptors [30 marks]

(a) Consider the following NFA, M_1:

(i) [3 marks] Show the sequence of configurations that M_1 passes through in accepting the input *aaabc*. Note that you should show *all* states that M may be in after accepting part of the input.

(ii) [3 marks] Write a regular expression describing the language accepted by M.

(iii) [8 marks] Draw a transition diagram for the DFA obtained by applying the *subset construction* to M. You only need to show reachable states.

(b) [10 marks] Given an NFA $M = (Q, q_I, A, N, F)$, show how to construct an NFA, $M' = (Q', q'_I, A', N', F')$, which accepts the reflection of the language accepted by M (i.e. $\mathcal{L}(M') = \mathcal{L}(M)^R$).

Give a brief argument to show that the resulting NFA does in fact accept the required language.

(c) [6 marks] Given an NFA$_\epsilon$ $M = (Q, q_I, A, N, F)$ (i.e. an NFA with null transitions), describe an algorithm to determine whether M accepts the empty string.

Question 7. Context Free Grammars [25 marks]

(a) [5 marks] Given a grammar, $G = (V_N, V_T, S, P)$, explain what it means for a tree to be a *parse tree* for a string w from G.

(b) Consider the grammar $G_P = (\{S\}, \{a, b\}, S, \{S \to \lambda \mid aSa \mid bSb\})$.

(i) [2 marks] Write all of the strings of length less than or equal to 4 that can be produced from G_P.

(Note that a string s can be produced from G iff s is in the language defined by G, i.e. $s \in \mathcal{L}(G)$.)

(ii) [4 marks] Show that every string produced from G_P is a *palindrome*. (A string s is a palindrome if it is equal to its own reflection, i.e. if $s^R = s$.)

(iii) [6 marks] Can every palindrome over $\{a, b\}$ be produced by G_P? If so, prove that this is the case. If not, modify the grammar so that it does produce all palindromes over $\{a, b\}$ and prove that the resulting grammar does indeed produce all palindromes over $\{a, b\}$.

(c) [8 marks] Given two grammars, $G = (V_N, V_T, S, P)$ and $G' = (V'_N, V'_T, S', P')$, defining languages X and Y (i.e. $X = \mathcal{L}(G)$ and $Y = \mathcal{L}(G')$), show how you can construct a grammar $G'' = (V''_N, V''_T, S'', P'')$ which defines the concatenation of X and Y (i.e. $\mathcal{L}(G'') = V_N \frown V'_N = \{\}$).

Give a brief argument to show that the resulting grammar does in fact define the required language.

****