Victoria University of Wellington

DEGREE EXAMINATIONS — 1994

COMP 426

(END OF YEAR)

COMP 426

Computational Logic 2

Time Allowed: THREE hours.

Instructions: Candidates should attempt all questions.
This exam will be marked out of 100.

Question 1. **[20 marks]**

This question concerns extensions to Diller's **Z** specification of the Library Problem, dealing with "overdue" books and fines.

First, consider a version of the Library system in which due dates are allocated, and recorded, when books are checked out. Below are the main parts of the **Z** specification that are added or changed.

We introduce a basic set, *Date*, representing dates, a function *add_date* such that, for any day d and natural number n, $add\_date(d, n)$ returns a date n days later than d, and a global constant, *LoanPeriod*, indicating the number of days for which books are issued.

$[Date]$

$$add\_date : Date \times \mathbb{N} \longrightarrow Date$$

$$LoanPeriod : \mathbb{N}$$

The state schema *LibraryDB* is extended to include a function *duedate* to record due dates for books currently checked out.

$$
\begin{array}{|l}
\textit{LibraryDB} \\
\hline
\textit{borrower}, \textit{staff} : \mathbb{F}\ \textit{Person} \\
\textit{available}, \textit{checkedout} : \mathbb{F}\ \textit{Copy} \\
\textit{previouslyborrowedby}, \textit{borrowedby} : \textit{Copy} \nrightarrow \textit{Person} \\
\textit{duedate} : \textit{Copy} \nrightarrow \textit{Date} \\
\hline
\textit{borrower} \cap \textit{staff} = \varnothing \\
\textit{available} \cap \textit{checkedout} = \varnothing \\
\operatorname{dom} \textit{borrowedby} = \textit{checkedout} \\
\operatorname{ran} \textit{borrowedby} \subseteq \textit{borrower} \\
\operatorname{dom} \textit{previouslyborrowedby} \subseteq \textit{available} \cup \textit{checkedout} \\
\operatorname{ran} \textit{previouslyborrowedby} \subseteq \textit{borrower} \\
\forall p : \textit{Person} \mid p \in \textit{borrower} \bullet \\
\quad \#(\textit{borrowedby}^{\sim}(\!|\{p\}|\!)) \leq \textit{MaxCopiesAllowed} \\
\operatorname{dom} \textit{duedate} = \textit{checkedout} \\
\hline
\end{array}
$$

The operation schema *CheckOutCopy* is modified to take the current date as an additional input ($d?$) and record the due date for the book being checked out.

$$
\begin{array}{|l}
\textit{CheckOutCopy} \\
\hline
\Delta \textit{LibraryState} \\
\Xi \textit{ParaLibrary} \\
n? : \textit{Person} \\
c? : \textit{Copy} \\
d? : \textit{Date} \\
\hline
n? \in \textit{borrower} \\
c? \in \textit{available} \\
\#(\textit{borrowedby}^{\sim}(\!|\{n?\}|\!)) < \textit{MaxCopiesAllowed} \\
\textit{available}' = \textit{available} \setminus \{c?\} \\
\textit{checkedout}' = \textit{checkedout} \cup \{c?\} \\
\textit{borrowedby}' = \textit{borrowedby} \cup \{c? \mapsto n?\} \\
\textit{previouslyborrowedby}' = \textit{previouslyborrowedby} \\
\textit{duedate}' = \textit{duedate} \cup \{c? \mapsto \textit{add\_date}(d?, \textit{LoanPeriod})\} \\
\textit{borrower}' = \textit{borrower} \\
\textit{staff}' = \textit{staff} \\
\hline
\end{array}
$$

The operation schema *Return* is also modified, to update *duedate* when a book is returned.

$$
\begin{array}{|l}
\hline
\textit{Return} \\
\Delta \textit{LibraryState} \\
\Xi \textit{ParaLibrary} \\
c? : \textit{Copy} \\
\hline
c? \in \textit{checkedout} \\
\\
\textit{available}' = \textit{available} \cup \{c?\} \\
\textit{checkedout}' = \textit{checkedout} \setminus \{c?\} \\
\textit{borrowedby}' = \{c?\} \mathbin{⩤} \textit{borrowedby} \\
\textit{previouslyborrowedby}' = \textit{previouslyborrowedby} \oplus \{c? \mapsto \textit{borrowedby}(c?)\} \\
\\
\textit{duedate}' = \{c?\} \mathbin{⩤} \textit{duedate} \\
\\
\textit{borrower}' = \textit{borrower} \\
\textit{staff}' = \textit{staff} \\
\hline
\end{array}
$$

(a) Show that the operations *CheckOutCopy* and *Return* preserve the invariant added to *LibarayDB*, viz. $\operatorname{dom} \textit{duedate} = \textit{checkout}$. [2 marks]

(b) Why can the new value for *duedate* in *CheckOutCopy* be defined using $\cup$ rather than $\oplus$? [1 mark]

(c) In this definition, *LoanPeriod* is a global constant. How would you modify the specification to allow library staff to modify *LoanPeriod* ? [1 mark]

(d) Show how you would modify the *CheckOutCopy* schema to reflect the additional requirement that a borrower not be allowed to borrow additional books if they have any book overdue. [2 marks]

Now, consider a version of the Library system in which borrowers are fined for not returning books by the due date, as follows:

- Fines are charged at a fixed rate for each day a book is overdue.
- A record is maintained of the total outstanding fine (if any) owed by each borrower.
- When a borrower returns an overdue book, the fine is calculated and added to the fine record for that borrower.
- The *CheckOutBook* operator is further modified so that a borrower may not take any books out if they have fines outstanding (in addition to the other constraints already imposed).
- A new operator is added to reduce a borrower's fine when all or part of the fine is paid. A user can pay off any part of the outstanding fine, but cannot pay more than they owe.

(e) Indicate the changes and/or additions required to the Library specification to accommodate these requirements. State any additional assumptions you need to make in formalising these requirements. [10 marks]

(f) Give the precondition for the final version of *CheckOutBook*, and explain how this can be used in constructing a robust version of this operation. [2 marks]

(g) Below is the schema for interrogating the records relating to a given borrower:

$$
\begin{array}{|l}
\hline
\textit{BooksBorrowedBy} \\
\Xi \textit{LibraryState} \\
n? : \textit{Person} \\
out! : \mathbb{F}\ \textit{Copy} \\
\hline
n? \in \textit{borrower} \\
out! = \textit{borrowedby}^{\sim} (\!| \{n?\} |\!) \\
\hline
\end{array}
$$

Modify this schema so that, for each book currently checked out by the borrower, it also outputs the due date for that book, and the fine (if any) that would be charged if the book was returned that day. [2 marks]

Question 2. **[15 marks]**

This question concerns a **Z** specification of a class register system, intended to record the students enrolled in a class and which of them have completed their assigned homework, and a proposed data refinement of it.

First, consider the following (partial) abstract specification of the class register system.

We introduce a basic set, *Student*, to represent individual students, and a global constant *MaxSize* of type $\mathbb{N}$ to specify the maximum class size.

$[\textit{Student}]$

$$
\begin{array}{|l}
\textit{MaxSize} : \mathbb{N}
\end{array}
$$

The abstract state consists of a pair of sets, one to record the students enrolled and one to record those who have completed the homework.

$$
\begin{array}{|l}
\hline
\textit{ClassA} \\
\textit{enrolled}, \textit{completed} : \mathbb{P}\, \textit{Student} \\
\hline
\#\textit{enrolled} \leq \textit{MaxSize} \\
\textit{completed} \subseteq \textit{enrolled} \\
\hline
\end{array}
$$

Initially, no students are enrolled.

$$
\begin{array}{|l}
\hline
\textit{InitA} \\
\textit{ClassA}' \\
\hline
\textit{enrolled}' = \varnothing \\
\hline
\end{array}
$$

We define an operation to enrol a student in the class.

$$
\begin{array}{|l}
\hline
\textit{EnrolA} \\
\Delta \textit{ClassA} \\
s? : \textit{Student} \\
\hline
s? \notin \textit{enrolled} \\
\#\textit{enrolled} < \textit{MaxSize} \\
\textit{enrolled}' = \textit{enrolled} \cup \{s?\} \\
\textit{completed}' = \textit{completed} \\
\hline
\end{array}
$$

(a) Show that in the initial state we must have $\textit{completed}' = \varnothing$. [2 marks]

(b) Give the precondition for *EnrolA*. [2 marks]

Now, consider a concrete representation for the abstract state, using two arrays, *cl* and *ex*, and a counter *size*. The array *cl* records the students in the class, with *size* indicating how much of the array is in use. Elements in the array *ex* records whether the student recorded in the corresponding element of *cl* has completed their homework. We model the arrays by total functions from $1 \mathrel{..} \textit{MaxSize}$ to the appropriate set.

$$
\begin{array}{|l}
\hline
\textit{ClassC} \\
\textit{cl} : 1 \mathrel{..} \textit{MaxSize} \longrightarrow \textit{Student} \\
\textit{ex} : 1 \mathrel{..} \textit{MaxSize} \longrightarrow \textit{Boolean} \\
\textit{size} : 0 \mathrel{..} \textit{MaxSize} \\
\hline
\operatorname{dom} \textit{cl} = \operatorname{dom} \textit{ex} = 1 \mathrel{..} \textit{size} \\
\hline
\end{array}
$$

The retrieve relation, relating the concrete and abstract states is as follows:

$$
\begin{array}{|l}
\hline \textit{Retr} \\
\textit{ClassA} \\
\textit{ClassC} \\
\hline
\textit{enrolled} = \{i : 1 \mathrel{..} \textit{size} \bullet \textit{cl}(i)\} \\
\textit{completed} = \{i : 1 \mathrel{..} \textit{size} \mid \textit{ex}(i) = \textit{True} \bullet \textit{cl}(i)\} \\
\hline
\end{array}
$$

Initially, no part of cl is in use, so $InitA$ is implemented by $InitC$:

$$
\begin{array}{|l}
\hline \textit{InitC} \\
\textit{ClassC}' \\
\hline
\textit{size}' = 0 \\
\hline
\end{array}
$$

$EnrolA$ is implemented by $EnrolC$:

$$
\begin{array}{|l}
\hline \textit{EnrolC} \\
\Delta \textit{ClassC} \\
s? : \textit{Student} \\
\hline
s? \notin \{i : 1 \mathrel{..} \textit{size} \bullet \textit{cl}(i)\} \\
\textit{size} < \textit{MaxSize} \\
\\
\textit{cl}' = \textit{cl} \oplus \{\textit{size}' \mapsto s?\} \\
\textit{ex}' = \textit{ex} \oplus \{\textit{size}' \mapsto \textit{False}\} \\
\textit{size}' = \textit{size} + 1 \\
\hline
\end{array}
$$

(c) Explain what it means for a representation to be *adequate*, and show that this representation is adequate. [2 marks]

(d) Give the pre-condition for $EnrolC$. [2 marks]

(e) State the proof obligation for data refinement of initialisation, and prove that $InitC$ is a valid data refinement of $InitA$. [2 marks]

(f) State the proof obligation for data refinement of operations, and prove that $EnrolC$ is a valid data refinement of $EnrolA$. [5 marks]

Question 3. **[20 marks]**

The following specification statement defines the problem of determining whether two sequences, A and B, of the same length, are identical.

$$z\!:\![\,\#A = \#B\,,\; z \equiv (A = B)\,] \qquad (1)$$

where: $\#A$ is the length of A, and
$A = B \;\widehat{=}\; (\forall\, i \in 1 \mathinner{..} \#A \bullet A[i] = B[i])$.

(a) Suppose you decide (for no very good reason!) to solve this problem using an algorithm that counts the number of indices at which the two sequences agree and then compares this count with the length of A. Show that the following is a valid refinement of specification statement (1):

$$\begin{array}{l} [\![\ \textbf{var}\ c \bullet \\ \quad c\!:\![\,\#A = \#B\,,\; c = \#\{i \in 1 \mathinner{..} \#A \mid A[i] = B[i] \bullet i\}\,]; \qquad (2) \\ \quad z := (c = \#A) \\ ]\!] \end{array}$$

Indicate the refinement rule used at each step in the derivation, and give a brief justification of any applicability condition. [4 marks]

(b) Explain how a *loop invariant* can be obtained from specification statement (2), using Gries' strategy of *weakening the postcondition*, and show the resulting loop invariant. [4 marks]

(c) Derive an algorithm that satisfies specification statement (2), using the refinement calculus. Indicate the refinement rule used at each step in the derivation, and give a brief justification of any applicability condition. [12 marks]

Question 4. **[25 marks]**

Consider the problem of finding the minimum element in a non-empty array segment $A[1..n]$, specified as follows:

$$r:[\,1 \leq n \leq \#A\,,\; r = min(A[1..n])\,]$$

The following is a program which implements this specification:

$$
\begin{array}{l}
|[\ \textbf{var}\ k, m \bullet \\
\quad k, m := 1, A[1]; \\
\quad \textbf{do}\ k \neq n \rightarrow \\
\quad\quad \textbf{if}\ A[k] \leq m \rightarrow m := A[k] \\
\quad\quad [\!]\ A[k] \geq m \rightarrow \textbf{skip} \\
\quad\quad \textbf{fi}; \\
\quad\quad k := k + 1 \\
\quad \textbf{od}; \\
\quad r := m \\
]|
\end{array}
$$

Suppose you wish to transform this program into one which records the position, p, of a minimal element, rather than recording the minimum value.

(a) Explain how this data refinement can be performed using the *auxiliary variables* technique, using the coupling invariant:

$$I \;\widehat{=}\; m = A[p] \;\wedge\; 1 \leq p \leq q$$

Show the resulting program at each step. [10 marks]

(b) Show that the result of applying the auxiliary variables technique is a data refinement of the original program. I.e. show that if:

(i) $Sa \preceq_{I,\epsilon,c} Sac$,

(ii) $Sac \sqsubseteq Sac'$, and

(iii) $Sac' \preceq_{I,a,\epsilon} Sc$

then $Sa \preceq_{I,a,c} Sc$

where $Sa \preceq_{I,a,c} Sc$ iff, for all postconditions R,

$$(\exists\, a \bullet I \;\wedge\; wp(Sa, R)) \Rightarrow wp(Sc, (\exists\, a \bullet I \;\wedge\; R)) \qquad [10 \text{ marks}]$$

(c) Now, show how the same data refinement can be performed using Morgan and Gardiner's "calculation" approach. Indicate the parts of the program that need to be transformed, and show how the appropriate data refinement calculators are used to perform the necessary transformations. [10 marks]

Question 5. **[20 marks]**

A statement S is *conjunctive* if, for all postconditions R_1 and R_2,

$$wp(S, R_1 \ \wedge \ R_2) = wp(S, R_1) \ \wedge \ wp(S, R_2)$$

S is *disjunctive* if, for all postconditions R_1 and R_2,

$$wp(S, R_1 \vee R_2) = wp(S, R_1) \vee wp(S, R_2)$$

The dual, $\overline{S}$, of a statement S is defined so that, for all postconditions R,

$$wp(\overline{S}, R) = \neg wp(S, \neg R)$$

(a) Prove that assignment, **skip**, **abort** and **magic** are both conjunctive and disjunctive. [8 marks]

(b) Prove that a statement S is conjunctive iff its dual, $\overline{S}$, is disjunctive. [2 marks]

(c) Prove that $|[$ **var** $x \bullet S$ $]|$ is conjunctive if S is conjunctive, but is not disjunctive. [3 marks]

(d) Prove that $w{:}[\, P \,,\, Q \,]$ is conjunctive, but not disjunctive. [3 marks]

(e) Prove that $S_1; S_2$ is conjunctive if S_1 and S_2 are both conjunctive, and disjunctive if S_1 and S_2 are both disjunctive. [4 marks]

****