Dipartimento di Elettronica e Informazione
Politecnico di Milano
20133 Milano (Italia)
Piazza Leonardo da Vinci, 32
Tel. (39) 02-2399.3400
Fax (39) 02-2399.3411

prof. ***Carlo Ghezzi e Elisabetta Di Nitto***

Software Engineering II
Part I

2 February 2007

Last name

First name

Identification number (Matricola)

Section (specify the professor you are associated with, either Prof. Ghezzi or Prof. Di Nitto)

Notes

1. Missing identification data invalidate the exam.
2. Return **only** these pages. Extra sheets will be ignored. You may use a pencil.
3. The exam is in 2 parts. For part 1 you will not be allowed to use books or class notes. For part 2 you will be allowed to use books and class notes. The first part must be in 30 min., the second part in 55 min. The final result is the sum of the scores obtained in both parts.
4. Any use of electronic devices (computers, calculators, cell phones, …) is forbidden.
5. You cannot keep a copy of the exam when you leave the room.

Question 1 – (2 points)

Consider the following statement: "The number of errors in a program strictly decreases with successive versions. When errors are discovered, they are in fact removed by the next version of the application". Is this statement right? Give a short motivation for your answer (otherwise your answer, even if correct, does not count).

Answer

False. Error removal may introduce new errors. Furthermore, the next version may be the result of perfective or adaptive maintenance, not just corrective maintenanca. Thus errors in general do not strictly decrease in number.

Question 2 –(3 points)

1. What is a path condition in symbolic execution?

Answer: It is the condition on input variables that must be satisfied to make the path executable.

2. Consider the following program fragment

```
input (x, y);
if (x>0 and y>0)
        while x ≠ y
                if x> y
                        x= x-y;
                else    y= y-x;
```

Write the path condition for a path that executes the while loop once, going through the else part and then terminating the loop.

Answer: Let X and Y be the symbolic values for variable x and y. To enter the loop and execute the else branch it must be X>0 and Y>0 and Y>X, that is Y>X>0. Then to exit the loop it must also be X=Y-X, i.e., Y=2X. The path condition therefore is:
Y=2X and X>0

Question 3 – (2 points)

What is a pipe-and-filter architecture? Briefly describe its features, possibly identifying its strengths and weaknesses.

A pipe&filter architecture connects the output of one component to the input of the next. An example is a compiler implemented as a sequential pipeline of lexical analyzer, syntax analyzer, code generator. Another example is composition in UNIX via pipes. The advantage is the ability to redirect the output of one component to the input of a different component, if the format of data is compatible. The negative aspect is a tendency towards "batch processing".

Question 4 – (2 points)

Provide a concise definition of structural and functional testing.

Answer

Structural testing (also called white or glass box testing) tests a system based on the implementation. That is, based on what it does.

Functional testing (also called black box) tests a system based on its specification, e.e., what it is supposed to do.

Software Engineering II

2 February 2007

Last name

First name

Identification number (Matricola)

Section (specify the professor you are associated with, either Prof. Ghezzi or Prof. Di Nitto)

Part II

Question 1 (6.5 points)

Provide an Alloy specification of the following problem:
A prison is a set of cells. Each cell can contain at most 5 prisoners. Prisoners may belong to a gang. You should write a constraint such that no two prisoners are put in the same cell if they belong to different gangs.
How about a constraint that requires all members of the same gang to be put in the same cell? Would this constraint be always satisfiable?

```
module prison

sig Gang {}

sig Prisoner {
        ofGang: lone Gang
        }

sig Cell {
        members: set Prisoner
        }
        {
                #members < 6
        }


fact onePrisonerInOneCell {
        all p: Prisoner | one c:Cell |
                p in c.members
        }
```

(note that this also means that each prisoner is in a cell of the prison!)

```
fact sameGangInCell {
        all p1, p2: Prisoner, c: Cell |
                p1 != p2 && p1 in c.members && p2 in c.members implies p1.ofGang = p2.ofGang
        }
```

The following fact answers the second part of the question

```
fact allSameGangInCell {
        all p1, p2: Prisoner, c: Cell | p1 != p2 && p1.ofGang = p2.ofGang && p1 in c.members implies
        p2 in c.members
        }
```

To show a contradiction try this:

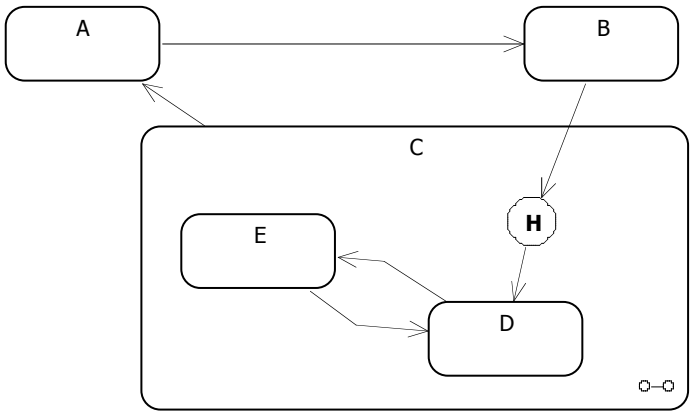
```
pred show() {}

run show  for  exactly 6 Prisoner, exactly 1 Cell, exactly 1 Gang
```

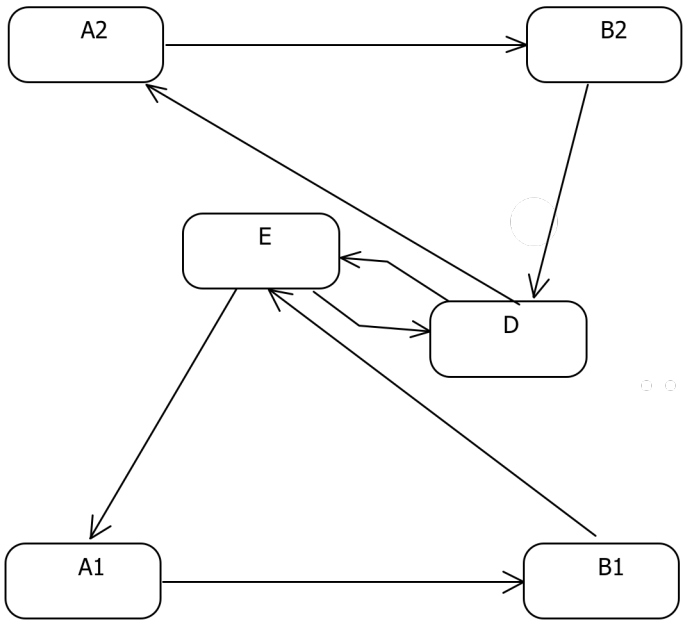
Question 2 (6.5 points)

Consider the following Statechart with a a substate an a history state.

1. Can you describe in simple words a general algorithm that generates an equivalent "plain" finite-state machine?
2. In general, supposing that you start with a Statechart with N "normal" states and one state decomposed into M substates. How many states has the resulting "plain" finite-state machine?



Solution



Sketch of a translation method:
Supposing you have only one macro state with N inner states, you have to replicate the part of the Statechart external to the macrostate N times and then connect it to each of the internal states of the macrostate. Each inner state is then connected to a copy. The copy "remembers" the inner state that was exited, so that subsequent entry into the macro state goes to the correct inner state.