classmate
Date 07/11
Page 37

TUTORIAL - 7 & 8

Q.1.) Write a Program Specification to compute greatest common divisor.

A.1.)

Pre-condition $\{ i_1 > 0$ and $i_2 > 0 \}$

Procedure P

Post-condition output $O = Z$ such that

exist z, k_1, k_2 $(i_1 = z * k_1$ and $i_2 = z * k_2)$

and

not exist y (exist k_3, k_4) $(i_1 = y * k_3$ and $i_2 = y * k_4$ and $y > z)$

Q.2.) Write a Program Specification to produce reverse of input sequence.

A.2.)

Precondition $\{ n > 0 \}$

Procedure P

Post-condition for all i $(1 \le i \le n)$, output O_i = Input I_{n-i+1}

Q.3.) Give a logic specification for a program that reads a sequence of $n+1$ values and checks whether the first value also appears in next input n values.

A3.)

$\{ n > 0 \}$

check Element $(n, i_0, \{i_1, \ldots i_n\})$

{

O, implies (exist j $(1 \le j \le n)$ and $i_0 = i_j$)

}

(Extra space)

Q4.) Give a logic specification for a program that first read two words (i.e. two sequences of alphabetic characters by a blank and terminated by '#'). The second word may be null, the first must not.

Then, the program reads a sequence of other words, seperated by blanks and terminated by '#'. and rewrites the sequences, substituting ~~the~~ all occurances of first word by second.

→

P.T.O

{ $n_1 > 2$ $n_2 > 0$

$words_{n_1}$ = '#'

exists j and $words_j$ = ' ' and $1 < j < n_1$

for all j in $1 < j < n1$ ($words_j$ in ¬[a-z A-Z \]')

$sentence_{n_2}$ = '#'

for all j in $1 \le j \le n_2$ $sentence_j$ in '[a-z A-Z \]

}

replaceWord (n_1, words, n_2, sentence)

{ $O\_sentence_{n_3}$ = '#'

$n_3 \geq 0$

firstword = words [1 ...... position of (' ')]

Secondword = words [position of (' ') . .... n1]

firstword in sentence

impues secondword in O_sentence

}

Q5.> Mention advantages and disadvantages of using FSM specifications for systems.

A5>

Advantages of FSM

1. FSM simplicity make it easy for inexperienced developers to implement with little to no extra knowledge. (low entry level)
2. Predictability (in deterministic FSM)
3. Quick to design, quick to implement & quick in execution
4. Relatively flexible. (Number of ways to implement FSM)
5. Easy to transfer meaningful abstract representation to a coded implementation.
6. Low processor overhead & easy determination of reachability of state.

Disadvantages of FSM

1. The predictable nature of deterministic FSM can be ~~difficult to manage~~ unwanted in some domains like computer games.
2. Finite memory
   > Expressive power is limited
3. State explosion
   > Given a number of FSMs with $k_1, k_2, \dots k_n$ states, their composition is a FSM with $k_1 \times k_2 \times k_3 \dots k_n$. (Exponential)
4. FSMs are essentially a synchronous model (at any time, a global state of the system, must be defined and a single transition must occur.

Q6.> Write Algebraic specification for stack operations.

algebra Stack Of Item
imports Boolean;
introduces
sort stack, Item;
operations
Create: → Stack;
IsEmpty: Stack → Boolean;
Push: Stack × Item → Stack;
Pop: Stack → Stack;
Top: Stack → Item;

create() → Stack
push(Stack, Integer) → Stack
pop(Stack) → Stack
peek(Stack) → Integer
IsEmpty(Stack) → Boolean

constraints Create, IsEmpty, Push, Pop, Top so that
Stack generated by [create, Push]

for all [S: stack, i: item]
IsEmpty (Create) = true;
IsEmpty (Push(s, i)) = false;
Pop (Create) = error;
Top (Create) = error;
Pop (Push(s, i)) = S;
Top (Push(s, i)) = i;
end Stack Of Item;

Pop (Create()) = Exception (empty stack)
pop (push(S, I)) = S
peek (create()) = exception (empty Stack)
peek (push(S, I)) = I
is Empty (create()) = true
is Empty (Push(S, I)) = false

Q7.> Compare or Differentiate PetriNets with FSMs.

A7.> We know that both Petrinets and FSM are models that represent the discrete interactions in system.
Both are not possible practically but define the behavior of system.

Here the difference b/w Petrinets (PN) and Finite State machines (FSM)

① FSM is a model that represent how single activity change its behavior over time whereelse
PN represents multi activity co-ordinates.

② In FSM, designer knows the starting Point and how the process is going, but
PW is asynchronous process in which designer does not know about starting point and how process is working. (concurrent system)

③ FSM occurs only on one system based on time, but
PN contain co-ordinate system that includes two or three processes, one depend on the previous one, so deadlock can be placed in PN.

④ FSM can be based on inputs and outputs and of two types - Mealey FSM and Moore FSM but
PN is based on places (like communication roles), transition (like transformation of objects) and tokens (like physical objects).

⑤ Petrinets are suited for timed processes whereelse FSMs are not suited for it.
