Aufgabe 1

- a LT property P_{inv} over AP is an invariant if there is a propositional formula ϕ such that

$$P_{inv} = \{ A_0 A_1 A_2 \dots \in (2^{AP})^\omega \mid \text{for all } j \geq 0 : A_j \vDash \phi \}$$

$$AP = \{ x = 0,\ x > 1,\ y = 0 \}$$

a) Initial informal Statement : " The variable x never has the value 0 and value > 1 assign at the same time"

$\Rightarrow$ At the same time, not possible to have x to be both 0 and greater than 1

LT as follow : $P = \{ \sigma \in (2^{AP})^\omega \mid \forall i \geq 0 ;\ \neg((x = 0) \wedge (x > 1)) \text{ at } i \}$

σ : set of AP (automic proposition)

Type: Safety property, ensure that x can not at the same time = 0 and greater than 1

b) Informal Statements : " The var x is only finitely often assign the value 0 and " " " a value > 1

$$P = \{ \sigma \in (2^{AP})^\omega \mid \exists n : i \geq n, (x = 0 \notin \sigma(i)) \quad \exists m : i \geq m, (y > 1 \notin \sigma(i)) \}$$

beyond the position n, $x = 0$ is not in the sequence

" " " m, $x > 1$ is " " "

Type: Safety property, not invariant because it limit the occurances but doesn't hold for all states

c) Informal Statement : " Variable x alternates between 0 and values > 1 "

$$P = \{ \sigma \in (2^{AP})^\omega \mid \forall i \geq 0, ((x = 0 \in \sigma(i)) \rightarrow (x > 1 \in \sigma(i+1))) \wedge ((x > 1 \in \sigma(i)) \rightarrow (x = 0 \in \sigma(i+1))) \}$$

if in position i is $x = 0$, then in $i+1$ should hold $x > 1$

i " $x > 1$ " " $x = 0$

Type: Liveness property: guarantees the progress and alternation

d) Informal Statement : " Variable x and y have the same value before x have the value > 1 for the first time

$$P = \{ \sigma \in (2^{AP})^{\omega} \mid \forall i < k \;,\; x = y \in \sigma(i) \}$$

k is the first position where $x > 1 \in \sigma(k)$

Type: Safety, restrict relation between x and y.

2. Verification of system properties

$AP = \{ start, cardIn, pinEntered, pinCorrect, pinNotCorrect, moneyRequested, amountCovered \}$

a) Formulate the meaning

$P_1 := \{ A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \mid \forall j \geq 0 : A_j \models \neg moneyRequested \vee pinCorrect \}$

- For every State in the ATM's operation, money can't be requested unless the correct PIN has been entered.
- Property: Safety property. It ensures that only authorized user (with correct pin) can request money

$P_2 := \{ A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \mid (\exists j \geq 0 ; A_j \models moneyRequest) \Rightarrow (\exists k > j : A_k \models Start) \}$

- If the money is requested at any point during the ATM's operation, the system muss return to the start state.
- Property: Liveness properties. Ensures, after every Transaction (succesful or not), the ATM resets to welcome state to next user

$P_3 := \{ A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \mid \exists j \leq 10 : A_j \models pinCorrect \}$

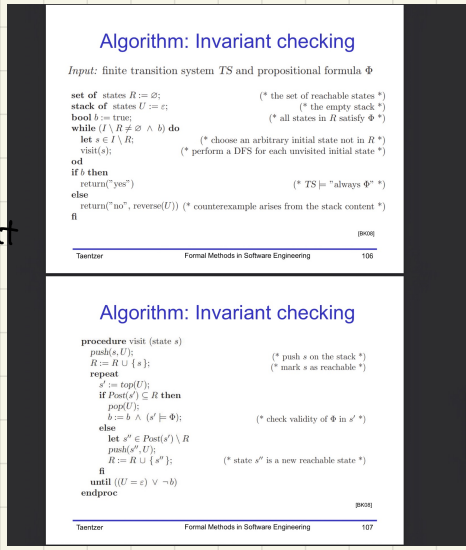
- There is only maximal 10 time to try for the correct Pin. Within 10 transition, there exist a state for correct pin.
- Property: Liveness properties. It ensures that it's possible to enter a correct Pin within 10 operations.

b) Validity of the Invariant using the algorithm

R (Reachable States) : Initially empty
U (Unvisited States) : Set of all states
s' (Current state) : Start State = { Welcome, Check, CheckAmount
s'' (next state) : Empty

Loop Iteration 1
- Start Loop
- R = { Welcome, Check, CheckAmount }
- U = σ_i / s' σ : all States
- Transition s' to s'' (insert_card)
- Update s' to s'' (s' = CardIn)

Loop Iteration 2
- Start Loop
- R = { Welcome, Check, CheckAmount, CardIn }
- U = σ_i / s' σ : all States
- Transition s' to s'' (EnterPin)
- Update s' to s'' (s' = PIN_Entered)


Algorithm: Invariant checking

Input: finite transition system TS and propositional formula Φ

set of states $R := \varnothing$; (* the set of reachable states *)
stack of states $U := \varepsilon$; (* the empty stack *)
bool $b := \text{true}$; (* all states in R satisfy Φ *)
while $(I \setminus R \neq \varnothing \ \wedge \ b)$ **do**
  let $s \in I \setminus R$; (* choose an arbitrary initial state not in R *)
  visit(s); (* perform a DFS for each unvisited initial state *)
od
if b **then**
  return("yes") (* $TS \models$ "always Φ" *)
else
  return("no", reverse(U)) (* counterexample arises from the stack content *)
fi

[BK08]

Taentzer Formal Methods in Software Engineering 106

Algorithm: Invariant checking

procedure visit (state s)
  $push(s, U)$; (* push s on the stack *)
  $R := R \cup \{s\}$; (* mark s as reachable *)
  repeat
    $s' := top(U)$;
    if $Post(s') \subseteq R$ **then**
      $pop(U)$;
      $b := b \ \wedge \ (s' \models \Phi)$; (* check validity of Φ in s' *)
    else
      let $s'' \in Post(s') \setminus R$
      $push(s'', U)$;
      $R := R \cup \{s''\}$; (* state s'' is a new reachable state *)
    fi
  until $((U = \varepsilon) \ \vee \ \neg b)$
endproc

[BK08]

Taentzer Formal Methods in Software Engineering 107

c) Validity / Invalidity of others properties

1. Money can not be requested if the correct Pin is not been entered
   → Valid: because the systems ensures that money can only be requested if the correct Pin has been entered. ATM won't allow money request without Pin.

2. If money is requested, the system must eventually return to the start state
   → Valid: When money is requested, the systems follows a sequence of transitions and return to start state.

3. There exist a state within 10 transitions where the correct pin is entered.
   → Valid: ATM only allows for Pin entry (PinEntered state) within 10 transactions, ensures that the correct pin will be entered.