Capitolo 1

Introduzione

Il *model checking* rappresenta una tecnica di verifica automatica per sistemi concorrenti a stati finiti. Esso permette di definire un modello del sistema con una notazione formale, specificare secondo una logica temporale le proprietà che tale sistema deve avere ed infine consente di verificare le specifiche definite. Poter disporre di un modello astratto del sistema, rappresenta un vantaggio di non poco conto. Poiché permette di focalizzarsi sugli aspetti implementativi, in quanto risulta possibile manipolare direttamente il modello senza dover effettuare un'implementazione vera e propria del sistema.

Di conseguenza, in presenza di errori, sarebbe necessario eseguire nuovamente la progettazione parziale oppure totale. Il processo del *model checking* si fonda su tre punti cardine: la definizione di un modello del sistema in analisi, la specifica di un insieme di proprietà mediante logiche temporali, quali *Linear Temporal Logic* (LTL) e *Computation Tree Logic* (CTL) e la verifica delle specifiche definite mediante appositi *tool*, per esempio *New Symbolic Model Verifier* (NuSMV).

NuSMV rappresenta un *tool open source* di tipo *general purpose* e automatico di verifica formale tramite *model checking*, progettato attraverso una collaborazione tra l'Istituto Trentino di Cultura di Trento (ITC-IRST, oggi Fondazione Bruno Kessler), la Carnegie Mellon University, l'Università di Genova e l'Università di Trento. Una volta definito un formalismo grafico per descrivere un modello, un punto fondamentale riguarda proprio la traduzione dal modello grafico stesso nel linguaggio supportato dal *tool* NuSMV.

L'utilizzo di quest'ultimo, diviene pertanto necessario per la verifica delle specifiche che il sistema deve rispettare. L'operazione di mappatura si basa principalmente sul meccanismo che lega gli stati, le transizioni e le azioni sugli stati; in questo modo è possibile poter descrivere l'evoluzione del sistema ad ogni *step* e per ciascuna variabile coinvolta.

In qualità di *model checker*, per NuSMV si rende necessario definire un modello di sistema secondo un formalismo proprio, ovvero quello delle macchine a stati finiti. Tale formalismo, permette di fatto di descrivere il modello di un determinato sistema; tradurre il modello nel linguaggio proprio del *model checker* e successivamente eseguire

la verifica formale del modello tradotto. L'obiettivo diviene, quindi, quello di astrarre l'utente dal formalismo utilizzato dal *tool*. Mediante questo approccio, si permette all'utente di focalizzarsi esclusivamente sulla descrizione del sistema, attraverso un solido modello di tipo logico matematico che trova la sua rappresentazione attraverso le macchine a stati finiti.

Infatti, il metodo di verifica formale implementato dal *tool* in analisi è il seguente: data una determinata proprietà da verificare, essa viene espressa attraverso la logica temporale con una formula φ e modello M. Se M, s $\models \varphi$ ovvero se M soddisfa φ.

Pertanto, se M è un modello per φ, allora il *tool* restituirà in *output* un valore *true.* Diversamente, nel caso *false,* il *tool* mostrerà a video un controesempio dell'esecuzione nel caso in cui la proprietà sottomessa sia stata violata.

Capitolo 2

Caso di studio: Bancomat (ATM)

Un bancomat o sportello automatico, *Automated Teller Machine* (ATM) rappresenta il sistema per il prelievo automatico di denaro contante dal proprio conto corrente bancario o postale, mediante l'utilizzo di una carta di debito nei distributori collegati per mezzo di una rete telematica, anche fuori dagli orari di lavoro degli istituti di credito ed in località differenti dalla sede della banca presso cui si intrattiene il rapporto di conto corrente.

Si è deciso di trattare in questo caso studio, tale sistema, per via della sua affinità con la modellazione del controllo degli accessi.

2.1 Modello del sistema

La rappresentazione fornita all'interno del presente elaborato vuole essere una semplificazione del sistema di bancomat reale. Difatti, all'interno di un sistema reale di questa tipologia è necessario tenere conto dell'eventualità nella quale si inserisca in maniera errata per tre volte consecutive il codice di autenticazione *Personal Identification Number* (PIN). La diretta conseguenza è quella nella quale, il bancomat che si sta utilizzando trattenga al suo interno, la carta di debito dell'utente. Di contro, nel modello qui presentato, non si prende in considerazione tale eventualità.

Inoltre, sebbene un sistema reale di questa tipologia, ponga dei limiti al prelievo giornaliero del contante, sulla base delle leggi vigenti e della disponibilità dello stesso denaro all'interno della macchina, il modello qui descritto non discrimina tale casualità.

Di seguito, viene riportata una possibile modellazione del sistema in analisi:

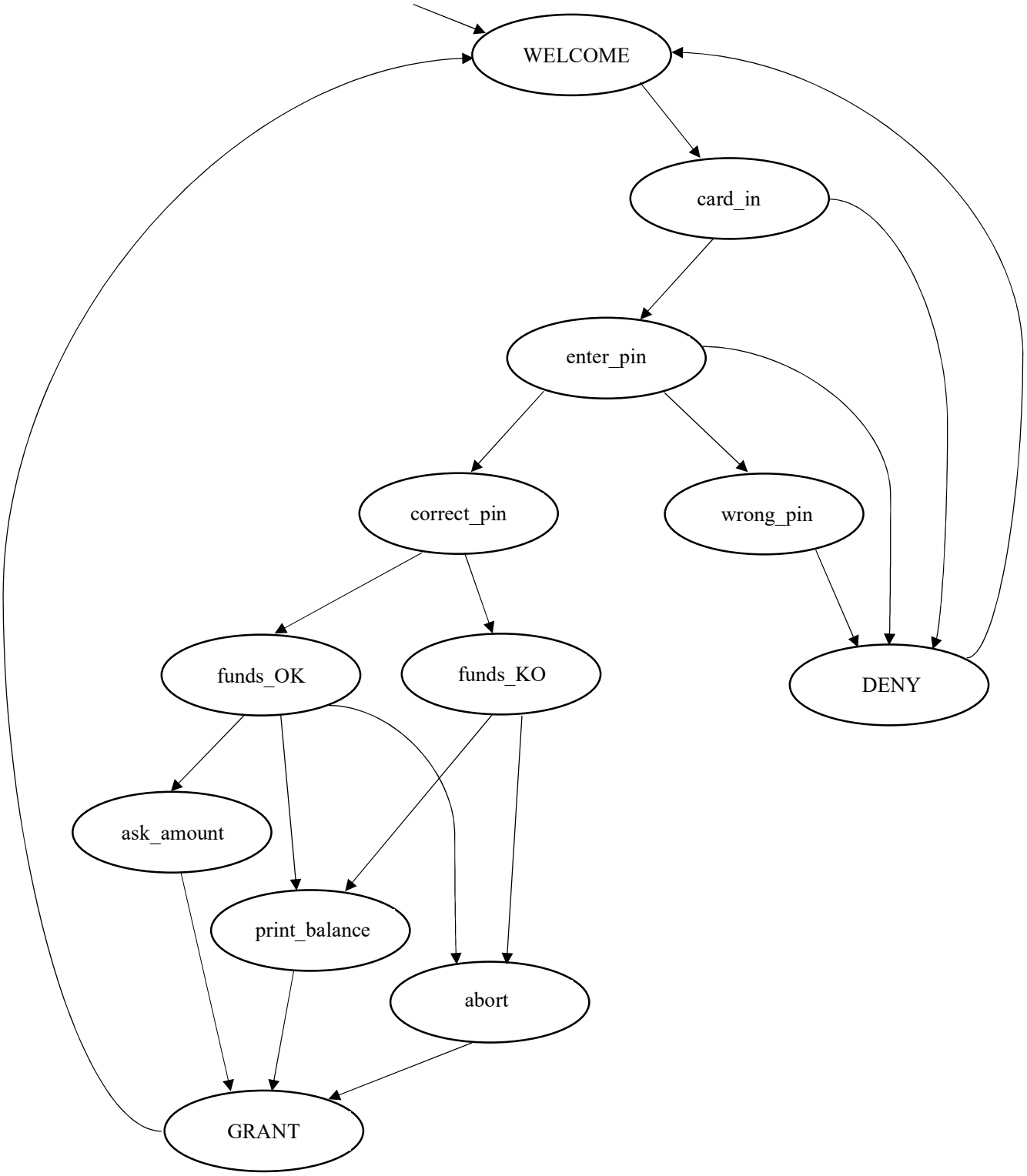

Figura 1.1 Macchina a stati finiti del sistema bancomat ATM

Le specifiche informali del sistema bancomat vengono definite attraverso i seguenti casi d'uso:

- L'utente inserisce la carta di debito all'interno del bancomat e il sistema richiede all'utente di inserire il proprio PIN autorizzativo. Pertanto, l'utente, inserisce correttamente il PIN, successivamente seleziona l'opzione di prelievo del denaro contante "*ask_amount*" e terminata l'operazione, il sistema restituisce la carta di debito all'utente.

- L'utente inserisce la carta di debito all'interno del bancomat e il sistema richiede all'utente di inserire il proprio PIN autorizzativo. Successivamente, l'utente inserisce correttamente il proprio PIN e seleziona l'opzione di interrogazione del proprio saldo monetario "*print_balance*". Conclusasi tale operazione, il sistema restituisce la carta di debito all'utente.

- L'utente inserisce la carta di debito all'interno del bancomat e il sistema richiede all'utente di inserire il proprio PIN autorizzativo. In seguito, l'utente decide di annullare l'operazione in corso, attraverso l'opzione "*abort*". Al termine dell'operazione, il sistema restituisce la carta di debito all'utente.

2.2 Specifiche formali

Vengono definite le seguenti specifiche formali in CTL per il sistema modellato in precedenza:

- Se viene inserita la carta di debito e viene inserito il PIN ed il PIN risulta corretto e sono presenti dei soldi e se ne richiede il prelievo, allora sarà possibile ottenere l'accesso:

$$AG((card\_in \wedge enter\_pin \wedge correct\_pin \wedge funds\_OK \wedge ask\_amount) \rightarrow AX(GRANT))$$

- Se viene inserita la carta di debito e viene inserito il PIN ed il PIN risulta corretto e sono presenti dei soldi oppure non sono presenti e si richiede la stampa del saldo corrente, allora sarà possibile ottenere l'accesso:

$$AG(((card\_in \wedge enter\_pin \wedge correct\_pin \wedge (funds\_OK \vee funds\_KO) \wedge print\_balance) \rightarrow AX(GRANT))$$

- Se viene inserita la carta di debito e viene inserito il PIN ed il PIN risulta corretto e sono presenti dei soldi oppure non sono presenti e si richiede l'annullamento dell'azione in corso, allora sarà possibile ottenere l'accesso:

$$AG(((card\_in \wedge enter\_pin \wedge correct\_pin \wedge (funds\_OK \vee funds\_KO) \wedge abort) \rightarrow AX(GRANT))$$

2.3 Validazione delle specifiche

Si è proceduto alla validazione delle seguenti specifiche, formalizzate in CTL, attraverso il *tool* NuSMV:

1. Se non viene inserita la carta di debito, allora sarà eventualmente possibile in futuro eseguire il prelievo:
   SPEC AG ((!card_in) -> AF(action = ask_amount));

- La seguente specifica non è verificata, difatti il *tool* riporta in *output* un controesempio:

  -- specification AG (!card_in -> AF action = ask_amount) is false
  -- as demonstrated by the following execution sequence
  Trace Description: CTL Counterexample
  Trace Type: Counterexample
  -> State: 1.1 <-
  card_in = FALSE
  enter_pin = FALSE
  state = correct_pin
  money = funds_OK
  action = print_balance
  access = WELCOME
  -- Loop starts here
  -> State: 1.2 <-
  access = DENY
  -> State: 1.3 <-

2. Se viene inserita la carta di debito e si inserisce il PIN e tale PIN risulta corretto, sono presenti dei soldi e si richiede di eseguire il prelievo, eventualmente in futuro verrà garantito l'accesso e concesso il prelievo:

   SPEC AG ((card_in = TRUE & enter_pin = TRUE & state = correct_pin & money = funds_OK & action = ask_amount) -> AF (access = GRANT));

- La seguente specifica è verificata, difatti il *tool* riporta in *output*:

  -- specification AG (((((card_in = TRUE & enter_pin = TRUE) & state = correct_pin) & money = funds_OK) & action = ask_amount) -> AF access = GRANT) is true

3. Se viene inserita la carta di debito e si inserisce il PIN e tale PIN risulta non corretto e sono presenti dei soldi e si richiede di eseguire il prelievo, il prossimo stato sarà quello nel quale sarà consentito l'accesso:

   SPEC AG ((card_in = TRUE & enter_pin = TRUE & state = wrong_pin & money = funds_OK & action = ask_amount) -> AX (access = GRANT));

- La seguente specifica non è verificata, difatti il *tool* riporta in *output* un controesempio:

  -- specification AG (((((card_in = TRUE & enter_pin = TRUE) & state = wrong_pin) & money = funds_OK) & action = ask_amount) -> AX access = GRANT) is false
  -- as demonstrated by the following execution sequence
  Trace Description: CTL Counterexample
  Trace Type: Counterexample
  -> State: 1.1 <-
  card_in = TRUE
  enter_pin = TRUE
  state = wrong_pin
  money = funds_OK
  action = ask_amount
  access = WELCOME
  -> State: 1.2 <-
  access = DENY

4. Se viene inserita la carta di debito e non viene inserito il PIN, sarà eventualmente possibile in futuro non ottenere l'accesso al sistema:

   SPEC AG ((card_in = TRUE & enter_pin = FALSE) -> AF (access = DENY));

- La seguente specifica è verificata, difatti il *tool* riporta in *output*:

  -- specification AG ((card_in = TRUE & enter_pin = FALSE) -> AF access = DENY) is true

5. Esiste almeno un percorso, nel quale, se viene inserita la carta di debito e viene immesso il PIN e tale PIN risulta corretto e si richiede la stampa del saldo corrente, allora il prossimo stato sarà quello nel quale verrà negato l'accesso:

SPEC EG ((card_in = TRUE & enter_pin = TRUE & state = correct_pin & action = print_balance) -> AX (access = DENY));

- La seguente specifica non è verificata, difatti il *tool* riporta in *output* un controesempio:

  -- specification EG ((((card_in = TRUE & enter_pin = TRUE) & state = correct_pin) & action = print_balance) -> AX access = DENY) is false
  -- as demonstrated by the following execution sequence
  Trace Description: CTL Counterexample
  Trace Type: Counterexample
  -> State: 1.1 <-
  card_in = TRUE
  enter_pin = TRUE
  state = correct_pin
  money = funds_OK
  action = print_balance
  access = WELCOME

6. Esiste almeno un percorso, nel quale se viene inserita la carta di debito e viene immesso il PIN, e questo non è corretto e si seleziona l'azione "*abort*", allora esiste almeno un percorso dove nel futuro verrà garantito l'accesso:

   SPEC EG ((card_in = TRUE & enter_pin = TRUE & state = wrong_pin & action = abort) -> EF (access = GRANT));

- La seguente specifica non è verificata, difatti il *tool* riporta in *output* un controesempio:

  -- specification EG ((((card_in = TRUE & enter_pin = TRUE) & state = wrong_pin) & action = abort) -> EF access = GRANT) is false
  -- as demonstrated by the following execution sequence
  Trace Description: CTL Counterexample
  Trace Type: Counterexample
  -> State: 1.1 <-
  card_in = TRUE
  enter_pin = TRUE
  state = wrong_pin
  money = funds_OK
  action = abort
  access = WELCOME

7. Se viene inserita la carta di debito e viene inserito il PIN e il PIN risulta corretto e non sono presenti dei soldi e si richiede di eseguire il prelievo, allora esiste almeno un percorso dove nel futuro verrà garantito l'accesso:

   SPEC AG ((card_in = TRUE & enter_pin = TRUE & state = correct_pin & money = funds_KO & action = ask_amount) -> EF (access = GRANT));

- La seguente specifica non è verificata, difatti il *tool* riporta in *output* un controesempio:

  -- specification AG (((((card_in = TRUE & enter_pin = TRUE) & state = correct_pin) & money = funds_KO) & action = ask_amount) -> EF access = GRANT) is false
  -- as demonstrated by the following execution sequence
  Trace Description: CTL Counterexample
  Trace Type: Counterexample
  -> State: 1.1 <-
  card_in = TRUE
  enter_pin = TRUE
  state = correct_pin
  money = funds_KO
  action = ask_amount
  access = WELCOME

8. Se viene inserita la carta di debito e non viene inserito il PIN e il PIN è corretto oppure è sbagliato e sono presenti dei soldi e se ne richiede il prelievo, allora sarà eventualmente possibile in futuro che l'accesso venga negato:

   SPEC AG ((card_in = TRUE & !enter_pin = TRUE & (state = correct_pin | state = wrong_pin) & money = funds_OK & action = ask_amount) -> AF (access = DENY));

- La seguente specifica è verificata, difatti il *tool* riporta in *output*:

  -- specification AG (((((card_in = TRUE & !enter_pin = TRUE) & (state = correct_pin | state = wrong_pin)) & money = funds_OK) & action = ask_amount) -> AF access = DENY) is true

9. Esiste almeno un percorso nel quale, viene inserita la carta di debito, non viene inserito il PIN e il PIN risulta corretto oppure sbagliato e sono presenti dei soldi oppure non sono presenti e si richiede di eseguire la

stampa del saldo, allora il prossimo stato sarà quello nel quale verrà negato l'accesso:

SPEC EG ((card_in = TRUE & !enter_pin = TRUE & (state = correct_pin | state = wrong_pin) & (money = funds_OK | money = funds_KO) & action = print_balance) -> AX (access = DENY));

- La seguente specifica è verificata, difatti il *tool* riporta in *output*:

-- specification EG (((((card_in = TRUE & !enter_pin = TRUE) & (state = correct_pin | state = wrong_pin)) & money = funds_OK) & action = ask_amount) -> AF access = DENY) is true

2.4 Algoritmo

È stato ideato il seguente algoritmo per rispettare sia il modello sia le specifiche formalizzate nelle pagine precedenti. Da notarsi la presenza di un unico modulo, questo in NuSMV significa che tale algoritmo si fonda su di un'unica macchina a stati finiti:

```
MODULE main
     VAR
          card_in    :  boolean;
          enter_pin  :  boolean;
          state      :  {correct_pin, wrong_pin};
          money      :  {funds_OK, funds_KO};
          action     :  {ask_amount, print_balance, abort};
          access     :  {WELCOME, GRANT, DENY};
     ASSIGN
          init(access) := WELCOME;
          next(access) := case
               card_in & enter_pin & state = correct_pin &
               money = funds_OK & action = ask_amount : GRANT;

               card_in & enter_pin & state = correct_pin &
               (money = funds_OK | money = funds_KO) &
               action = print_balance : GRANT;

               card_in & enter_pin & state = correct_pin &
               (money = funds_OK | money = funds_KO) &
               action = abort : GRANT;

          TRUE : DENY;
                          esac;

          next(card_in)    := card_in;
          next(enter_pin)  := enter_pin;
          next(state)      := state;
          next(money)      := money;
          next(action)     := action;
```