Marco Milesi – 1030184

Abstract State Machines

Realizzazione di un progetto ASM

INFORMATICA 3A
Università degli Studi di Bergamo

Introduzione

Il progetto in ASM simula il funzionamento di un distributore di bevande (acqua, caffè) e implementa 4 stati possibili e 4 input dell'utente.

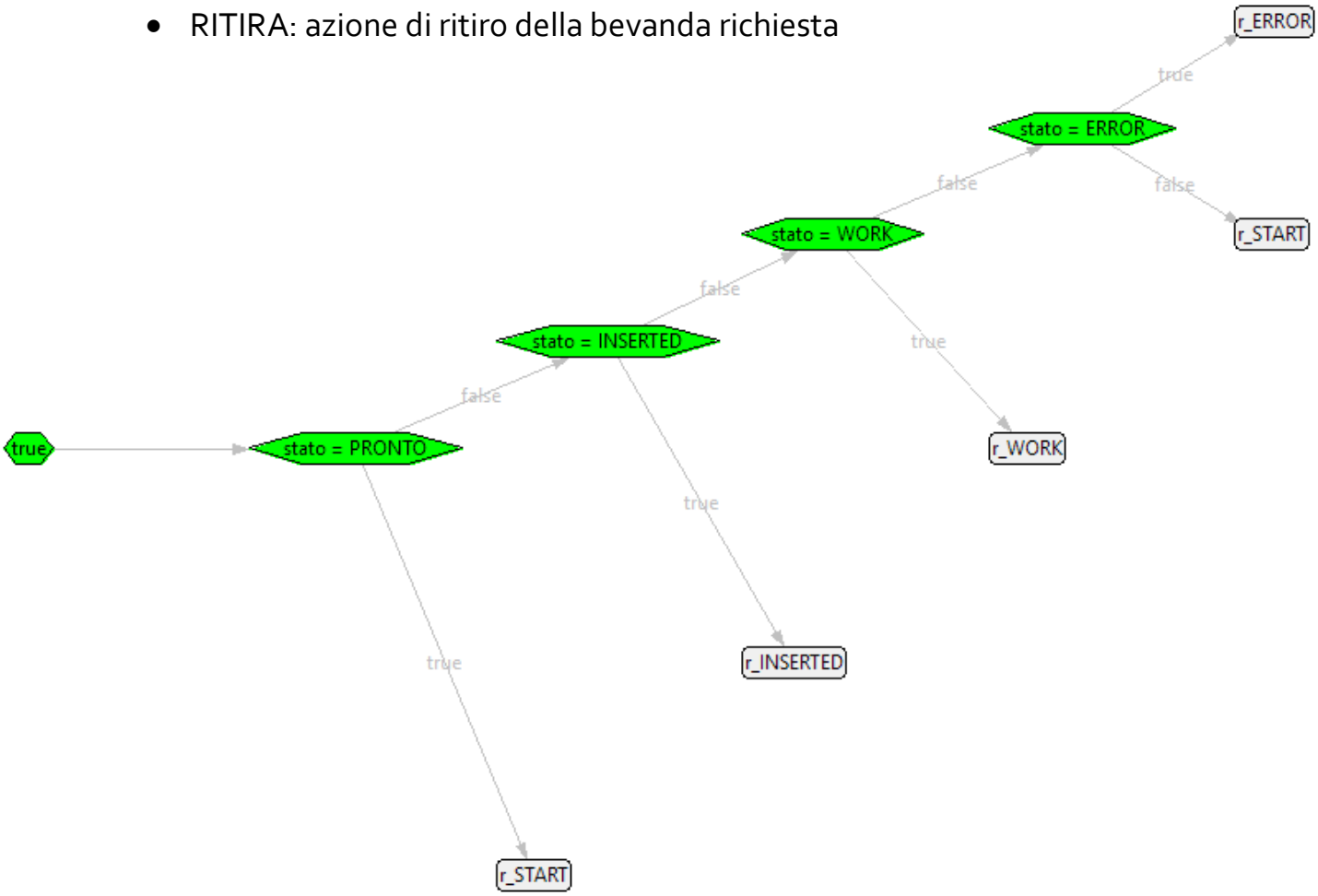
Funzionamento

La macchina a stati parte dallo stato PRONTO e può assumere i seguenti stadi:

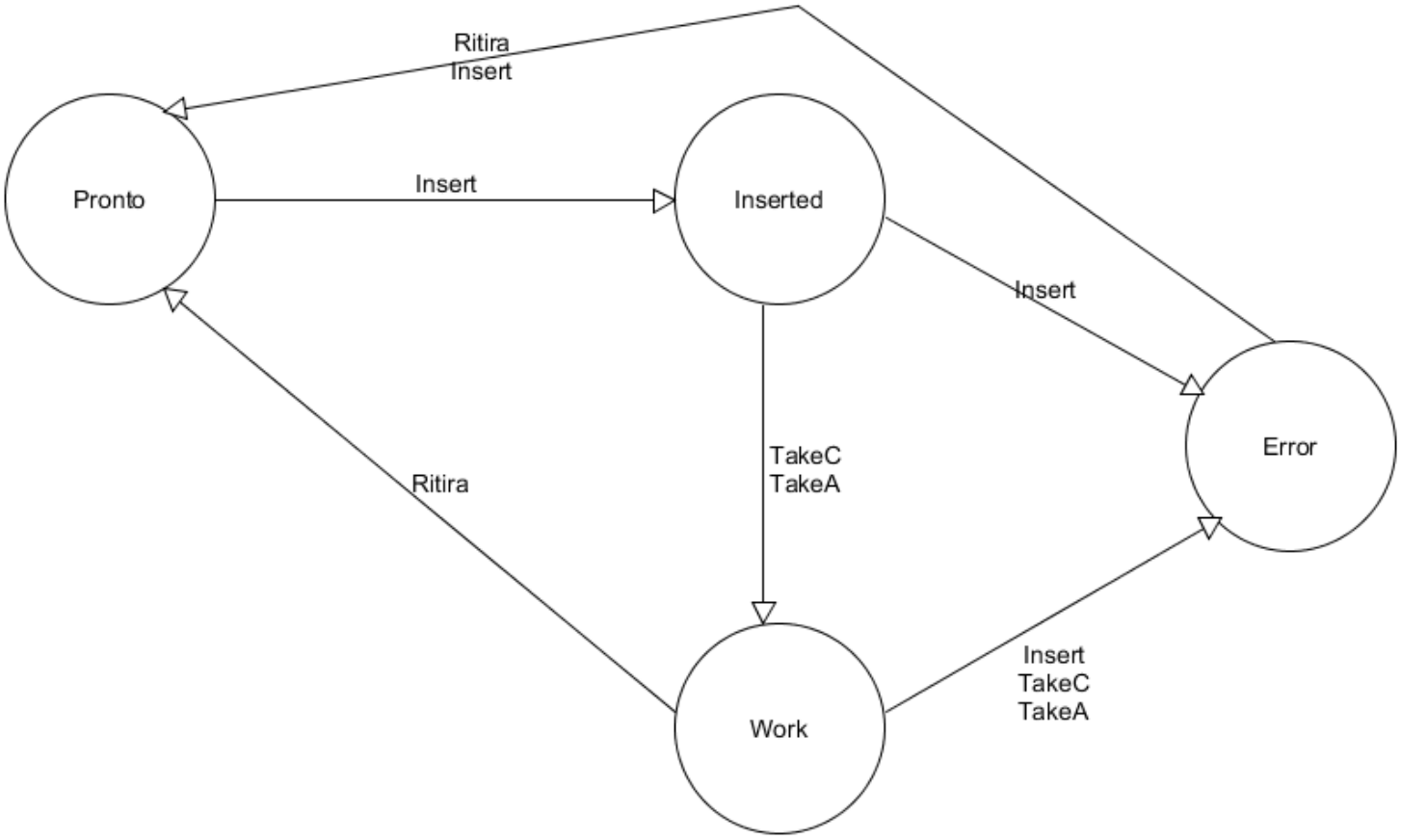
- PRONTO: stato iniziale della macchina
- INSERTED: contanti, o chiavetta, inserita nella macchina (pronta per l'erogazione)
- ERROR: stato di errore, conseguente a diverse azioni di seguito specificate
- WORK: stato di lavoro della macchina, che termina con il prelievo della bevanda

I 4 input consentiti da parte dell'utente sono:

- INSERT: inserimento dei contanti o della chiavetta
- TAKEC: pressione del pulsante per il caffè (take-caffè)
- TAKEA: pressione del pulsante per l'acqua (take-acqua)
- RITIRA: azione di ritiro della bevanda richiesta



Il diagramma degli stati associato al progetto, realizzato attraverso UMLET, è il seguente:



In particolare, sono state effettuate le seguenti assunzioni:

- Lo stato "Error" prevede la segnalazione su schermo. Ad esempio, in caso di doppio inserimento dei soldi, questi vengono rifiutati dal sistema e possono essere ritirati, oppure ignorati ripetendo l'operazione di "Insert"
- La richiesta di erogazione può essere fatta solo dopo l'inserimento di contanti o di una chiavetta, portando la macchina in "Inserted"

Scenario Tipico

[PRONTO] -> Insert ->
[INSERTED] -> TakeC -> [WORK] -> Ritira -> [PRONTO]

Output:

```
Running interactively MacchinaBevande.asm
INITIAL STATE:
Insert a symbol of Operazione in [INSERT, TAKEC, TAKEA, RITIRA] for operazione:
INSERT
<UpdateSet - 0>
stato=INSERTED
</UpdateSet>
<State 1 (controlled)>
stato=INSERTED
</State 1 (controlled)>
Insert a symbol of Operazione in [INSERT, TAKEC, TAKEA, RITIRA] for operazione:
TAKEC
<State 1 (monitored)>
operazione=TAKEC
</State 1 (monitored)>
<UpdateSet - 1>
stato=WORK
</UpdateSet>
<State 2 (controlled)>
stato=WORK
</State 2 (controlled)>
Insert a symbol of Operazione in [INSERT, TAKEC, TAKEA, RITIRA] for operazione:
RITIRA
<State 2 (monitored)>
operazione=RITIRA
</State 2 (monitored)>
<UpdateSet - 2>
stato=PRONTO
</UpdateSet>
<State 3 (controlled)>
stato=PRONTO
</State 3 (controlled)>
Insert a symbol of Operazione in [INSERT, TAKEC, TAKEA, RITIRA] for operazione:

//...
```

Codice

```
asm MacchinaBevande

import StandardLibrary

signature:

	// stati
	enum domain Stato = { PRONTO | INSERTED | ERROR | WORK }

	// input
	enum domain Operazione = { INSERT | TAKEC | TAKEA | RITIRA }



// FUNCTIONS

	dynamic controlled stato : Stato
	dynamic monitored operazione : Operazione

definitions:

	rule r_START =
		seq
			stato := PRONTO
			let ( $i = operazione ) in
				switch ( $i )
					case INSERT:
						stato := INSERTED
				endswitch
			endlet
		endseq

	rule r_INSERTED =
			let ( $i = operazione ) in
				switch ( $i )
					case INSERT:
						stato := ERROR
					case TAKEC:
						stato := WORK
					case TAKEA:
						stato := WORK
					case RITIRA:
						stato := INSERTED
				endswitch
			endlet

	rule r_WORK =
		let ( $i = operazione ) in
				switch ( $i )
					case INSERT:
						stato := PRONTO
					case TAKEC:
						stato := ERROR
					case TAKEA:
						stato := ERROR
					case RITIRA:
						stato := PRONTO
				endswitch
```

```
		endlet

	rule r_ERROR =
		let ($i = operazione) in
				switch ($i)
					case INSERT:
						stato := INSERTED
					case TAKEC:
						stato := PRONTO
					case TAKEA:
						stato := PRONTO
					case RITIRA:
						stato := PRONTO
				endswitch
		endlet

	// Main
	main rule r_Main =
		if (true) then
			if (stato = PRONTO) then
				r_START[]
			else
				if (stato = INSERTED) then
					r_INSERTED[]
				else
					if (stato = WORK) then
						r_WORK[]
					else
						if (stato = ERROR) then
							r_ERROR[]
						else
							r_START[]
					endif
				endif
		endif
		endif

default init s0:

function stato = PRONTO
```