Luca Parimbelli
Matricola: 1053142
https://github.com/lucaparimbelli/libreriaDigitale-Asmeta

Documentazione progetto Asmeta
Libreria digitale

Introduzione

Il progetto si pone l'obiettivo di utilizzare i principali costrutti di Asmeta presentati durante il corso di "Informatica III – modulo programmazione" per implementare una parte del progetto "libreria digitale" già realizzato in C++ e Scala. La parte del progetto della libreria digitale implementata in Asmeta consiste nella gestione del noleggio e della restituzione di alcuni articoli già inserti nella libreria digitale. In particolare, l'utente può scegliere se noleggiare o restituire un articolo tra 9 diversi articoli: 3 libri, 3 audio e 3 riviste.

Struttura

Il progetto è strutturato in 2 file:

- *libreria.asm*, che contiene la specifica Asmeta che realizza la logica del programma
- *scenariolibreria.avalla*, che contiene uno scenario di utilizzo del programma in cui vengono definiti dei test per le diverse funzionalità implementate nelle specifica

La specifica *libreria.asm*, una volta processata tramite il parser *AsmetaLc*, è stata testata e utilizzata principalmente mediante l'animatore *AsmetaA*. Un esempio di utilizzo di *AsmetaA* con la specifica *libreria.asm* è mostrato in *Fig. 1*.

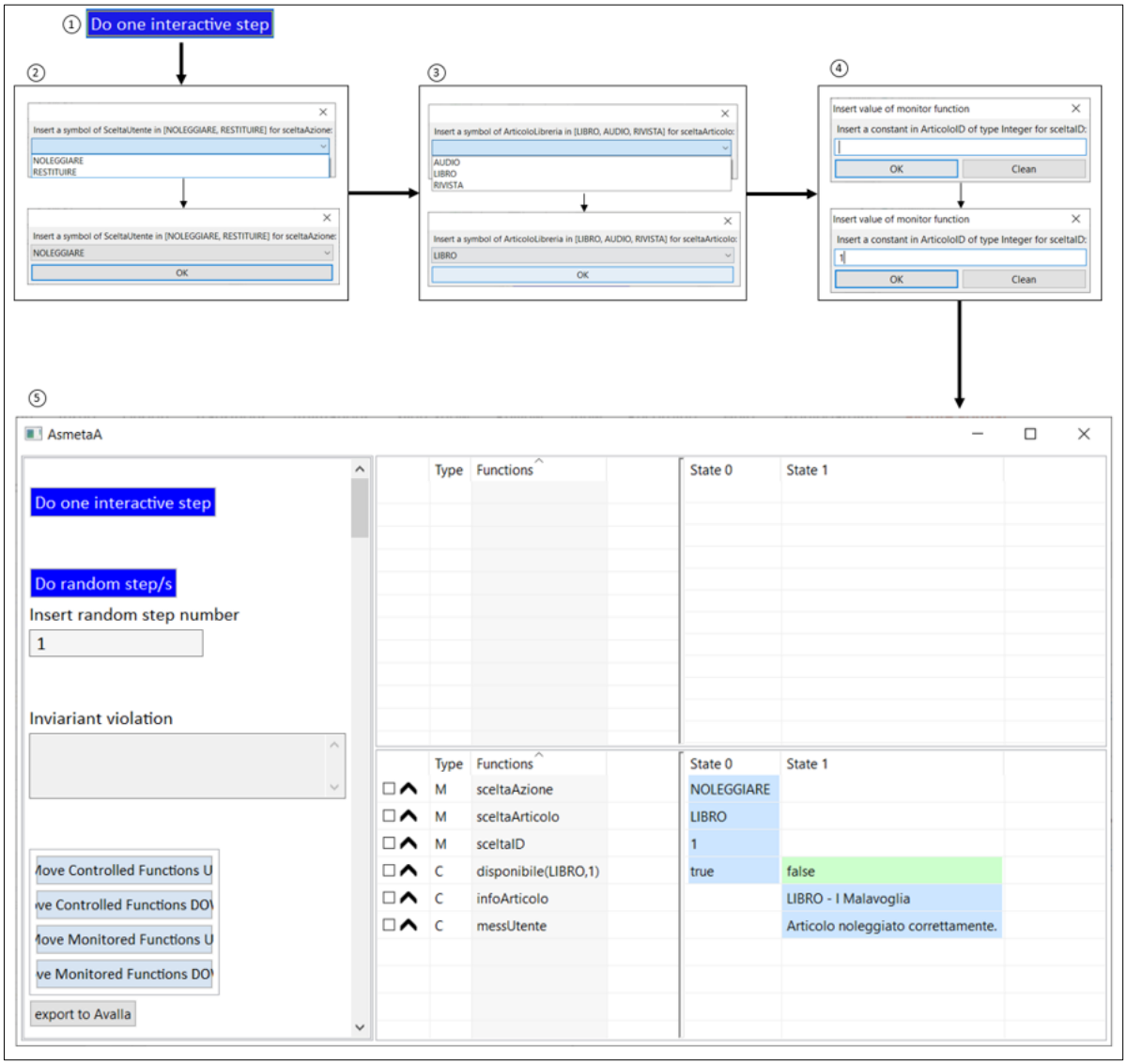

Fig. 1: AsmetaA con libreria.asm

Flow chart

Per poter rappresentare graficamente l'esecuzione della ASM, è stato redatto un flow chart (*Fig. 2*) che rappresenta gli stati in cui la ASM può trovarsi e le condizioni di transizione da uno stato all'altro.

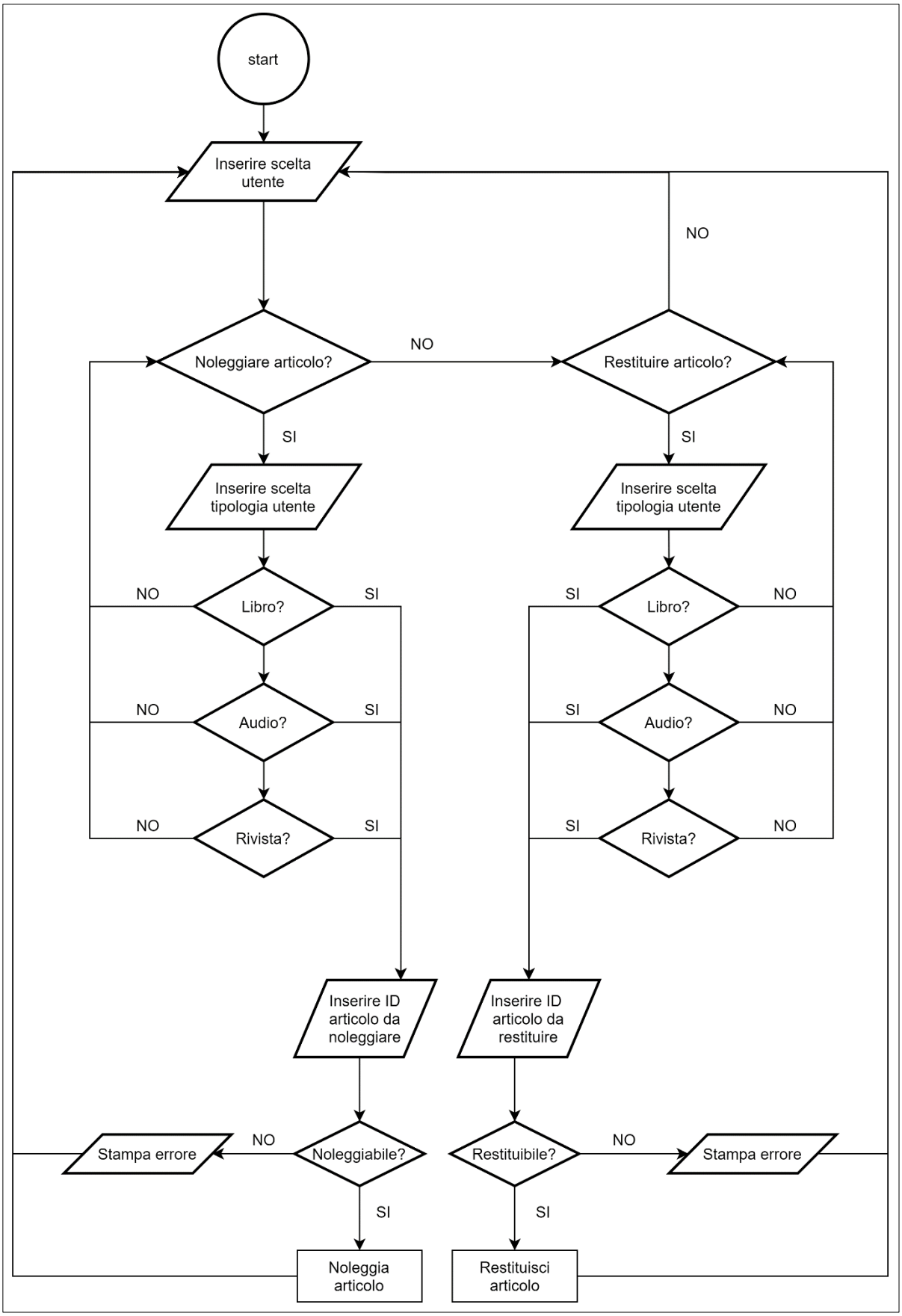

Fig. 2: flow chart per la specifica libreria.asm

Luca Parimbelli
Matricola: 1053142
https://github.com/lucaparimbelli/libreriaDigitale-Asmeta

Domini

Per poter implementare correttamente le funzionalità richieste dal programma sono stati definiti tre domini:

- *enum domain SceltaUtente*, che rappresenta il valore della prima scelta che l'utente si troverà a fronteggiare: decidere se noleggiare o restituire un articolo.
- *enum domain ArticoloLibreria*, che rappresenta la tipologia di articolo che si vuole noleggiare o restituire. Gli articoli possibili sono libro, audio oppure rivista.
- *domain ArticoloID*, che viene utilizzato per indicare l'ID dell'articolo che si vuole noleggiare/restituire. Per questioni di praticità, il programma è stato progettato con 3 diverse tipologie di articoli per ogni categoria (3 libri, 3 audio, 3 riviste).

La definizione dei domini sopracitati è riportata in *Fig. 3*.

```
 6 signature:
 7
 8 //Domains
 9     enum domain SceltaUtente = {NOLEGGIARE|RESTITUIRE}
10     enum domain ArticoloLibreria={LIBRO|AUDIO|RIVISTA}
11     //ArticoloID = ID da 1 a 3 che identifica un articolo specifico
12     domain ArticoloID subsetof Integer

29 definitions:
30     domain ArticoloID = {1:3}
```

Fig. 3: domini definiti in libreria.asm

Funzioni

In ASM è possibile definire diversi tipi di funzioni. In *libreria.asm* vengono utilizzate principalmente le seguenti tipologie di funzione:

- *monitored*, lette dalla specifica (ASM) e scritte dall'ambiente (utente)
- *controlled*, lette e scritte dalla specifica (ASM)
- *out*, scritte dalla specifica (ASM) e lette dall'ambiente (utente)
- *static*, in accordo con la documentazione ufficiale "una funzione statica è utilizzata quando vi è una fissa/immutabile connessione tra gli elementi del dominio e del codominio". Una funzione statica associa dunque a elementi del dominio degli elementi del codominio e questa relazione, essendo immutabile, non può essere modificata durante l'esecuzione.

Le funzioni che sono state definite nella specifica *libreria.asm* sono le seguenti:

- *monitored sceltaAzione*, che accetta in input un valore di tipo *SceltaUtente* e viene utilizzata per permettere all'utente di scegliere quale azione compiere (noleggiare/restituire)
- *monitored sceltaArticolo*, che accetta in input un valore di tipo *ArticoloLibreria* e viene utilizzata per permettere all'utente di selezionare una tra le 3 tipologie di articoli disponibili
- *monitored sceltaID*, che accetta in input un valore di tipo *ArticoloID* (un intero da 1 a 3) e viene utilizzata per permettere all'utente di selezionare quale articolo scegliere tra i 3 disponibili di una fissata categoria
- *controlled disponibile*, che accetta due valori in input e per questo motivo è definita come Product domain (vedi documentazione ufficiale). La funzione prende in input una coppia di valori del tipo (*ArticoloLibreria*, *ArticoloID*) e restituisce un valore *Boolean* che ne indica la loro disponibilità (true = articolo disponibile, false = articolo non disponibile)
- *static getInfoArticolo*, che è definita anch'essa come Product domain ed è una funzione statica in quanto stabilisce una relazione fissa tra uno specifico articolo [rappresentato dalla coppia (*ArticoloLibreria*, *ArticoloID*)] e una stringa che rappresenta quell'articolo. In particolare, data la tipologia di articolo e il suo ID, la funzione restituisce una stringa contenente il nome dell'articolo preso in considerazione.
- *out infoArticolo*, che è utilizzata per restituire un messaggio all'utente contente informazioni riguardanti l'articolo sul quale sono state effettuate delle operazioni
- *out messUtente*, che fornisce un messaggio all'utente circa l'esito delle operazione effettuate (positivo o negativo)

La definizione delle funzioni dell'elenco soprastante è riportata in *Fig. 4*.

```
14 //Functions
15     //valori immessi dall'utente
16     monitored sceltaAzione : SceltaUtente
17     monitored sceltaArticolo : ArticoloLibreria
18     monitored sceltaID : ArticoloID
19     //funzione che restituisce se un determinato articolo è disponibile per il noleggio o no
20     controlled disponibile : Prod(ArticoloLibreria, ArticoloID) -> Boolean
21     //funzione statica che restituisce informazioni sull'articolo. Il risultato prodotto
22     //dalla funzione verrà salvato in "infoArticolo"
23     static getInfoArticolo : Prod(ArticoloLibreria, ArticoloID) -> String
24     out infoArticolo : String
25     //messaggio restituito all'utente
26     out messUtente : String
```

Fig. 4: funzioni definite in libreria.asm

La funzione static *getInfoArticolo* è stata implementata in *definitions*: come si può osservare in *Fig. 5*, ad ogni coppia *(ArticoloLibreria, ArticoloID)* è associata una *String*.

```
29 definitions:
32     function getInfoArticolo($artName in ArticoloLibreria, $artID in ArticoloID) =
33      switch($artName, $artID)
34         case (LIBRO, 1):
35             "LIBRO - I Malavoglia"
36         case (LIBRO, 2):
37             "LIBRO - Anna Karenina"
38         case (LIBRO, 3):
39             "LIBRO - Fosca"
40         case(AUDIO,1):
41             "AUDIO - The best of Vivaldi"
42         case(AUDIO,2):
43             "AUDIO - West End Blues"
44         case(AUDIO,3):
45             "AUDIO - Beethoven Piano Concert"
46         case(RIVISTA,1):
47             "RIVISTA - Wired"
48         case(RIVISTA,2):
49             "RIVISTA - Focus"
50         case(RIVISTA,3):
51             "RIVISTA - Times"
52      endswitch
```

Fig. 5: implementazione funzione static getInfoArticolo

Regole

Con lo scopo di semplificare la scrittura e la lettura del programma, sono state definite due regole che permettono di evitare di riscrivere inutilmente codice ripetitivo. Le due regole sono:

- *rule r_noleggia*, che dato in ingresso un valore di tipo *ArticoloLibreria* e un valore di tipo *ArticoloID* prova a noleggiare tale articolo. Se l'articolo è disponibile, l'articolo viene noleggiato: la sua disponibilità assume valore false e all'utente viene restituito un messaggio di avvenuta prenotazione. Se l'articolo non è disponibile, l'utente viene avvisato di tale inconveniente.
- *rule r_restituisci*, che svolge il compito duale rispetto a quello definito da *r_noleggia*

Come si può osservare in *Fig. 6*, l'implementazione delle due regole ha richiesto l'utilizzo del blocco parallelo *par – endpar* in quanto vengono eseguite più di una regola in parallelo durante la fase di transizione della ASM.

```
54     rule r_noleggia($artName in ArticoloLibreria, $artID in ArticoloID) =
55         if(disponibile($artName, $artID) = true) then
56             par
57             disponibile($artName, $artID) := false
58             messUtente := "Articolo noleggiato correttamente."
59             infoArticolo := getInfoArticolo($artName, $artID)
60             endpar
61         else
62             par
63             messUtente := "Errore. L'articolo non e' disponibile per il noleggio."
64             infoArticolo := getInfoArticolo($artName, $artID)
65             endpar
66         endif
```

```
68     rule r_restituisci($artName in ArticoloLibreria, $artID in ArticoloID) =
69     if(disponibile($artName, $artID) = false) then
70             par
71             disponibile($artName, $artID) := true
72             messUtente := "Articolo restituito correttamente."
73             infoArticolo := getInfoArticolo($artName, $artID)
74             endpar
75         else
76             par
77             messUtente := "Errore. L'articolo e' gia' stato restituito."
78             infoArticolo := getInfoArticolo($artName, $artID)
79             endpar
80         endif
```

Fig. 6: regole r_noleggia e r_restituisci

Main rule

La main rule è la regola che viene eseguita nel momento in cui si esegue un passo di ASM. A partire dalla main rule vi sono le chiamate alle funzioni *r_noleggia* e *r_restituisci* presentate nel paragrafo precedente. La main rule implementata nel file *libreria.asm* è definita attraverso diversi switch-case che permettono di definire quale operazione compiere e su quale articolo sulla base delle scelte effettuate dall'utente. La main rule è riportata in *Fig. 7*.

```
main rule r_Main =
    switch(sceltaAzione)
        case NOLEGGIARE:
            switch(sceltaArticolo)

                case LIBRO:
                    switch(sceltaID)
                        case 1:
                            r_noleggia[LIBRO, 1]
                        case 2:
                            r_noleggia[LIBRO, 2]
                        case 3:
                            r_noleggia[LIBRO, 3]
                    endswitch

                case AUDIO:
                    switch(sceltaID)
                        case 1:
                            r_noleggia[AUDIO, 1]
                        case 2:
                            r_noleggia[AUDIO, 2]
                        case 3:
                            r_noleggia[AUDIO, 3]
                    endswitch

                case RIVISTA:
                    switch(sceltaID)
                        case 1:
                            r_noleggia[RIVISTA, 1]
                        case 2:
                            r_noleggia[RIVISTA, 2]
                        case 3:
                            r_noleggia[RIVISTA, 3]
                    endswitch

            endswitch

        case RESTITUIRE:

        switch(sceltaArticolo)

                case LIBRO:
                    switch(sceltaID)
                        case 1:
                            r_restituisci[LIBRO, 1]
                        case 2:
                            r_restituisci[LIBRO, 2]
                        case 3:
                            r_restituisci[LIBRO, 3]
                    endswitch

                case AUDIO:
                    switch(sceltaID)
                        case 1:
                            r_restituisci[AUDIO, 1]
                        case 2:
                            r_restituisci[AUDIO, 2]
                        case 3:
                            r_restituisci[AUDIO, 3]
                    endswitch

                case RIVISTA:
                    switch(sceltaID)
                        case 1:
                            r_restituisci[RIVISTA, 1]
                        case 2:
                            r_restituisci[RIVISTA, 2]
                        case 3:
                            r_restituisci[RIVISTA, 3]
                    endswitch

            endswitch

endswitch
```

Fig. 7: main rule

Inizializzazione

All'avvio del programma, la disponibilità di ogni articolo della libreria digitale è settata a true. Questo si può definire nella sezione che si occupa della definizione dello stato iniziale della ASM, ossia dopo la clausola *default init s0*. I valori della funzione *disponibile* per ogni combinazione possibile della coppia di dominio *(ArticoloLibreria, ArticoloID)* vengono settati pari a true tramite il codice mostrato in *Fig. 8*.

```
default init s0:
function disponibile($artName in ArticoloLibreria, $artID in ArticoloID) = true
```

Fig. 8: inizializzazione a true dei valori della funzione disponibile

Scenario in Avalla

Per la specifica *libreria.asm* è stato ideato uno scenario *scenariolibreria.avalla* che testa le diverse funzionalità della specifica: vengono noleggiati un libro, un audio e una rivista per poi essere tutti e tre restituiti. Nello scenario vengono anche testate le eccezioni (o meglio, i casi "eccezionali"), come ad esempio cercare di noleggiare nuovamente un articolo già noleggiato. Di seguito vengono presentati degli estratti del file *scenariolibreria.avalla* e della sua validazione mediante *AsmetaV* (*Fig. 9* e *Fig. 10*).

```
1 scenario scenarioavalla
2 load libreria.asm
3
4 //all'inizio verifico che tutti gli articoli siano disponibili
5 check disponibile(LIBRO,1) = true;
6 check disponibile(LIBRO,2) = true;
7 check disponibile(LIBRO,3) = true;
8 check disponibile(AUDIO,1) = true;
9 check disponibile(AUDIO,2) = true;
10 check disponibile(AUDIO,3) = true;
11 check disponibile(RIVISTA,1) = true;
12 check disponibile(RIVISTA,2) = true;
13 check disponibile(RIVISTA,3) = true;
14
    ...
    ...
    ...
84 //restituisco la rivista
85 check infoArticolo = "RIVISTA - Times";
86 check messUtente = "Articolo restituito correttamente.";
87 check disponibile(LIBRO,1) = true;
88 check disponibile(AUDIO,2) = true;
89 check disponibile(RIVISTA,3) = true; //controllo che la rivista sia tornata disponibile
90 step
91
```

Fig. 9: estratto del file scenariolibreria.avalla

```
** Simulation **

<Run>
<Transition>
check succeeded: disponibile(LIBRO,1) = true
check succeeded: disponibile(LIBRO,2) = true
check succeeded: disponibile(LIBRO,3) = true
check succeeded: disponibile(AUDIO,1) = true
check succeeded: disponibile(AUDIO,2) = true
check succeeded: disponibile(AUDIO,3) = true
check succeeded: disponibile(RIVISTA,1) = true
check succeeded: disponibile(RIVISTA,2) = true
check succeeded: disponibile(RIVISTA,3) = true
      ...
      ...
      ...
<Transition>
check succeeded: infoArticolo = \"RIVISTA - Times\"
check succeeded: messUtente = \"Articolo restituito correttamente.\"
check succeeded: disponibile(LIBRO,1) = true
check succeeded: disponibile(AUDIO,2) = true
check succeeded: disponibile(RIVISTA,3) = true
<State 8 (controlled)>
      ...
      ...
      ...
```

Fig. 10: validazione scenariolibreria.avalla mediante AsmetaV