Università degli Studi di Roma "Sapienza"

Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingegneria Informatica

anno accademico 2006-2007

Metodi Formali per l'Ingegneria del Software

MODELLAZIONE DEL SISTEMA TRENO-AUTO E VERIFICA DELLE PROPRIETA' TEMPORALI ATTRAVERSO NuSMV E LTL

Autori
Alfonso Calciano
Federico Covino

1. Specifiche della tesina

Effettuare il reverse engineering del sistema Treno-Auto attraverso diagrammi degli stati e delle transizioni UML. Codificare i diagrammi realizzati in moduli NuSMV e verificare infine tutte le possibili proprietà temporali del sistema attraverso l'uso della logica temporale lineare proposizionale.

2. Modello Fisico

Il sistema in esame consiste in quattro parti fondamentali: una pista percorsa dal treno; una pista percorsa dall'auto; il treno con locomotiva e rimorchio; l'automobile.

2.1. Descrizione del sistema

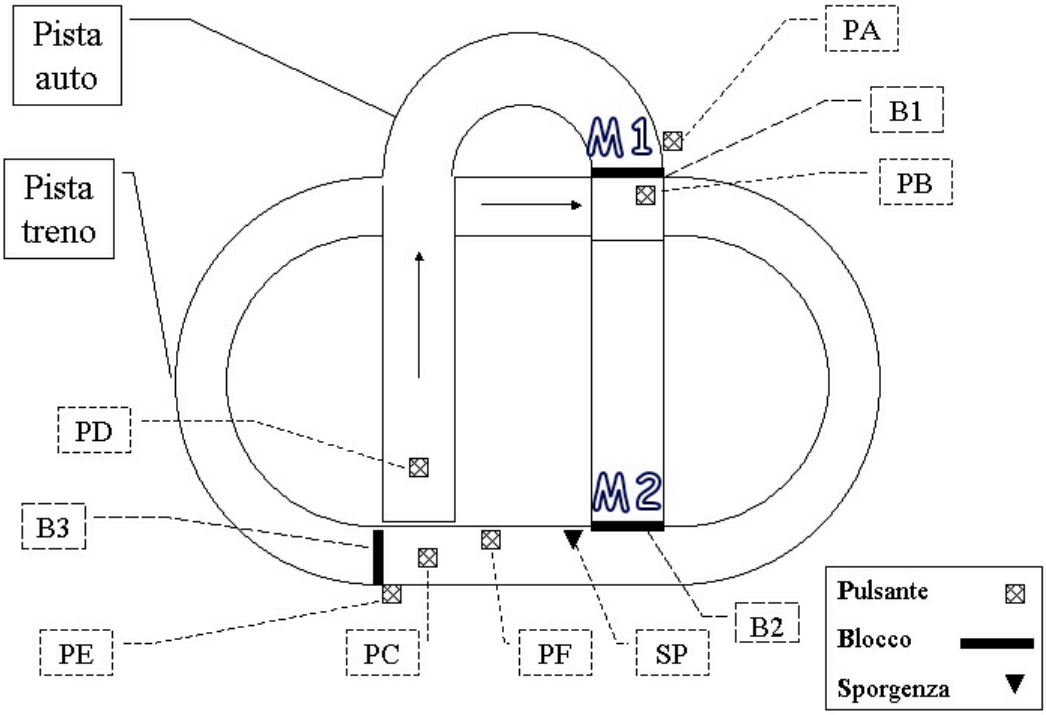
Il normale funzionamento del sistema prevede ciclicamente i seguenti passi:

Inizialmente l'auto si trova sul rimorchio del treno (posizione b3 della pista);

1. il treno percorre un giro completo della propria pista;
2. completato il giro, il treno si ferma e l'automobile scende dal rimorchio;
3. il treno riparte e l'automobile percorre metà della propria pista, superando il cavalcavia e fermandosi al primo incrocio a raso;
4. il treno supera l'incrocio a raso;
5. l'automobile riparte percorrendo la seconda metà della propria pista arrivando al secondo incrocio a raso, dove si ferma in attesa dell'arrivo del treno;
6. il treno arriva al punto di partenza del proprio percorso e si ferma in attesa che l'automobile salga sul rimorchio;
7. l'automobile sale sul rimorchio;
8. il treno riparte;

2.2. Mappa e schema dei segnali

Segue la mappa delle piste :



Il treno percorre ciclicamente la propria pista fermandosi di volta in volta nella posizione b3 per far scendere o salire l'automobile dal rimorchio. Il treno supera il blocco b3 solo in seguito a uno dei seguenti eventi: se l'automobile passa dalla posizione b2 al rimorchio; se l'automobile scende dal rimorchio (e in tal caso aziona il pulsando "PD"); se viene azionato forzatamente il comando "PE".

La posizione b1 rappresenta il primo incrocio a raso, il quale viene attraversato dall'automobile solo in seguito a due eventi: il passaggio del treno o l'azionamento del comando "PA". Facciamo notare che se si aziona il comando "PA" nel momento in cui sta passando il treno, allora si scatena un evento di *scontro* tra il treno e l'automobile.

Lo schema dei segnali fisici e dei loro effetti è riportato di seguito:

Nome pulsante	Azionato (oltre che dalle persone) da
PA	---
PB	SR
PC	ST
PD	SA
PE	---
PF	SR
PG	SP

Nome blocco	Sbloccato da
B1	PA, PB
B2	PF
B3	PC,PD, PE
B4	PG

Nome sporgenza	Si trova su
SA	Auto
SP	Pista
SR	Rimorchio treno
ST	Treno

Tale schema mostra che sia il treno sia l'automobile ricevono eventi e effettuano azioni solo da-e-verso la pista.

In particolare la pista si mantiene stabilmente in uno stato in cui tutti i blocchi sono chiusi e vengono disattivati singolarmente in modo impulsivo allo scatenarsi di qualche evento.

2.3. Problematiche spazio-temporali

Dall'analisi del funzionamento del sistema emerge una importante relazione spazio-temporale tra l'evoluzione del treno e l'evoluzione dell'automobile.

In particolare vi è una determinata tempistica tra il passaggio del treno sul primo incrocio a raso (posizione *b1* sulla pista) e il passaggio della macchina da *m1* a *m2*: nel momento in cui il treno supera la posizione *b1*, l'automobile parte raggiungendo la posizione *m2* prima che il treno arrivi sul punto *b3*. Questo rappresenta inoltre la condizione necessaria e sufficiente per la salita dell'auto sul rimorchio: nel caso in cui l'automobile raggiunga *m2* solo dopo l'arrivo del treno alla posizione *b3*, il sistema rimarrà in stato di stallo permanente.

Similmente, è possibile creare uno scontro tra treno e auto solo rispettando una particolare tempistica nell'evoluzione dei sistemi: è necessario infatti attivare il controllo "PA" (che in questo caso chiameremo controllo di "*scontro*") nel momento in cui il treno stia attraversando la posizione *b1* – ovviamente con l'automobile posizionata in *m1*.

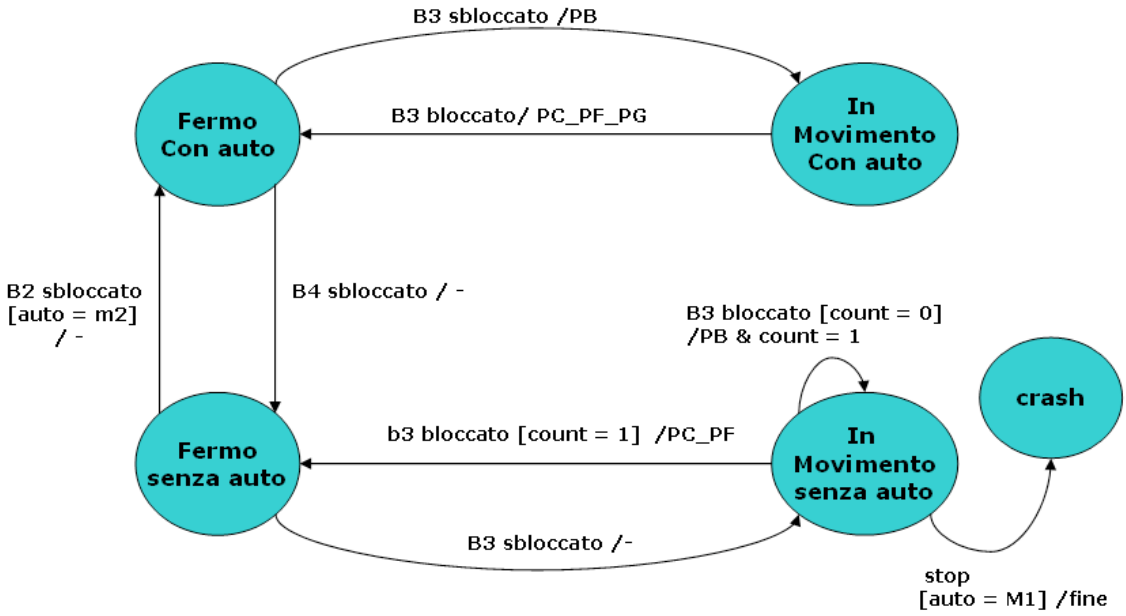
Come vedremo, ciò ha avuto forte impatto sulla modellazione dei diagrammi degli stati e delle transizioni.

3. Modellazione in D.S.T. : funzionamento normale

Per la modellazione del sistema si è scelto di ricorrere, in tutti i casi, alla definizione di tre sistemi:

- il treno con rimorchio
- la pista
- l'automobile

3.2. Treno



STATI

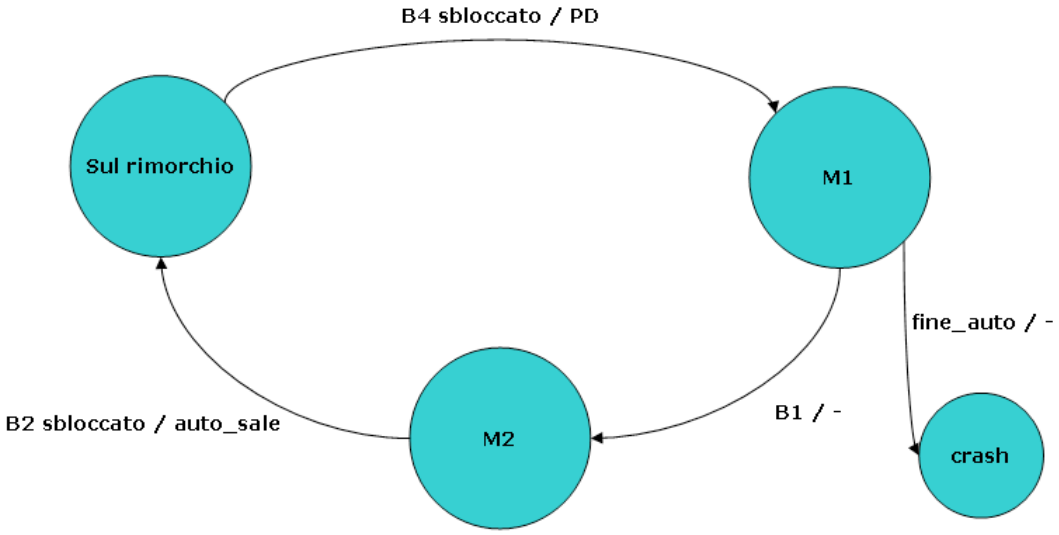
Gli stati del treno sono stati definiti in funzione di due proprietà: movimento e presenza dell'automobile. Sono quindi stati generati i quattro stati derivanti dalle possibili combinazioni. In più è stato necessario prevedere un quinto stato di crash per modellare il possibile scontro tra treno e auto.

TRANSIZIONI

Tutte le transizioni evolvono da eventi generati come azioni della pista. Allo stesso modo tutte le azioni del treno sono dirette esclusivamente al sistema Pista.

Per risolvere le problematiche spazio-temporali sopra descritte, è stato necessario creare delle dipendenze tra lo stato del treno e quello dell'automobile attraverso l'uso di *condizioni* su alcune transizioni; inoltre per modellare la dipendenza temporale tra l'arrivo del treno in *b3* e l'arrivo dell'automobile in *m2* è stato simulato un contatore modulo due che fa evolvere i sistemi Treno e Auto con una tempistica esattamente corrispondente all'evoluzione del sistema fisico reale.

3.3. Auto



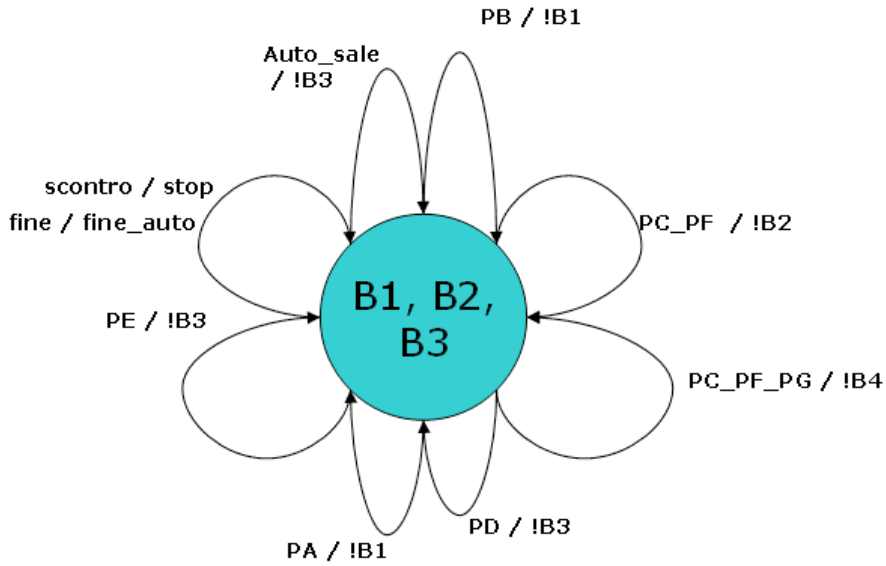
STATI

L'evoluzione del sistema auto è stata caratterizzata attraverso tre stati che descrivono esclusivamente la posizione dell'automobile. Anche qui, come per il Treno, è stato necessario introdurre un ulteriore stato per la modellazione di un eventuale scontro.

TRANSIZIONI

Tutte le transizioni evolvono da eventi generati come azioni della pista. Allo stesso modo tutte le azioni del auto sono dirette esclusivamente al sistema Pista.

3.4. Pista



Il sistema Pista rappresenta il canale di comunicazione che mappa eventi ed azioni tra treno e auto, catturando azioni dell'uno trasformandole in eventi dell'altro. Ciò ha dato forte slancio alla modularità della progettazione e ha reso possibile un'analisi accurata di tutte le proprietà temporali del Sistema nel suo complesso.

Facciamo notare quindi la presenza di un unico stato con tutte le transizioni ridotte a cappi.

EVENTI

- **PC_PF_PG**: evento generato all'arrivo del treno in posizione b3 con l'auto presente sul rimorchio, che determina il passaggio della macchina verso la posizione m1;
- **PC_PF**: generato all'arrivo del treno in posizione b3 senza l'auto sul rimorchio
- **PD**: evento generato dall'automobile nel momento in cui scende dal rimorchio
- PA: evento generato dall'esterno che sblocca il passaggio b1;
- **PE**: evento generato dall'esterno che sblocca il passaggio b3;
- **Scontro**: evento generato dall'esterno che sblocca il passaggio b1 causando lo scontro tra l'automobile e il treno;
- **Fine**: evento generato dal treno necessario per forzare l'auto allo stato di crash;
- **Auto_sale**: evento generato dall'auto nel passare da m2 al rimorchio;
- **PB**: evento generato dal treno al passaggio sulla posizione b1;

3.5. Codifica NuSMV

3.5.1. Moduli treno, auto, pista

Per codificare il sistema in NuSmv abbiamo previsto la creazione di tre moduli,uno per ogni sottosistema sincrono(treno,pista,auto) più il modulo main che funge da cliente dei precedenti e fornisce loro gli eventi. Riportiamo di seguito il codice dei vari sottosistemi.

MODULO TRENO

```
MODULE treno

VAR
stato :{fermo_con_auto, fermo_senza_auto, mov_con_auto, mov_senza_auto, crash};
evento : {b1,b2, b3, b4, stop, null, fine_auto};
azione : {pb, pc_pf_pg, pc_pf,fine, null};
count : {0,1};
a: auto;


TRANS
case

stato = fermo_con_auto & evento = b3 : next(stato) = mov_con_auto &
next(azione) = pb;

stato = fermo_con_auto & evento = b4 : next(stato) = fermo_senza_auto &
next(azione) = null;

stato = fermo_senza_auto & evento = b2 & a.stato = m2 : next(stato) =
fermo_con_auto & next(azione) = null;

stato = fermo_senza_auto & evento = b3 : next(stato) = mov_senza_auto &
next(count) = 0 &  next(azione) = null;

stato = mov_con_auto  & evento != b3 : next(stato) = fermo_con_auto &
next(azione) = pc_pf_pg;

stato = mov_senza_auto  & evento = stop & a.stato = m1 : next(stato) = crash &
next(azione) = fine;

stato = mov_senza_auto  & evento != b3 & count = 0 : next(stato) =
mov_senza_auto & next(count) = 1 & next(azione) = pb;

stato = mov_senza_auto  & evento != b3 & count = 1 : next(stato) =
fermo_senza_auto & next(count) = 0 & next(azione) = pc_pf;

1: next(stato) = stato & next(azione) = null;

esac
```

La codifica delle *condizioni* sullo stato dell'auto è stata realizzata sfruttando l'object orientation di NuSMV, dichiarando un variabile di tipo auto all'interno del modulo Treno.

MODULO AUTO

```
MODULE auto

VAR
      stato  : {m1, m2,sul_rimorchio, crash};
      evento : {b1, b2, b3, b4,fine_auto, null,pc_pf,pc_pf_pg , stop};
      azione : {pd,auto_sale, null};

TRANS
case

stato = sul_rimorchio & evento = b4 : next(stato) = m1 & next(azione) = pd;

stato = m2 & evento = b2 : next(stato) = sul_rimorchio & next(azione) =
auto_sale;

stato = m1 & evento = b1 : next(stato) = m2 & next(azione) = null;

stato = m1 & evento = fine_auto : next(stato) = crash & next(azione) = null;

1: next(stato) = stato & next(azione) = null;

esac
```

MODULO PISTA

```
MODULE pista
VAR
      stato  :  {s};
      evento : {auto_sale, pb,pc_pf,pc_pf_pg,pd,pa,pe,scontro,fine, null};
      azione : {b3,b2,b1,b4,stop,fine_auto, null};
TRANS
case

stato = s & evento = auto_sale : next(stato) = s & next(azione) = b3;

stato = s & evento = pc_pf  : next(stato) = s & next(azione) = b2;

stato = s & evento = pc_pf_pg : next(stato) = s & next(azione) = b4;

stato = s & evento = pd : next(stato) = s & next(azione) = b3;

stato = s & evento = pa : next(stato) = s & next(azione) = b1;

stato = s & evento = pe : next(stato) = s & next(azione) = b3;

stato = s & evento = pb : next(stato) = s & next(azione) = b1;

stato = s & evento = fine : next(stato) = s & next(azione) = fine_auto;

stato = s & evento = scontro : next(stato) = s & next(azione) = stop;

1: next(stato) = stato & next(azione) = null;

esac;
```

3.5.2. Modulo main

Il modulo main definisce lo stato iniziale e, tramite la funzione di transizione, specifica la legge con cui vengono generati gli eventi per ogni sottosistema.

```
MODULE main

VAR
      t: treno;
      p: pista;

ASSIGN

      init(t.count) := 0;
      init(t.stato) := ...;
      init(t.a.stato) := ...;
      init(p.evento) := ...;
      init(t.evento) := ...;
      init(t.a.evento) := ...;

      --sincronizzazione dei sistemi
      next(t.a.evento) := next(p.azione);
      next(t.evento) := next(p.azione);

TRANS
case

next(t.azione) = pc_pf :  next(p.evento) = pc_pf;

next(t.azione) = fine :  next(p.evento) = fine;

next(t.azione) != pc_pf & next(t.azione) != null :   next(p.evento) =
next(t.azione) ;

next(t.a.azione) !=  null :   next(p.evento) = next(t.a.azione) ;

--evitiamo stati impossibili
t.stato = crash : t.a.stato = crash;

t.a.stato = crash : t.stato = crash;
----------------------------

1: next(p.evento) = null;

esac
```

PROBLEMATICA NELLA CODIFICA

Il sistema pista riceve eventi sia dal Treno che dall'Auto. Nell'inserire l'inizializzazione

```
next(p.evento) := {next(t.azione) , next(t.a.azione)};
```

abbiamo riscontrato il seguente problema: la pista processa l'azione `null` nel caso in cui sia generata da uno solo dei sistemi, andando quindi ad ignorare l'azione non nulla dell'altro sistema. Per ovviare a tale limite abbiamo inserito la sezione TRANS che forza la pista a non ignorare in nessun caso azioni non `null`.

3.6. Model Checking

Nella sezione LTLSPEC abbiamo verificato diverse famiglie di proprietà temporali del sistema globale. In particolare è stato testato il funzionamento corretto, la presenza di possibili stalli o crash a partire da vari stati iniziali, sfruttando quindi NuSMV come Model Checker.

Riportiamo di seguito il codice LTL di alcune proprietà verificate:

- Corretto funzionamento:

```
F (t.a.stato = sul_rimorchio);
G F (t.a.stato = sul_rimorchio) & G F (t.a.stato != sul_rimorchio);
```

- Presenza stalli

```
G !( F G (t.stato = mov_senza_auto) | F G (t.stato = mov_con_auto) | F G
(t.stato = fermo_con_auto) | F G(t.stato = fermo_senza_auto) | F
G(t.a.stato = sul_rimorchio ) | F G(t.a.stato = m1 ) | F G(t.a.stato = m2
) )
```

- Mancanza di stalli:

```
F G (t.stato = mov_senza_auto) | F G (t.stato = mov_con_auto) | F G
(t.stato = fermo_con_auto) | F G(t.stato = fermo_senza_auto) | F
G(t.a.stato = sul_rimorchio ) | F G(t.a.stato = m1 ) | F G(t.a.stato = m2
)
```

- Prima o poi si verifica il crash

```
F (t.stato = crash & t.a.stato = crash )
```

3.7. Model Finding

Come ultima analisi sono state testate diverse proprietà temporali senza assegnare alcuno stato iniziale: in tal modo NuSMV è stato sfruttato per verificare la validità delle proprietà temporali in tutti i cammini sulla struttura temporale definita.

Riportiamo di seguito il codice LTL di alcune proprietà verificate:

- Il sistema va sempre in stallo

```
G  ( F G (t.stato = mov_senza_auto) | F G (t.stato = mov_con_auto) | F G
(t.stato = fermo_con_auto) | F G(t.stato = fermo_senza_auto) | F
G(t.a.stato = sul_rimorchio ) | F G(t.a.stato = m1 ) | F G(t.a.stato = m2
) )
```

- L'auto scende e sale dal rimorchio infinite volte

```
G F (t.a.stato = sul_rimorchio) & G F (t.a.stato != sul_rimorchio)
```

- il sistema va in crash

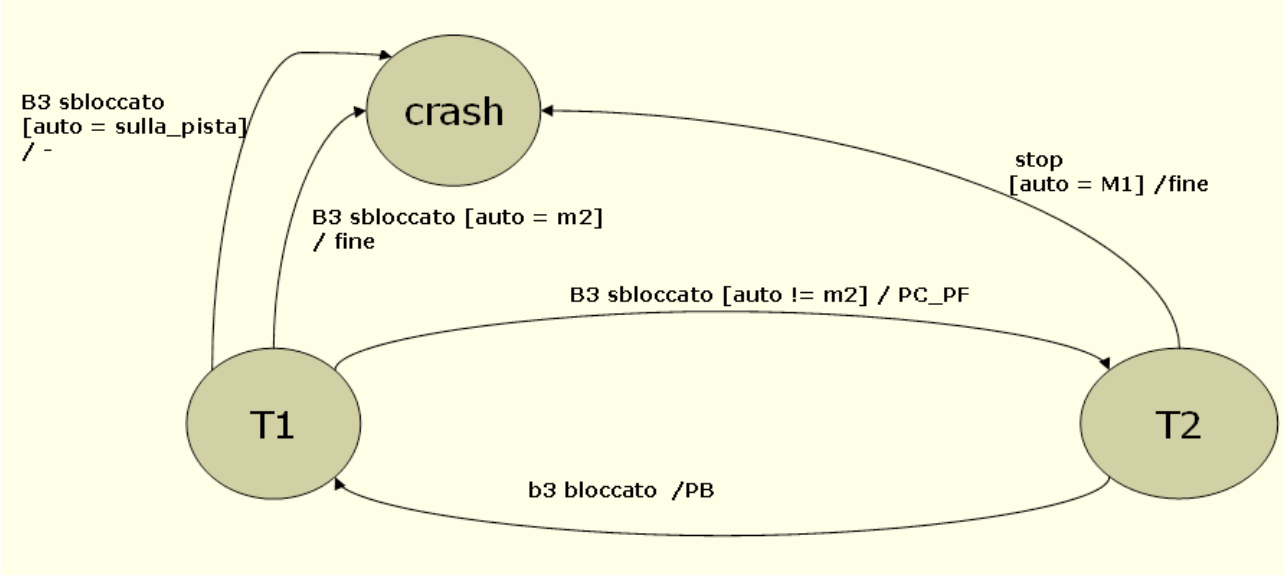
```
G  F (t.stato = crash &  t.a.stato = crash)
```

4. Modellazione in D.S.T. : treno invertito

Le potenzialità di NuSMV possono essere sfruttate per analizzare l'evoluzione del sistema anche nel caso in cui supponiamo di posizionare al contrario il treno sui binari della pista.

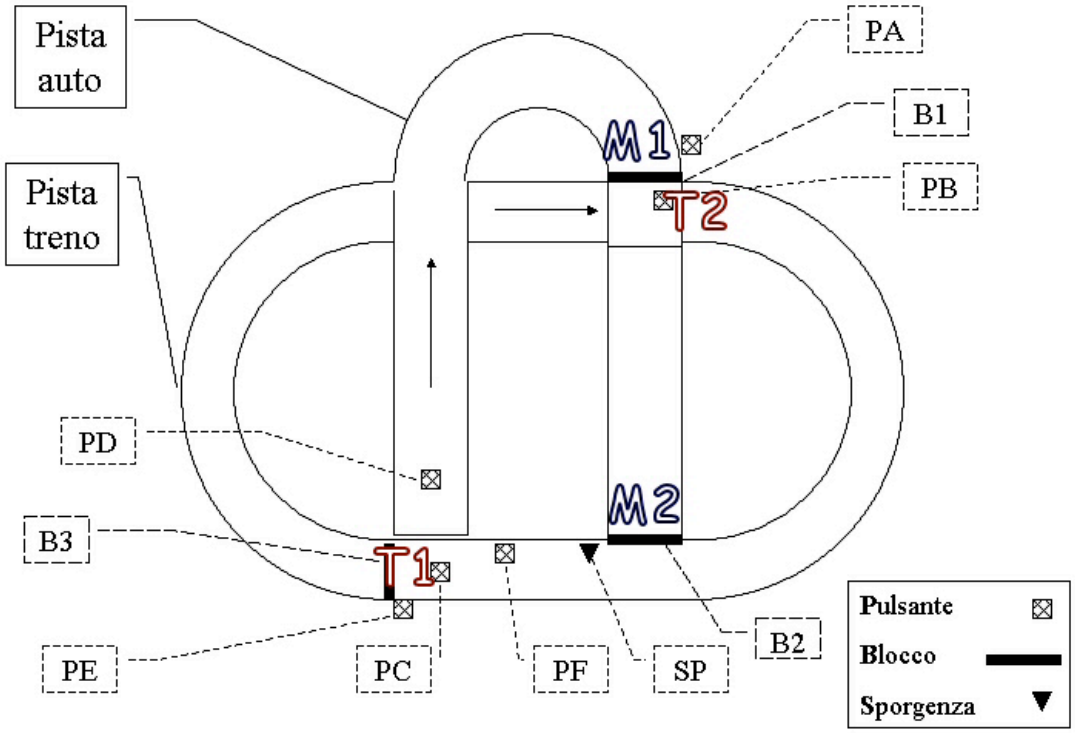
In tal caso le interazioni tra Treno e Auto cambiano notevolmente: siamo quindi interessati ad analizzare se esistono altri tipi di stalli, altri crash tra i due sistemi, se l'automobile è in grado di salire o scendere dal rimorchio etc.

4.2. Treno



STATI

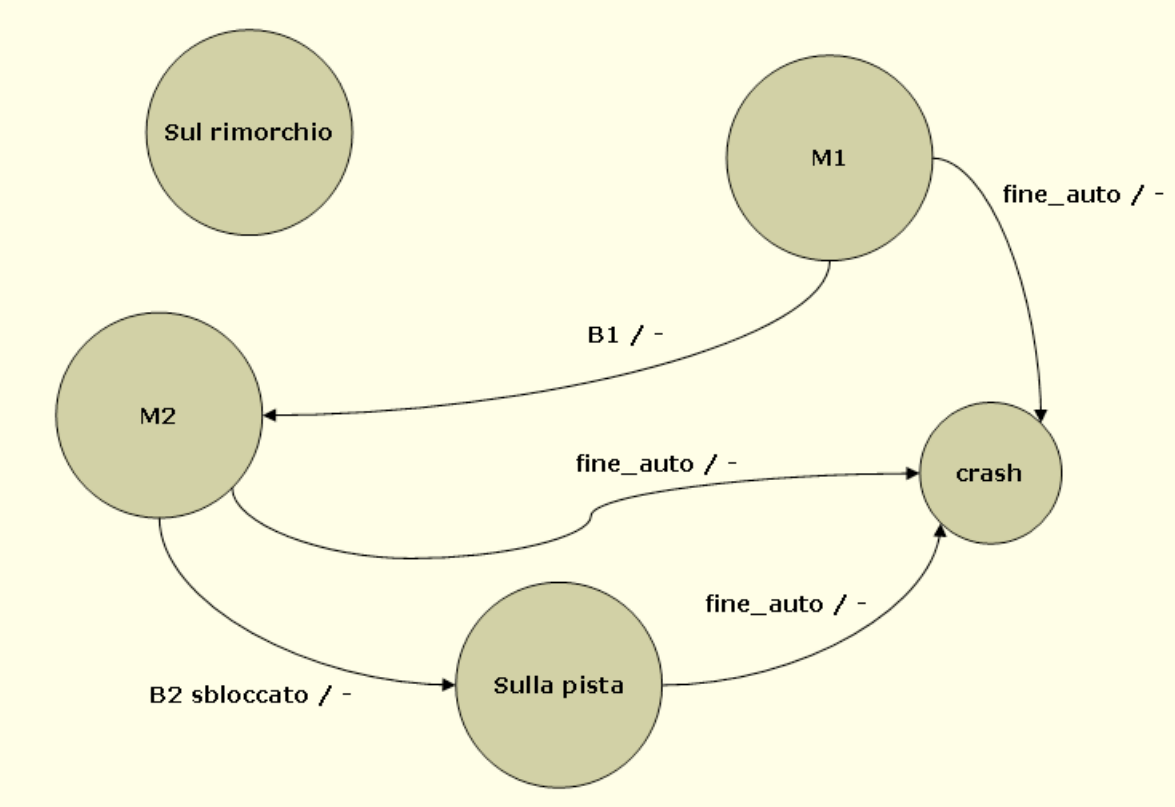
Gli stati principali sono due. *T1* indica che il treno è fermo alla sinistra del blocco *b3*, mentre *T2* indica che il treno è in movimento in senso inverso e sta quindi percorrendo la pista.
Un terzo stato è stato aggiunto anche in questo caso per modellare eventuali scontri tra treno e auto. Per chiarezza segue la mappa della pista:



TRANSIZIONI

Tutte le transizioni evolvono da eventi generati come azioni della pista. Allo stesso modo tutte le azioni del treno sono dirette esclusivamente al sistema Pista.
Rimane necessario, anche in questo caso, l'uso di *condizioni* per modellare diversi rapporti causa-effetto tra lo stato del treno e quello dell'automobile.

4.3. Automobile



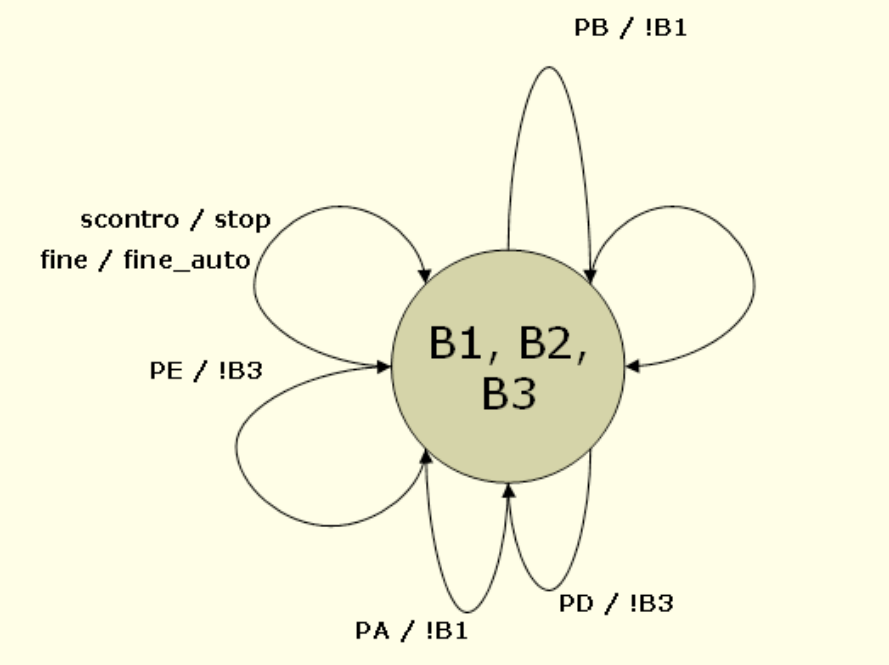
STATI

Gli stati denotano la posizione dell'automobile nel Sistema. In questo secondo caso di modellazione è stato introdotto un nuovo stato *sulla_pista* per catturare in modo più aderente al sistema fisico un nuovo scenario di scontro tra auto e trneo.
Interessante notare che ovviamente rimane possibile posizionare manualmente la macchina sul rimorchio del treno.

TRANSIZIONI

Alcune transizioni rimangono uguali al caso del funzionamento normale. Da sottolineare il passaggio dalla posizione *m2* allo stato *sulla_pista* al verificarsi dell'evento "b2 sbloccato".
Interessante notare come lo stato *sul_rimorchio* sia privo di archi entranti o uscenti.

4.4. Pista



Il sistema pista rimane invariato dal punto di vista degli stati. Le differenze nelle transizioni consistono sostanzialmente nella mancanza di alcune di esse rispetto al caso del funzionamento normale.

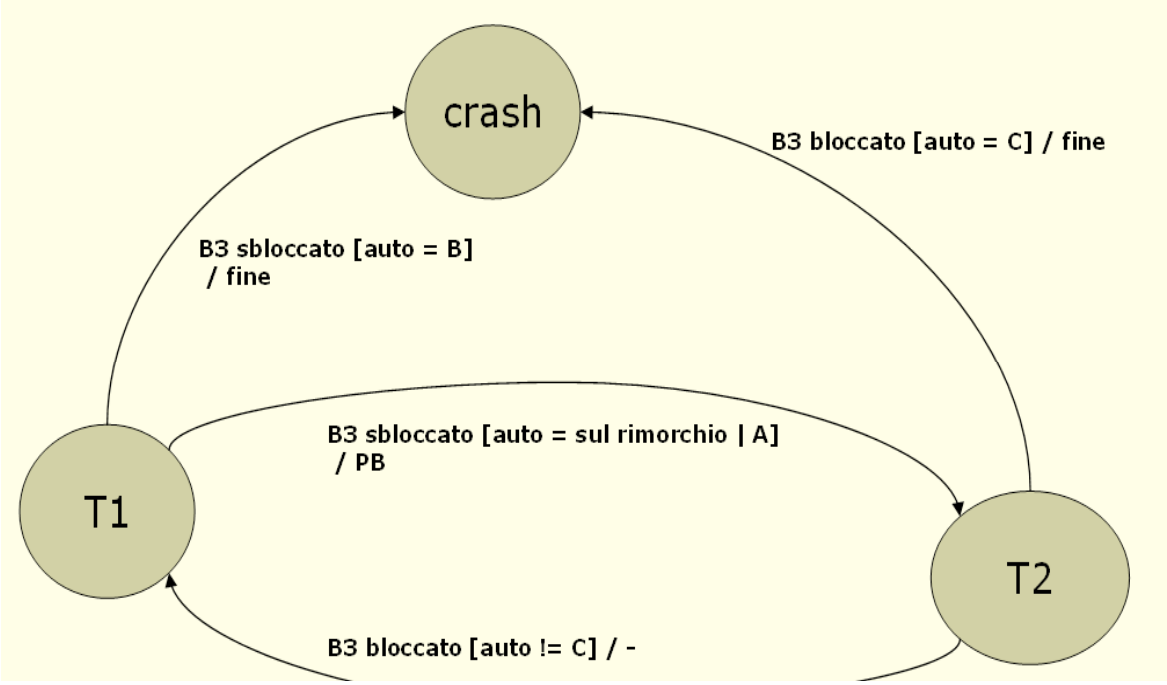
4.5. Codifica NuSMV

Riportiamo di seguito alcune delle proprietà verificate attraverso l'uso di NuSMV sia come model checker che "model finder"

- L'auto prima o poi sale sul rimorchio
  ```
  F (t.a.stato = sul_rimorchio)
  ```
- il sistema va sempre in crash
  ```
  G F (t.stato = crash & t.a.stato = crash)
  ```
- il sistema va sempre in stallo
  ```
  G  ( F G(t.stato = t1) | F G (t.stato = t2)  |  F G(t.a.stato = m1 ) | F
  G(t.a.stato = m2 ))
  ```
- il sistema recupera dal crash
  ```
  G ( (t.stato = crash & t.a.stato = crash) -> F (t.stato != crash &
  t.a.stato != crash) )
  ```

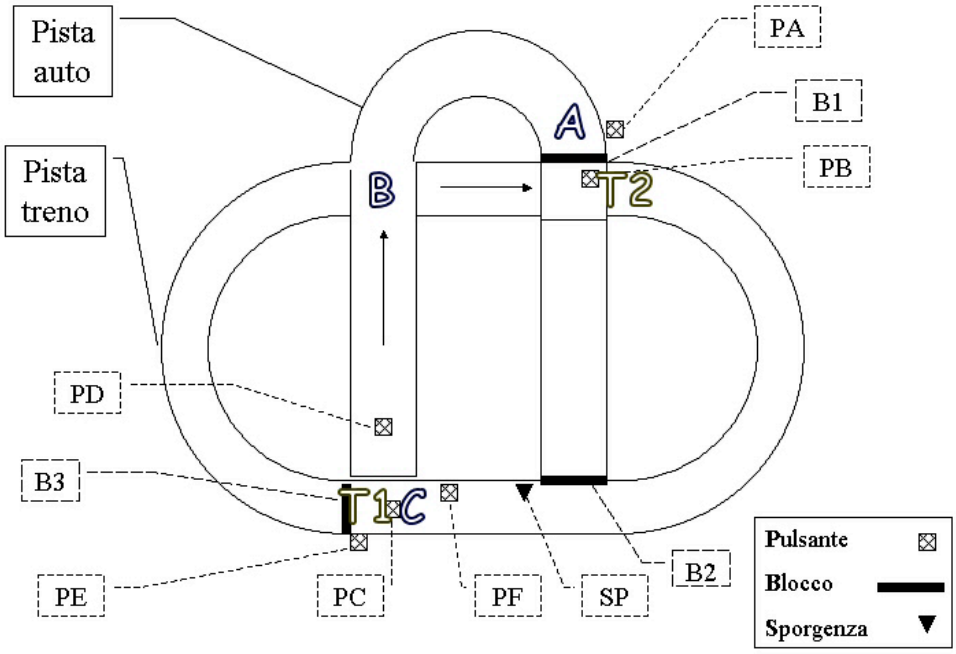
5. Modellazione in D.S.T. : automobile invertita

5.2. Treno



STATI

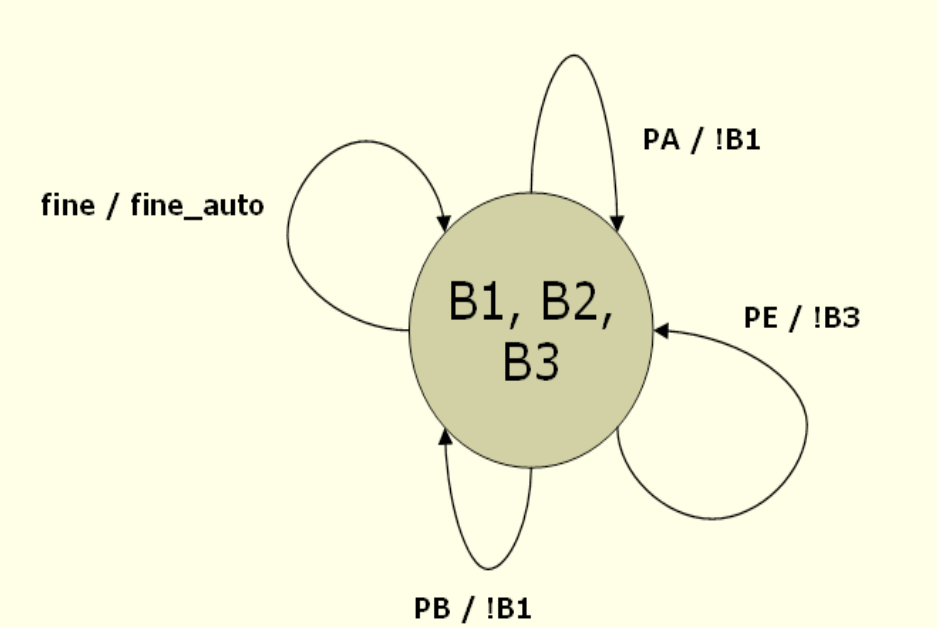
Gli stati principali sono due. *T1* indica che il treno è fermo alla destra del blocco *b3*, mentre *T2* indica che il treno è in movimento e sta quindi percorrendo la pista.
Un terzo stato è stato aggiunto anche in questo caso per modellare eventuali scontri tra treno e auto. Per chiarezza segue la mappa della pista:



TRANSIZIONI

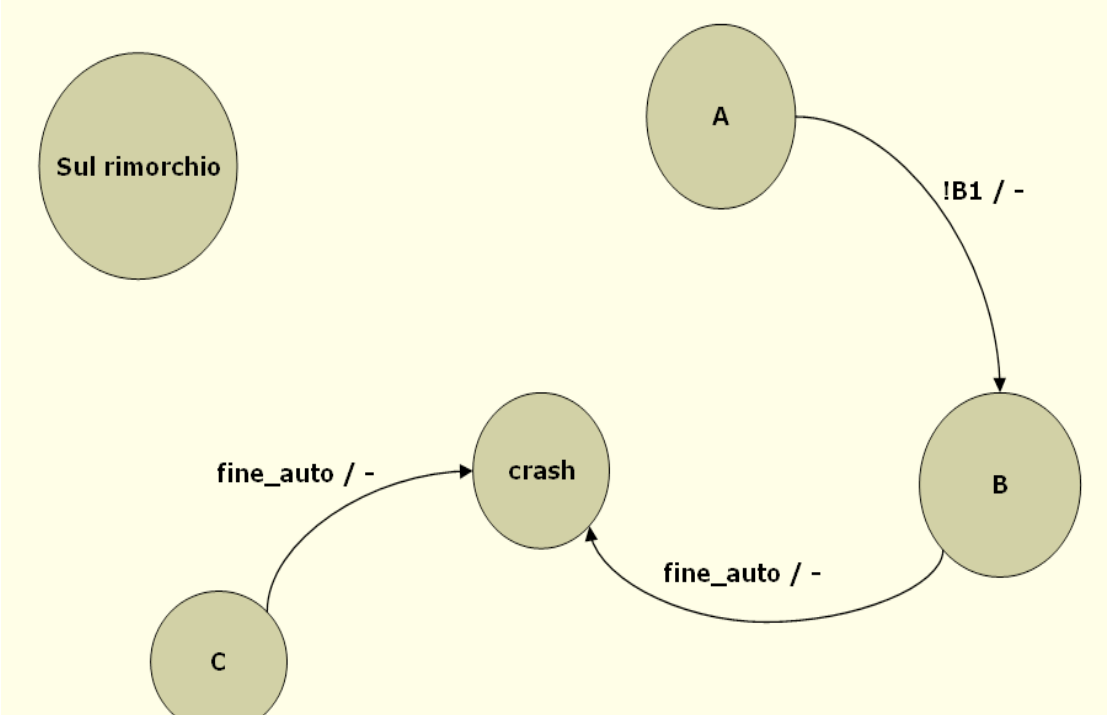
In questo caso è stato utilizzato un maggiore livello di astrazione dal sistema reale rispetto la modellazione del funzionamento normale. Le azioni del treno sono esclusivamente quelle di interesse in uno scenario in cui l'automobile viene fatta procedere in senso inverso, in particolare: l'azione *fine* forza l'intero sistema – nelle dovute condizioni – ad andare in uno stato di "crash"; l'azione *PB* è di interesse per l'interazione indiretta col sistema Auto; azione nulla negli altri casi.

5.3. Pista



La pista continua ovviamente a mantenere una topologia a "margherita". Le transizioni sono state adattate al livello di astrazione che è stato assunto nella modellazione dei sistemi Treno e Auto.

5.4. Automobile



STATI

Gli stati del DST denotato la posizione dell'automobile sul sistema.
Nella posizione *A* l'auto si trova completamente prima del blocco *b1* senza intralciare quindi l'eventuale passaggio del treno.
La posizione *B* invece invade completamente i binari e vuole quindi modellare lo stato dell'auto nel momento in cui precipita dal cavalcavia fin sulla pista del treno.
La posizione *C* viene assunta dall'auto - in tempo assumibile pari a zero - nel momento in cui venga posta tra la posizione *PD* e il cavalcavia (vedi mappa).
Ulteriori due stati per modellare eventuali scontri con il treno e per permettere all'utente di posizionare l'auto direttamente sul rimorchio.

TRANSIZIONI

Unico spostamento possibile che l'auto può compiere è dalla posizione *A* alla posizione *B* . Negli altri casi il sistema va in crash, a parte lo stato *sul_rimorchio* che non presenta alcun arco in entrata o in uscita.

5.5. Codifica NuSMV

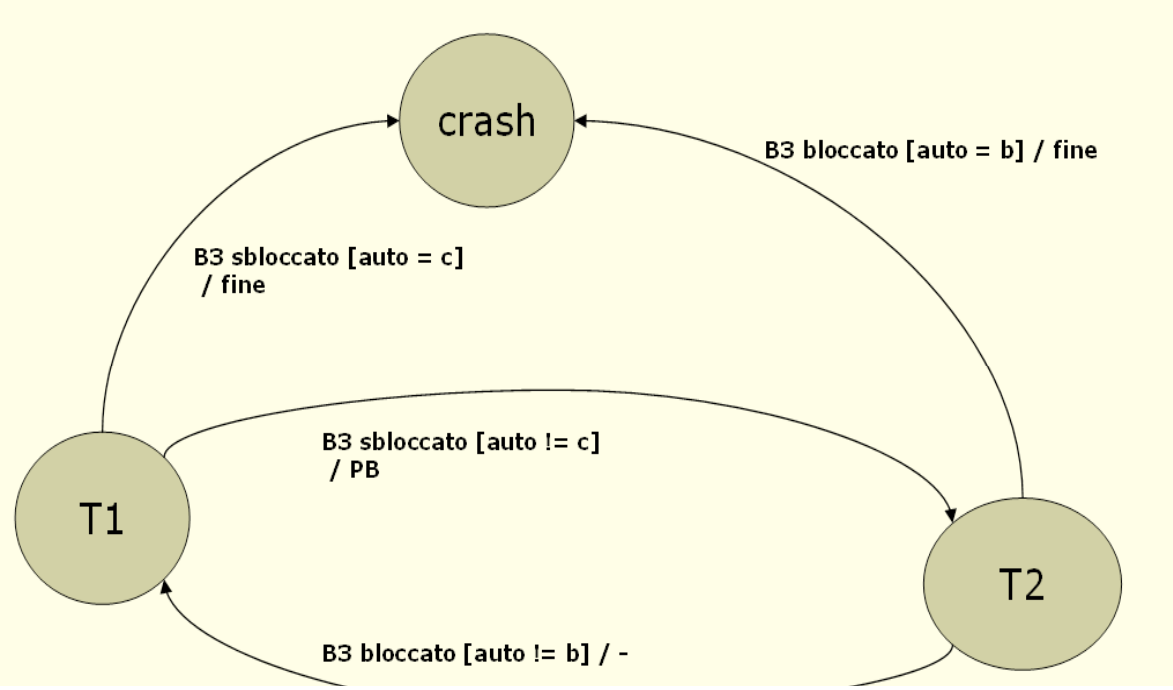
Riportiamo di seguito alcune delle proprietà verificate attraverso l'uso di NuSMV sia come model checker che "model finder"

- Il sistema va sempre in crash
  `G F (t.stato = crash & t.a.stato = crash)`

- il sistema non va mai in crash
  `G ! (t.stato = crash & t.a.stato = crash)`

6. Modellazione in D.S.T. : automobile e treno invertiti

6.2. Treno



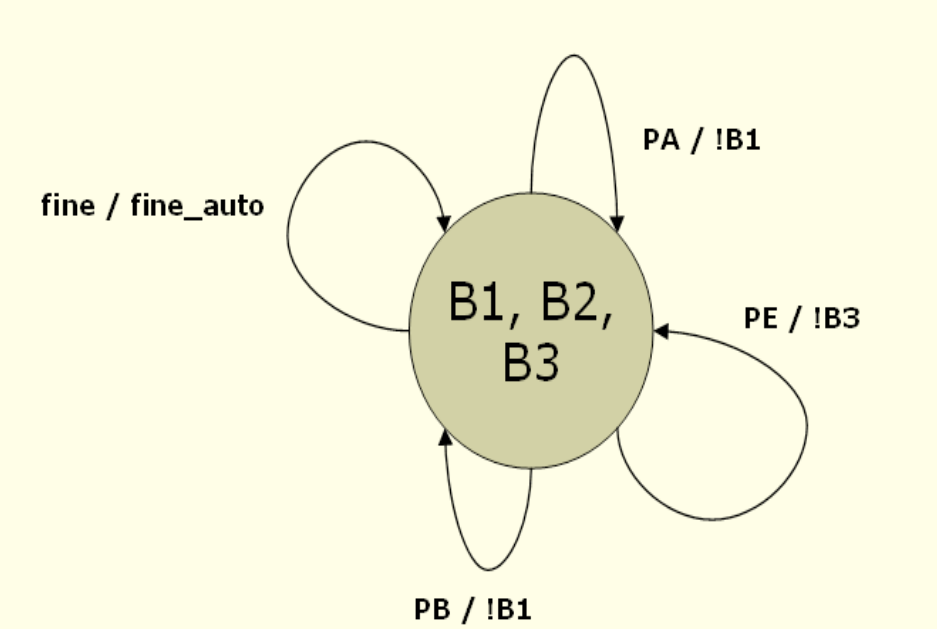
STATI

Gli stati principali sono due. *T1* indica che il treno è fermo alla sinistra del blocco *b3*, mentre *T2* indica che il treno è in movimento e sta quindi percorrendo la pista in senso inverso.
Un terzo stato è stato aggiunto anche in questo caso per modellare eventuali scontri tra treno e auto. Gli stati sono quindi gli stessi di quelli utilizzati nella modellazione del Sistema in cui si prevedeva soltanto il treno invertito e non l'automobile.

TRANSIZIONI

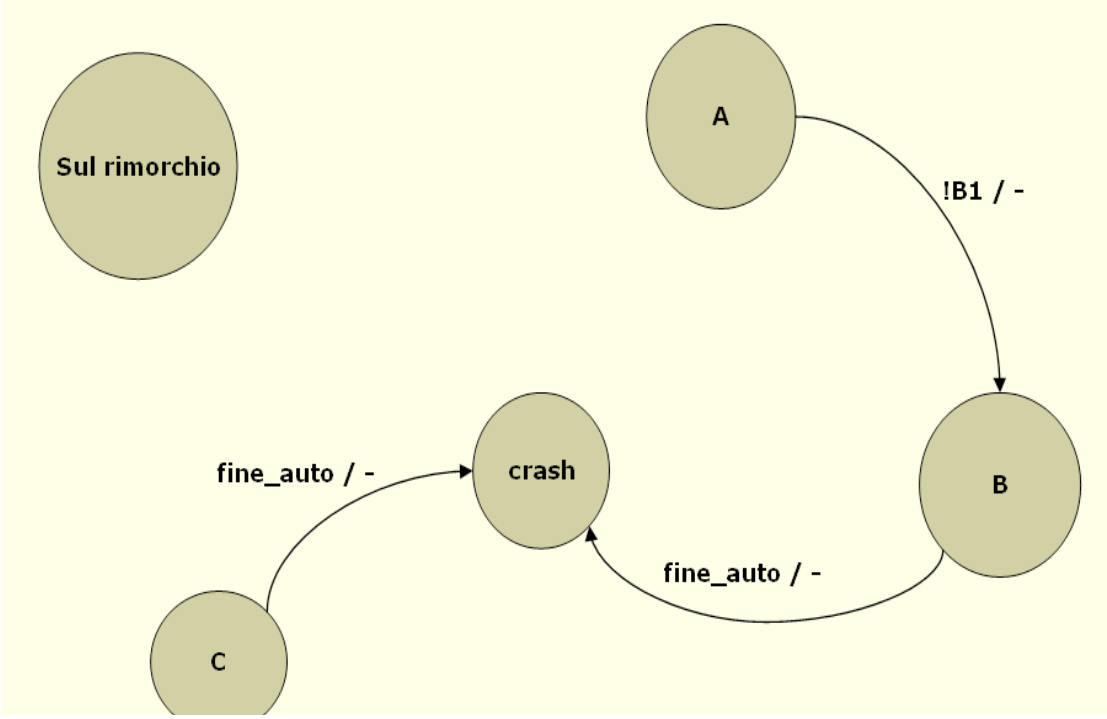
Le transizioni catturano tutti gli scenari di possibili crash o di funzionamento corretto del treno, il quale potenzialmente può alternasi tra gli stati *T1* e *T2*

6.3. Pista



La pista continua ovviamente a mantenere una topologia a "margherita" e in particolare è invariata rispetto al caso precedente.

6.4. Automobile



Il diagramma degli stati e transizioni dell'automobile risulta ovviamente identico al precedente caso analizzato.

6.5. Codifica NuSMV

Riportiamo di seguito alcune delle proprietà verificate attraverso l'uso di NuSMV sia come model checker che "model finder"

- Il sistema va sempre in crash
  G F (t.stato = crash & t.a.stato = crash)

- il sistema non va mai in crash
  G ! (t.stato = crash & t.a.stato = crash)