Capitolo 6

Logica proposizionale temporale

Molti aspetti importanti del software sono intrinsecamente *dinamici*, ovvero hanno a che fare con l'evoluzione dello stato del sistema. L'esempio principale è ovviamente quello dei programmi scritti in linguaggi procedurali, come C e Java, ma anche documenti prodotti in fasi più alte del ciclo di vita del software (come protocolli, diagrammi UML degli stati, transizioni e attività, ...) hanno queste caratteristiche.

La logica del prim'ordine non si presta molto bene a caratterizzare aspetti dinamici. Come vedremo infatti nel capitolo 7, uno dei suoi usi principali è nella semantica dei diagrammi UML delle classi, che caratterizzano bene tutti gli aspetti *statici*, ovvero indipendenti dall'evoluzione.

Per questi motivi in questo capitolo introduciamo un'estensione della logica proposizionale, la cosiddetta *logica proposizionale temporale lineare (LTL)*. Ci limitiamo alla versione proposizionale essenzialmente per motivi pratici: poiché le deduzioni in LTL hanno elevata complessità computazionale, risultano già sufficientemente difficili da mettere in difficoltà i sistemi più efficienti.

6.1 Sintassi

Nell'ambito della logica vengono studiate molte modellazioni del tempo, che differiscono fra loro per vari aspetti:

istanti o intervalli: le formule logiche possono essere valutate rispetto a singoli istanti, oppure rispetto a intervalli di tempo;

tempo discreto o continuo: nelle logiche a tempo discreto il tempo viene rappresentato tramite i numeri interi; nelle logiche a tempo continuo vengono usati i numeri reali;

passato e futuro: possiamo essere interessati a modellare solamente il futuro, oppure anche il passato;

linearità del tempo: il modello *lineare* del tempo prevede che ogni istante abbia solamente un successore; nel modello *ramificato* sono permessi diversi successori.

LTL è una logica lineare che modella il tempo con istanti discreti, in cui è di interesse solamente il futuro. LTL non è l'unica logica temporale di interesse per l'informatica: una logica molto usata è CTL (dall'inglese *computation tree logic*), che differisce da LTL per avere un modello del tempo ramificato. Inoltre, esistono varianti di LTL che modellano anche il passato. In questo testo limiteremo la nostra attenzione a LTL nella sua versione base, in quanto è sufficientemente espressiva da permetterci di modellare molti aspetti del software. A CTL verranno dedicati solamente alcuni commenti alla fine del paragrafo 6.3.

Come nel caso della logica proposizionale, faremo sempre riferimento ad un un insieme **LP** di lettere proposizionali. In LTL l'insieme dei simboli leciti è più ampio che nella logica proposizionale, poiché abbiamo bisogno di esprimere la verità delle proprietà in vari istanti di tempo.

Definizione 6.1.1 (Alfabeto di LTL) *L'alfabeto di LTL è dato da:*

- *l'alfabeto della logica proposizionale (cfr. definizione 3.2.1);*
- *i* connettivi temporali *X, F, G, U, W, R.*

Come le formule proposizionali (cfr. definizione 3.2.2), anche le formule di LTL vengono definite in maniera induttiva, tenendo conto che alcuni connettivi temporali sono unari, ed altri binari.

Definizione 6.1.2 (Formule di LTL) *Il linguaggio (insieme delle formule) di LTL è definito induttivamente come segue:*

- *se ϕ è una lettera proposizionale, cioè $\phi \in \mathbf{LP}$, allora ϕ è una formula di LTL;*
- *se ϕ e ψ sono formule di LTL, lo sono anche:*
  - (ϕ)
  - $\neg\phi$
  - $\phi \vee \psi$
  - $\phi \wedge \psi$
  - $\phi \rightarrow \psi$
  - $\phi \equiv \psi$
  - $X\ \phi$
  - $G\ \phi$
  - $F\ \phi$
  - $\phi\ U\ \psi$
  - $\phi\ W\ \psi$
  - $\phi\ R\ \psi$

Si noti come la definizione 6.1.2 implichi che le formule della logica proposizionale siano formule di LTL. Tuttavia, in questo caso l'insieme dei connettivi include alcuni connettivi temporali, che permettono di descrivere aspetti della realtà in vari istanti di tempo. Se una formula è priva di connettivi temporali (ovvero è una formula della logica proposizionale), allora si riferisce esclusivamente all'istante attuale. La formula $X\ \phi$ afferma la verità di ϕ nel prossimo istante temporale (il simbolo X viene usato con riferimento al temine inglese per *successivo*, ovvero *neXt*). La formula $G\ \phi$ afferma la verità di ϕ in tutti gli istanti futuri, compreso l'attuale (*Globally*). $F\ \phi$ afferma la verità di ϕ in almeno un istante futuro, compreso l'attuale (*Future*). $\phi\ U\ \psi$ afferma che prima o poi ψ diventerà vero, e che, fino a quel momento (*Until*), ϕ è vero. $\phi\ W\ \psi$ differisce dal precedente perché ψ potrebbe non diventare mai vero (*Weak until*). $\phi\ R\ \psi$ afferma che fino a quando ϕ non diventerà vero (*Release*), ψ sarà vero. Queste nozioni intuitive saranno formalizzate nel paragrafo 6.2.

Esempio 6.1.3 Le seguenti sequenze di simboli dell'alfabeto della logica proposizionale sono formule di LTL.

(1) $(p \wedge \neg p) \rightarrow q$

(2) $(p\ W\ q)\ U\ G\ s$

(3) $p\ W\ q\ U\ \ G\ s$

(4) $p\ W\ (q\ U\ G\ s)$

(5) $s\ W\ (Xs)$

(6) $(p\ W\ Xp)\ W\ q$

(7) $p\ W\ q \equiv s$

(8) $(p\ W\ q) \equiv s$

(9) $p\ W\ (q \equiv s)$

(10) $p\ U\ q\ U\ s$

(11) $(p\ U\ q)\ U\ s$

(12) $p\ U\ (q\ U\ s)$

Al contrario, le sequenze

(13) $\ R\ p$

(14) $p\ F\ q\ X\ (Xs)$

(15) $\forall Y\ F\ p(Y)$

(16) $p\ U\ \ U\ q \equiv s$

(17) $p\ X\ q\ U\ s$

(18) $F\ \forall Y\ p(Y)$

non sono formule di LTL. ∘

6.2 Semantica

Nella logica proposizionale possiamo definire la semantica utilizzando la nozione di interpretazione (cfr. definizione 3.3.1). In LTL abbiamo bisogno di esprimere l'evoluzione del tempo, e in particolare il fatto che ci sia un'interpretazione per ogni istante di tempo. La nozione di *struttura temporale* cattura questa intuizione, in quanto prevede la possibilità di associare un'interpretazione ad ogni istante.

Similmente alla logica proposizionale la semantica di LTL viene fornita mediante tre passi fondamentali.

1. Si definisce la nozione di *struttura temporale*, cioè di quel meccanismo che consente di assegnare, per ogni istante di tempo, un valore di verità ad ogni lettera proposizionale.
2. Si definisce la legge per stabilire il significato di ogni formula rispetto ad una struttura temporale.
3. Si stabilisce il significato di ogni formula senza riferimento a particolari strutture temporali, e si illustrano diverse nozioni interessanti legate a tale significato.

La prossima definizione costituisce il primo passo.

Definizione 6.2.1 (Struttura temporale) *Una* struttura temporale M *è una tripla* $\langle S, R, L\rangle$ *dove:*

- S *è un insieme finito di* stati*;*
- $R \subseteq S \times S$ *è una* relazione di transizione *con la seguente proprietà:*

$$\forall s \in S \; \exists s' \in S \; (s, s') \in R; \tag{6.1}$$

- $L : S \longrightarrow (\mathbf{LP} \longrightarrow \{\mathsf{T}, \mathsf{F}\})$ *è una* funzione di etichettatura*, che associa ad ogni stato un'interpretazione (cfr. definizione 3.3.1) delle lettere dell'alfabeto* $\mathbf{LP}$*.*

Commentiamo la definizione appena data con alcune osservazioni.

- Uno stato rappresenta la realtà in un certo istante di tempo. In particolare, ogni stato s è caratterizzato dall'interpretazione $L(s)$ delle lettere di $\mathbf{LP}$, che serve per indicare quali siano le proprietà vere e quali quelle false in un dato istante. $L(s)$ è una funzione; di conseguenza, data una lettera proposizionale $p \in \mathbf{LP}$, $L(s)(p)$ può valere T o F.
- La relazione di transizione R rappresenta la successione degli stati nel tempo, e le possibili evoluzioni del sistema sono rappresentate da cammini nella relazione (cfr. definizione 6.2.3).
- La proprietà (6.1) impone che per ogni stato esista almeno uno stato successivo, e quindi garantisce che esistano cammini infiniti.
- Una struttura temporale viene efficacemente rappresentata tramite grafi etichettati sui nodi, come mostrato dall'esempio 6.2.2.
- Le strutture temporali vengono definite come grafi per poter valutare sia formule di LTL sia formule di CTL. Come vedremo nel seguito, in LTL ogni istante di tempo (rappresentato, ad es., dallo stato s) ha un solo istante successivo, scelto fra gli stati della struttura temporale ad esso adiacenti (ovvero quelli che hanno un arco in entrata proveniente da s), mentre in CTL tutti gli stati nella struttura adiacenti ad s sono ammessi come rappresentanti dei possibili istanti successivi. In letteratura a volte la semantica di LTL viene data utilizzando strutture *lineari*, ovvero in cui ogni stato ha esattamente uno stato adiacente.
- Per indicare una struttura temporale si trovano in letteratura anche i termini *modello*, *sistema di transizioni* e *struttura di Kripke*. In particolare il primo giustifica il termine *model checking*, (cfr. definizioni 6.2.9 e 6.2.11), ampiamente usato per descrivere l'adeguatezza di una struttura temporale rispetto a una formula.

Esempio 6.2.2 Sia $\mathbf{LP} = \{p, q, r\}$. Un esempio di struttura temporale è mostrato dal grafo in figura 6.1, dove:

- l'insieme degli stati $S = \{s_1, s_2, s_3, s_4\}$ corrisponde ai nodi del grafo;
- la relazione di transizione R è rappresentata dagli archi del grafo; si noti che la proprietà (6.1) è soddisfatta;
- ogni stato $s \in S$ è etichettato con l'interpretazione $L(s)$; per convenzione, vengono evidenziate solamente le lettere di $\mathbf{LP}$ che hanno il valore T. Ad esempio, $L(s_1)(p) = L(s_1)(q) = \mathsf{T}$ e $L(s_1)(r) = \mathsf{F}$.

∘

Per descrivere la semantica di LTL abbiamo bisogno di varie definizioni, la prima delle quali è quella di *cammino in una struttura temporale*.

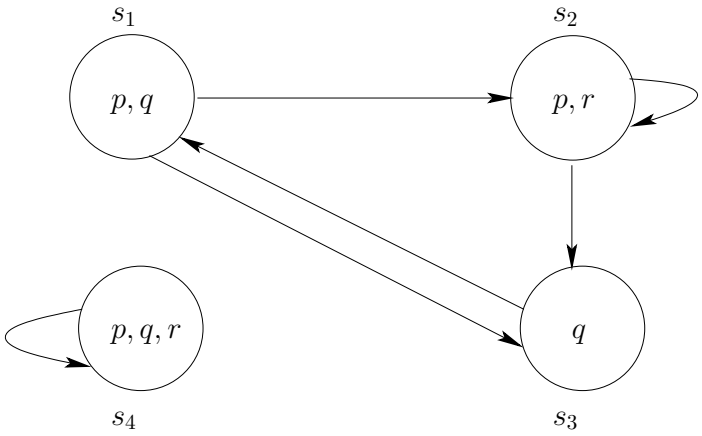

Figura 6.1 Una struttura temporale

Definizione 6.2.3 (Cammino in una struttura temporale) *Un* cammino π in una struttura temporale $M = \langle S, R, L\rangle$ *è una sequenza infinita di stati* $s_1, s_2, s_3, \ldots$ *di* S *tale che, per ogni* i *(*$i \geq 1$*),* $(s_i, s_{i+1}) \in R$*. Per ogni* $i \geq 1$*,* π^i *denota il suffisso di* π *che inizia da* s_i *e comprende quest'ultimo.*

A volte è utile visualizzare tutti i possibili cammini a partire da un certo stato di una struttura temporale. Allo scopo possiamo disegnare in parte un albero infinito la cui radice è lo stato che ci interessa, e i cui cammini a partire dalla radice corrispondono ai cammini di cui alla definizione 6.2.3. La figura 6.2 mostra una parte di albero infinito con riferimento allo stato s_1 della struttura temporale di figura 6.1.

La definizione seguente chiarisce il significato di una formula rispetto a un cammino ed è, per LTL, l'analoga della 3.3.3 per la logica proposizionale.

Definizione 6.2.4 (Valutazione di una formula rispetto a un cammino) *Sia dato l'insieme* **LP** *delle lettere proposizionali e sia* **FORM** *l'insieme delle formule di LTL risultanti. Sia* $\pi = s_1, s_2, \ldots$ *un cammino in una struttura temporale* $M = \langle S, R, L\rangle$*. Definiamo, in dipendenza da* M *e* π*, la funzione*

$$\mathsf{eval}^{M,\pi} : \mathbf{FORM} \longrightarrow \{\mathsf{T}, \mathsf{F}\}.$$

La funzione è definita in modo induttivo, seguendo la struttura induttiva delle formule:

- *se* $p \in \mathbf{LP}$*:*

$$\mathsf{eval}^{M,\pi}(p) \quad = \quad L(s_1)(p)$$

- *se* ϕ *e* $\psi \in \mathbf{FORM}$ *allora:*

(1)	$\mathsf{eval}^{M,\pi}(\ (\phi)\ )$	$=$	$\mathsf{eval}^{M,\pi}(\phi)$
(2)	$\mathsf{eval}^{M,\pi}(\neg\phi)$	$=$	T *se* $\mathsf{eval}^{M,\pi}(\phi) = \mathsf{F}$
	$\mathsf{eval}^{M,\pi}(\neg\phi)$	$=$	F *altrimenti*
(3)	$\mathsf{eval}^{M,\pi}(\phi \vee \psi)$	$=$	T *se* $\mathsf{eval}^{M,\pi}(\phi) = \mathsf{T}$ *oppure* $\mathsf{eval}^{M,\pi}(\psi) = \mathsf{T}$
	$\mathsf{eval}^{M,\pi}(\phi \vee \psi)$	$=$	F *altrimenti*
(4)	$\mathsf{eval}^{M,\pi}(\phi \wedge \psi)$	$=$	T *se* $\mathsf{eval}^{M,\pi}(\phi) = \mathsf{eval}^{M,\pi}(\psi) = \mathsf{T}$
	$\mathsf{eval}^{M,\pi}(\phi \wedge \psi)$	$=$	F *altrimenti*
(5)	$\mathsf{eval}^{M,\pi}(\phi \rightarrow \psi)$	$=$	$\mathsf{eval}^{M,\pi}(\neg\phi \vee \psi)$

$$
\begin{array}{lllll}
(6) & \mathsf{eval}^{M,\pi}(\phi \equiv \psi) & = & \mathsf{eval}^{M,\pi}(\ (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)\ ) & \\
(7) & \mathsf{eval}^{M,\pi}(X\ \phi) & = & \mathsf{eval}^{\pi^2}(\phi) & \\
(8) & \mathsf{eval}^{M,\pi}(G\ \phi) & = & \mathsf{T} \textit{ se per ogni } i\ (i \geq 1) \textit{ vale } \mathsf{eval}^{\pi^i}(\phi) = \mathsf{T} & \\
 & \mathsf{eval}^{M,\pi}(G\ \phi) & = & \mathsf{F} \textit{ altrimenti} & \\
(9) & \mathsf{eval}^{M,\pi}(F\ \phi) & = & \mathsf{T} \textit{ se esiste } i\ (i \geq 1) \textit{ per cui vale } \mathsf{eval}^{\pi^i}(\phi) = \mathsf{T} & \\
 & \mathsf{eval}^{M,\pi}(F\ \phi) & = & \mathsf{F} \textit{ altrimenti} & \\
(10) & \mathsf{eval}^{M,\pi}(\phi\ U\ \psi) & = & \mathsf{T} \textit{ se esiste } i\ (i \geq 1) \textit{ per cui vale } \mathsf{eval}^{\pi^i}(\psi) = \mathsf{T} \textit{ e tale che} & \\
 & & & \textit{per ogni } j\ (1 \leq j \leq i-1) \textit{ vale } \mathsf{eval}^{\pi^j}(\phi) = \mathsf{T} & \\
 & \mathsf{eval}^{M,\pi}(\phi\ U\ \psi) & = & \mathsf{F} \textit{ altrimenti} & \\
(11) & \mathsf{eval}^{M,\pi}(\phi\ W\ \psi) & = & \mathsf{T} \textit{ se esiste } i\ (i \geq 1) \textit{ per cui vale } \mathsf{eval}^{\pi^i}(\psi) = \mathsf{T} \textit{ e tale che} & \\
 & & & \textit{per ogni } j\ (1 \leq j \leq i-1) \textit{ vale } \mathsf{eval}^{\pi^j}(\phi) = \mathsf{T}, & \\
 & & & \textit{oppure se per ogni } k\ (k \geq 1) \textit{ vale } \mathsf{eval}^{\pi^k}(\phi) = \mathsf{T} & \\
 & \mathsf{eval}^{M,\pi}(\phi\ W\ \psi) & = & \mathsf{F} \textit{ altrimenti} & \\
(12) & \mathsf{eval}^{M,\pi}(\phi\ R\ \psi) & = & \mathsf{T} \textit{ se esiste } i\ (i \geq 1) \textit{ per cui vale } \mathsf{eval}^{\pi^i}(\phi) = \mathsf{T} \textit{ e tale che} & \\
 & & & \textit{per ogni } j\ (1 \leq j \leq i) \textit{ vale } \mathsf{eval}^{\pi^j}(\psi) = \mathsf{T}, & \\
 & & & \textit{oppure se per ogni } k\ (k \geq 1) \textit{ vale } \mathsf{eval}^{\pi^k}(\psi) = \mathsf{T} & \\
 & \mathsf{eval}^{M,\pi}(\phi\ R\ \psi) & = & \mathsf{F} \textit{ altrimenti.} &
\end{array}
$$

Data una formula di LTL ϕ e un cammino π:

- *se* $\mathsf{eval}^{M,\pi}(\phi) = \mathsf{T}$*, scriveremo anche* $M, \pi \models \phi$*,*
- *se* $\mathsf{eval}^{M,\pi}(\phi) = \mathsf{F}$*, scriveremo anche* $M, \pi \not\models \phi$*.*

Facciamo alcune osservazioni riguardo la definizione data.

- Le regole da (1) a (6) sono identiche a quelle della logica proposizionale (cfr. definizione 3.3.3).
- I connettivi temporali unari X, G e F hanno il significato intuitivo di *verità all'istante successivo*, *verità in ogni istante presente o futuro* e *verità in qualche istante presente o futuro*, rispettivamente.
- Per quanto riguarda i connettivi binari:
  - $\phi\ U\ \psi$ afferma che prima o poi ψ diventerà vero, e che, fino al momento precedente, ϕ è vero;
  - $\phi\ W\ \psi$ è simile, ma ψ potrebbe non diventare mai vero,
  - $\phi\ R\ \psi$ afferma che fino al momento (compreso) in cui ϕ non diventerà vero ψ sarà vero; è anche ammesso che ϕ non diventi mai vero.

  La figura 6.3 rappresenta intuitivamente il significato di tutti i connettivi temporali.
- Nel paragrafo 3.3 abbiamo visto che esistono insiemi minimali di connettivi logici (ad esempio $\{\neg, \vee\}$ e $\{\wedge, \neg\}$) che possono servire per definire tutti gli altri. La situazione è analoga in LTL, ed esempi di tali insiemi minimali di connettivi temporali sono $\{U, X\}$, $\{R, X\}$ e $\{W, X\}$.
- La minimalità di un insieme $\mathcal{I}$ di connettivi temporali può essere provata dimostrando che un connettivo non in $\mathcal{I}$ può essere espresso utilizzando solo quelli di $\mathcal{I}$, possibilmente attraverso varie riscritture. In questo contesto utilizziamo una nozione intuitiva di equivalenza, che verrà introdotta formalmente in seguito (cfr. definizione 6.2.8).

  Ad esempio, la minimalità di $\{U, X\}$ può essere mostrata in questo modo (ϕ è una qualsiasi formula di LTL):

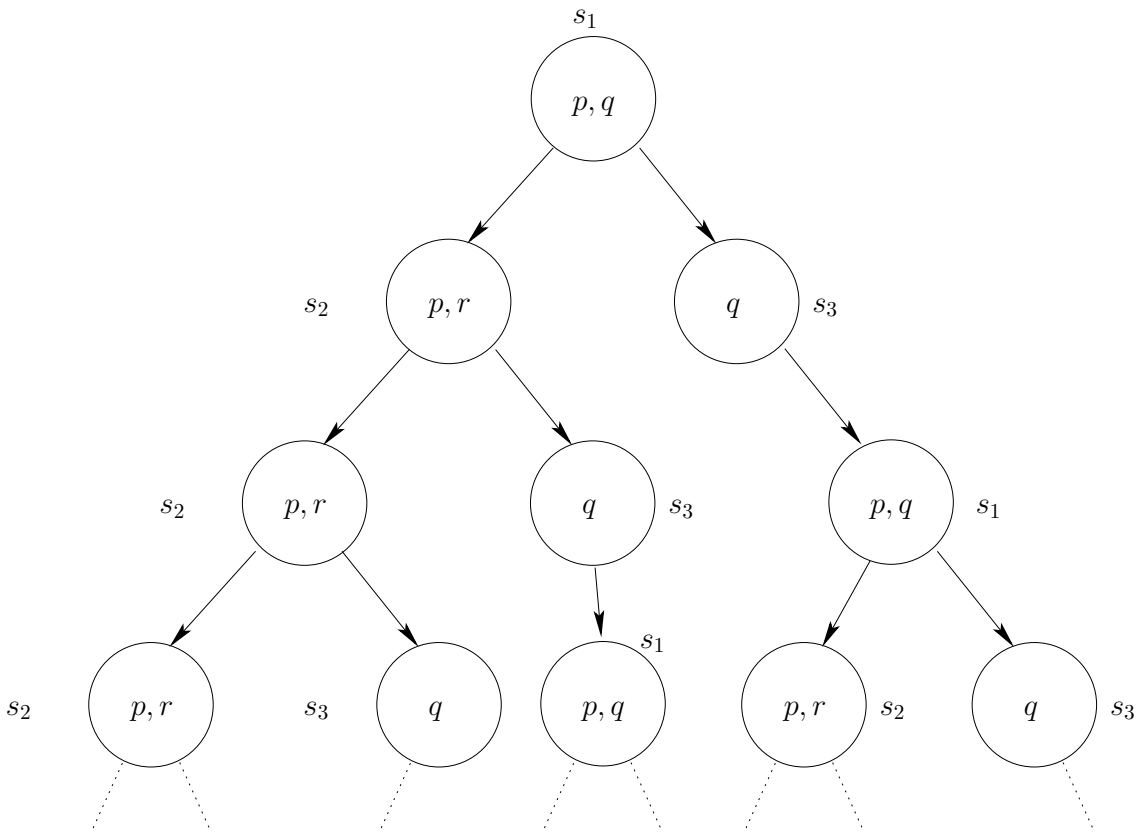

Figura 6.2 La struttura temporale di figura 6.1 espansa come albero infinito a partire dallo stato s_1

- $G\ \phi$ può essere riscritta come $\neg F\ \neg\phi$ (cfr. punti (8) e (9)).
- $F\ \phi$ può essere riscritta come $\mathsf{true}\ U\ \phi$ (cfr. punti (9) e (10)), dove true rappresenta una qualsiasi tautologia proposizionale, ad esempio $p \vee \neg p$.
- $\phi\ W\ \psi$ può essere riscritta come $(\phi\ U\ \psi) \vee G\ \phi$ (cfr. punti (10) e (11)).
- $\phi\ R\ \psi$ può essere riscritta come $\neg(\neg\phi\ U\ \neg\psi)$ (cfr. punti (10) e (12)).

- Come per le formule proposizionali, esiste ambiguità nella valutazione delle formule di LTL: facendo riferimento all'esempio 6.1.3, non sappiamo se la formula (3) debba essere valutata come la (2) oppure come la (4). La stessa cosa può essere detta delle formule (7), (8), (9) e (10), (11), (12).

  Anche in LTL facciamo ricorso a convenzioni sulla precedenza dei connettivi. Seguendo la scelta fatta per la logica proposizionale, daremo maggiore priorità ai connettivi unari $(\neg, X, F, G)$ rispetto a quelli binari $(U, R, W, \vee, \wedge, \rightarrow, \equiv)$.

  Riassumendo, adottiamo la seguente gerarchia tra i connettivi:

  1. $\neg, X, F, G$,
  2. U, R, W,
  3. $\wedge$,
  4. $\vee$,
  5. $\rightarrow$, $\equiv$

  e, nei casi in cui siano coinvolti connettivi dello stesso livello, assumiamo che i connettivi siano associativi a sinistra. Così la formula $p\ W\ q\ U\ G\ s$ viene interpretata come $(p\ W\ q)\ U\ G\ s$, la formula $p\ W\ q \equiv s$ come $(p\ W\ q) \equiv s$, e la formula $p\ U\ q\ U\ s$ come $(p\ U\ q)\ U\ s$.

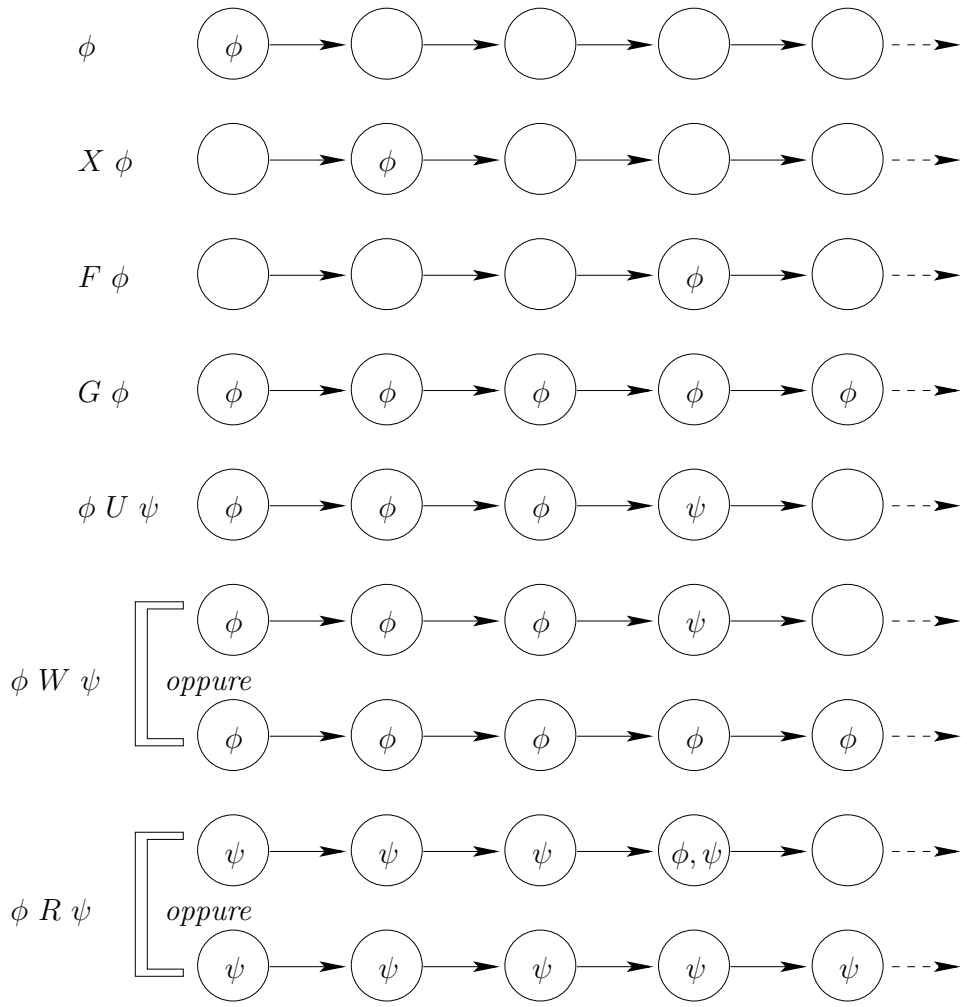

Figura 6.3 Semantica dei connettivi temporali

Esempio 6.2.5 Con riferimento alla struttura temporale dell'esempio 6.2.2, è possibile prendere in considerazione vari cammini. Nella seguente tabella, ogni colonna corrisponde a una formula ϕ di LTL, ogni riga a un cammino π nella struttura temporale, e gli elementi hanno il valore di $\mathsf{eval}^{M,\pi}(\phi)$.

	p	$X\ r$	Gp	$F\ \neg q$	$q\ U\ r$	$p\ W (q \wedge r)$	$p\ R\ r$
$\pi_1 = s_1, s_2, s_2, \ldots$	T	T	T	T	T	T	T
$\pi_2 = s_3, s_1, s_3, s_1, \ldots$	F	F	F	F	F	F	F

○

Esercizio 6.1 [Soluzione a pagina 85] Con riferimento all'esempio 6.2.5, estendere la tabella considerando anche i cammini $\pi_3 = s_4, s_4, \ldots$ e $\pi_4 = s_2, s_3, s_1, s_2, s_3, s_1, \ldots$. ○

Similmente al caso della logica proposizionale, data una formula di LTL ci si può chiedere:

- se non esistano strutture temporali e cammini in cui essa viene valutata a T;
- se esista almeno una struttura temporale e un cammino in cui essa viene valutata a T;
- se venga valutata a T in ogni struttura temporale e in ogni cammino.

La definizione che segue è, per LTL, l'analoga della 3.3.8 per la logica proposizionale.

Definizione 6.2.6 (Formule soddisfacibili, insoddisfacibili e valide) *Sia ϕ una formula di LTL. ϕ è:*

- soddisfacibile *se esiste una struttura temporale M e un cammino π in essa tale che $M, \pi \models \phi$,*
- insoddisfacibile *altrimenti,*
- valida *se in ogni struttura temporale M e in ogni cammino π in essa vale $M, \pi \models \phi$ (in questo caso si dice anche che ϕ è una* tautologia*).*

Esempio 6.2.7 Tutte le formule dell'esempio 6.2.5, ovvero p, $X\ r$, Gp, $F\ \neg q$, $q\ U\ r$, $p\ W (q \wedge r)$ e $p\ R\ r$ sono soddisfacibili, in quanto sono valutate a T dal cammino π_1. Tali formule non sono valide, in quanto sono valutate a F dal cammino π_2.

Tutte le tautologie della logica proposizionale sono anche tautologie di LTL. Un'altra tautologia di LTL è la formula

$$G\ \phi \equiv \neg F\ \neg \phi \tag{6.2}$$

(cfr. punti (8) e (9) della definizione 6.2.4). La negazione della formula (6.2) è invece insoddisfacibile (cfr. esercizio 6.3). ◦

Esercizio 6.2 Siano ϕ e ψ formule qualsiasi di LTL. Si dimostri che le seguenti formule sono tautologie:

(1) $(G\ \phi) \rightarrow \phi$

(2) $\psi \rightarrow (\phi\ U\ \psi)$

(3) $\phi \rightarrow F\ \phi$

(4) $(\neg\phi\ W\ \psi) \rightarrow G(F\ \phi \rightarrow F\ G\ \psi)$

◦

Esercizio 6.3 Sia ϕ una formula qualsiasi di LTL. Dimostrare che ϕ è valida se e solo se $\neg\phi$ è insoddisfacibile. ◦

Una definizione strettamente collegata alla precedente è quella di *equivalenza logica*, che si basa sull'uguaglianza delle strutture e dei cammini che rendono vera una formula.

Definizione 6.2.8 (Equivalenza logica) *Siano ϕ e ψ due formule di LTL. Si dice che ϕ è* logicamente equivalente *a ψ se:*

$$\{M, \pi \mid M, \pi \models \phi\} \ = \ \{M, \pi \mid M, \pi \models \psi\}.$$

Esercizio 6.4 Siano ϕ e ψ due formule di LTL. Dimostrare che ϕ è logicamente equivalente a ψ se e solo se $\phi \equiv \psi$ è una tautologia. ◦

Il prossimo esercizio illustra alcune utili equivalenze fra formule di LTL.

Esercizio 6.5 Siano ϕ e ψ formule qualsiasi di LTL. Si dimostrino le seguenti equivalenze:

(1)	$\neg G\ \phi$	eq.	$F\ \neg\phi$
(2)	$\neg F\ \phi$	eq.	$G\ \neg\phi$
(3)	$\neg X\ \phi$	eq.	$X\ \neg\phi$
(4)	$\neg(\phi\ R\ \psi)$	eq.	$(\neg\phi\ U\ \neg\psi)$
(5)	$\neg(\phi\ U\ \psi)$	eq.	$(\neg\phi\ R\ \neg\psi)$
(6)	$X(\phi\ \vee\ \psi)$	eq.	$X\ \phi\ \vee\ X\ \psi$
(7)	$X(\phi\ \wedge\ \psi)$	eq.	$X\ \phi\ \wedge\ X\ \psi$
(8)	$X(\phi\ \rightarrow\ \psi)$	eq.	$X\ \phi\ \rightarrow\ X\ \psi$
(9)	$X(\phi\ \equiv\ \psi)$	eq.	$X\ \phi\ \equiv\ X\ \psi$
(10)	$F(\phi\ \vee\ \psi)$	eq.	$F\ \phi\ \vee\ F\ \psi$

(11) $G(\phi \wedge \psi)$ eq. $G\,\phi \wedge G\,\psi$

(12) $X(\phi\ U\ \psi)$ eq. $X\phi\ U\ X\psi$

(13) $X(\phi\ W\ \psi)$ eq. $X\phi\ W\ X\psi$

(14) $GG\,\phi$ eq. $G\,\phi$

(15) $FF\,\phi$ eq. $F\,\phi$

(16) $(\phi\ U\ \psi)\ U\ \psi$ eq. $\phi\ U\ \psi$

(17) $(\phi\ W\ \psi)\ W\ \psi$ eq. $\phi\ W\ \psi$

(18) $FGF\,\phi$ eq. $GF\,\phi$

(19) $GFG\,\phi$ eq. $FG\,\phi$

(20) $F\,\phi$ eq. $\phi \vee XF\,\phi$

(21) $G\,\phi$ eq. $\phi \wedge XG\,\phi$

(22) $\phi\ U\ \psi$ eq. $\psi \vee (\phi \wedge X\ (\phi\ U\ \psi))$

(23) $XG\,\phi$ eq. $GX\,\phi$

(24) $XF\,\phi$ eq. $FX\,\phi$

(25) $F\,\phi$ eq. $\mathsf{true}\ U\ \phi$

(26) $G\,\phi$ eq. $\mathsf{false}\ R\ \phi$

(27) $\phi\ W\ \psi$ eq. $(\phi\ U\ \psi) \vee G\,\phi$

(28) $\phi\ U\ \psi$ eq. $(\phi\ W\ \psi) \wedge F\ \psi$

(29) $\phi\ W\ \psi$ eq. $\psi\ R(\phi \vee \psi)$

(30) $\phi\ R\ \psi$ eq. $\psi\ W(\phi \wedge \psi)$

true rappresenta una qualsiasi tautologia proposizionale e false una qualsiasi formula proposizionale insoddisfacibile (ad es. $p \vee \neg p$ e $p \wedge \neg p$, rispettivamente). ∘

Fra le equivalenze menzionate nell'esercizio precedente, sono di particolare interesse quelle che esprimono:

- *dualità* (1-5) fra connettivi temporali (ad es., G è duale di F e X è duale di se stesso);
- *distributività* (6-13) di connettivi temporali su altri connettivi (ad es., G si distribuisce su $\wedge$);
- *idempotenza* (14-17) di sequenze di connettivi (ad es., la sequenza GG è equivalente a G);
- *assorbimento* (18-19) di sequenze di connettivi (ad es., la sequenza FGF viene assorbita in GF);
- *espansione* (20-22) di connettivi (ad es., F si espande tenendo conto dell'istante successivo).

Prima di concludere il paragrafo sulla semantica di LTL dobbiamo fornire alcune ulteriori definizioni. La definizione 6.2.4 chiarisce come valutare una formula di LTL una volta specificata una struttura temporale e un cammino in essa. Di seguito mostriamo la definizione, più astratta, di una funzione che valuti una formula data una struttura temporale a prescindere dal cammino.

Definizione 6.2.9 (Valutazione di una formula rispetto a una struttura temporale) *Sia* **FORM** *come in definizione 6.2.4, e sia una struttura temporale $M = \langle S, R, L\rangle$. Definiamo, in dipendenza da M, la funzione*

$$\mathsf{eval}^M : \mathbf{FORM} \longrightarrow \{\mathsf{T}, \mathsf{F}\}$$

nel seguente modo:

$\mathsf{eval}^M(\phi) = \mathsf{T}$ *se per ogni cammino* π *di* M *vale* $\mathsf{eval}^{M,\pi}(\phi) = \mathsf{T}$
$\mathsf{eval}^M(\phi) = \mathsf{F}$ *altrimenti.*

Data una formula di LTL ϕ *e una struttura temporale* M*:*

- *se* $\mathsf{eval}^M(\phi) = \mathsf{T}$*, scriveremo anche* $M \models \phi$*,*
- *se* $\mathsf{eval}^M(\phi) = \mathsf{F}$*, scriveremo anche* $M \not\models \phi$*.*

Esempio 6.2.10 Vogliamo analizzare alcune formule di LTL e verificare come vengono valutate nella struttura temporale di figura 6.1. Chiaramente nessuna formula fra quelle dell'esempio 6.2.5 viene valutata a T, come testimoniato dal cammino π_2, in cui vengono valutate a F. Alcune formule che vengono valutate a T sono:

- $G(p \vee q)$
- $F\ p$
- $G\ r \rightarrow G\ p$

∘

Esercizio 6.6 [Soluzione a pagina 86] Con riferimento alla struttura temporale dell'esempio 6.2.2, dire quali delle seguenti formule vengono valutate a T e quali a F:

- $F\ X\ p$
- $F\ q$
- $G(p \wedge \neg q \rightarrow r)$
- $G\ p \rightarrow G\ r$
- $G(q\ U\ p)$
- $G(p\ U\ q)$
- $G\ p \rightarrow p\ U\ r$

Per quelle valutate a F esibire un cammino che lo testimoni. ∘

Molto spesso di un sistema dinamico (ad es., un diagramma UML degli stati e delle transizioni, cfr. capitolo 8) conosciamo lo *stato iniziale*. È utile quindi fornire la definizione 6.2.9 specializzata rispetto ai cammini che originano da un certo stato.

Definizione 6.2.11 (Valutazione di una formula rispetto a una struttura temporale e a uno stato iniziale) *Siano* **FORM** *e* $M = \langle S, R, L\rangle$ *come in definizione 6.2.9 e sia* $s \in S$*. Definiamo, in dipendenza da* M *e* s*, la funzione*

$$\mathsf{eval}^{M,s} : \mathbf{FORM} \longrightarrow \{\mathsf{T}, \mathsf{F}\}$$

nel seguente modo:

$\mathsf{eval}^{M,s}(\phi) = \mathsf{T}$ *se per ogni cammino* $\pi = s, \ldots$ *di* M *vale* $\mathsf{eval}^{M,\pi}(\phi) = \mathsf{T}$
$\mathsf{eval}^{M,s}(\phi) = \mathsf{F}$ *altrimenti.*

Data una formula di LTL ϕ*, una struttura temporale* $M = \langle S, R, L\rangle$ *e uno stato* $s \in S$*:*

- *se* $\mathsf{eval}^{M,s}(\phi) = \mathsf{T}$*, scriveremo anche* $M, s \models \phi$*,*
- *se* $\mathsf{eval}^{M,s}(\phi) = \mathsf{F}$*, scriveremo anche* $M, s \not\models \phi$*.*

La figura 6.2 è utile per evidenziare tutti i cammini con stato iniziale s_1 della struttura temporale di figura 6.1.

Come vedremo già nel prossimo paragrafo e più in dettaglio nei capitoli successivi, le formule di LTL vengono frequentemente usate per rappresentare specifiche di sistemi dinamici, come diagrammi UML degli stati e programmi in linguaggi imperativi. Le strutture temporali vengono spesso usate per descrivere implementazioni, e le definizioni appena date sono utili nei casi in cui si desideri verificare l'adeguatezza dell'implementazione (comprendente un eventuale stato iniziale) rispetto alla specifica, effettuando l'operazione che spesso viene definita come *model checking*.

Esempio 6.2.12 Detta M la struttura temporale di figura 6.1, abbiamo:

- $M, s_1 \not\models G\ p$
- $M, s_4 \models G\ p$

∘

Esercizio 6.7 [Soluzione a pagina 86] Con riferimento all'esercizio 6.6, per le formule che vengono valutate a F chiarire se esista uno stato iniziale rispetto alle quali vengono valutate a T. ∘

6.3 Rappresentazione in LTL di frasi in italiano

In questo paragrafo mostriamo come si possano rappresentare frasi in italiano attraverso formule di LTL. Iniziamo da un esempio che mostra tutti i connettivi temporali.

Esempio 6.3.1 Le frasi che ci interessano sono le seguenti:

1. sto mangiando,
2. mangio sempre,
3. prima o poi mangerò,
4. mangerò fino ad essere sazio,
5. mangerò fino ad essere sazio, ma potrei non saziarmi mai,
6. quando smetterò di mangiare sarò già sazio.

Si supponga di associare alle lettere proposizionali m ed s rispettivamente il valore di verità che attribuiamo ai concetti "mangio" e "sono sazio". Le frasi di cui sopra possono essere rappresentate mediante le seguenti formule:

1. m,
2. $G\ m$,
3. $F\ m$ (o $\neg m \wedge F\ m$, interpretando il tempo futuro dell'italiano come non comprendente il presente),
4. $m\ U\ s$
5. $m\ W\ s$,
6. $s\ R\ m\ \wedge\ F\ s$.

∘

Vista l'importanza di LTL per la specifica di sistemi dinamici, il prossimo esempio mostra come alcuni requisiti di natura progettuale possano essere agevolmente rappresentati mediante formule temporali.

Esempio 6.3.2 Un incrocio automobilistico fra tre strade a senso unico di marcia è regolato da due semafori: S_1 per le auto provenienti dalla strada 1 e S_2 per le auto provenienti dalla strada 2. Tutte le auto si dirigono verso la strada 3 (cfr. figura 6.4).

Ogni semaforo ha due luci (rossa e verde). I requisiti da rispettare nel progetto del sistema di accensione delle luci dei semafori sono i seguenti:

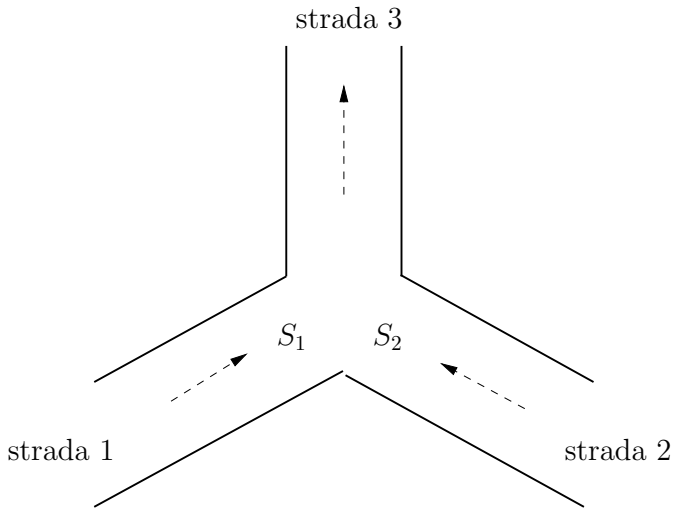

Figura 6.4 Un incrocio automobilistico

a) S_1 deve essere sempre rosso o verde, ma mai contemporaneamente;

b) idem, per S_2;

c) in ogni istante, se S_1 è verde, allora S_2 è rosso;

d) idem, scambiando S_1 con S_2;

e) S_1 diventa verde un numero infinito di volte;

f) idem, per S_2.

Innanzitutto definiamo l'insieme **LP** delle lettere proposizionali di interesse come $\{r_1, v_1, r_2, v_2\}$, che rappresentano l'accensione delle varie luci. I requisiti a-d) sono di facile traduzione, mentre per quanto riguarda il requisito e), esso esprime la necessità che per ogni istante di tempo (G) esista un istante successivo (F) in cui v_1 sia vera, ovvero che valga $GF\ v_1$. Notiamo che gli istanti in cui v_1 è vera non sono necessariamente consecutivi. Un'altra formula che rende vera v_1 un numero infinito di volte è $FG\ v_1$; poiché quest'ultima pone il vincolo non necessario che tali istanti siano consecutivi, ad essa va preferita la prima. Riassumendo, i requisiti a-f) possono essere rappresentati mediante le seguenti formule di LTL:

a') $G(r_1 \equiv \neg v_1)$;

b') $G(r_2 \equiv \neg v_2)$;

c') $G(v_1 \rightarrow r_2)$;

d') $G(v_2 \rightarrow r_1)$;

e') $GF\ v_1$;

f') $GF\ v_2$.

Notiamo che è possibile definire una struttura temporale che soddisfa le sei formule a'-f'): sia ϕ la loro congiunzione e M la struttura temporale di figura 6.5; poiché ϕ è vera nel cammino $s_0, s_1, s_2, s_1, \ldots$ ϕ è soddisfacibile (cfr. definizione 6.2.6). In effetti ϕ è vera in tutti i cammini di M, quindi vale $M \models \phi$ (cfr. definizione 6.2.9).

∘

Esercizio 6.8 [Soluzione a pagina 86] Con riferimento all'esempio 6.3.2, progettare una struttura temporale N per i cui stati s valga sempre $N, s \not\models \phi$ (cfr. definizione 6.2.11). ∘

Anche l'esempio successivo tratta requisiti in linguaggio naturale, e ci servirà per evidenziare che esistono importanti classi di frasi in italiano che *non è possibile* tradurre con formule di LTL.

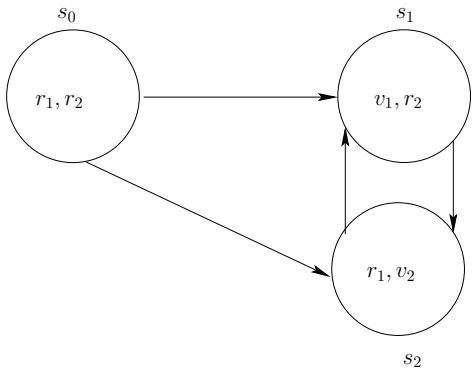

Figura 6.5 Struttura temporale che soddisfa le formule a'-f') dell'esempio 6.3.2

Esempio 6.3.3 Un sistema ha due luci, ciascuna delle quali può essere accesa o spenta. Consideriamo i seguenti requisiti:

a) prima o poi entrambe le luci saranno accese;

b) entrambe le luci sono e rimarranno accese;

c) in ogni stato è possibile fare sì che prima o poi entrambe le luci siano accese.

Sia $\{l_1, l_2\}$ l'insieme **LP** delle lettere proposizionali di interesse, dove ciascuna lettera rappresenta l'accensione di una delle due luci. I requisiti a-b) fanno riferimento ad una *necessità*, e sono agevolmente tradotti con formule di LTL:

a') $F(l_1 \wedge l_2)$;

b') $G(l_1 \wedge l_2)$.

Il requisito c) fa invece riferimento ad una combinazione di *possibilità* e *necessità*, che non è esprimibile in LTL. Per chiarire meglio questo aspetto, consideriamo la struttura temporale M di figura 6.6. Per ogni stato s di essa vale (cfr. definizione 6.2.11):

- $M, s \not\models F(l_1 \wedge l_2)$ e
- $M, s \not\models G(l_1 \wedge l_2)$.

Intuitivamente, possiamo viceversa affermare che M soddisfa il requisito c), in quanto lo stato s_3 (in cui entrambe le luci sono accese) è raggiungibile da tutti gli stati. Tale requisito può essere espresso in CTL attraverso la formula:

c') $AG\ EF(l_1 \wedge l_2)$;

dove AG ed EF (acronimi dell'inglese *Always Globally* ed *Eventually in the Future*, rispettivamente) sono connettivi temporali che servono per rappresentare rispettivamente:

- la *necessità* (A) che una proprietà sia vera in *tutti* (G) i cammini, e
- la *possibilità* (E) che una proprietà sia vera in *qualche* (F) cammino.

La semantica di connettivi temporali come AG ed EF viene usualmente data facendo riferimento ad alberi infiniti come quello mostrato parzialmente in figura 6.2: data una proprietà ϕ, affinché $EF\ \phi$ sia vera nella radice è sufficiente che ϕ sia vera in un nodo qualsiasi dell'albero; affinché $AG\ \phi$ sia vera nella radice è sufficiente che ϕ sia vera in tutti i nodi dell'albero.

○

L'esempio appena visto mostra che esistono proprietà esprimibili in CTL e non in LTL; rimandando a testi specifici per lo studio di CTL, per completezza ricordiamo che:

- CTL ha un insieme di connettivi temporali diverso da LTL;
- esistono proprietà esprimibili in LTL e non in CTL;
- la logica CTL* contiene strettamente sia LTL sia CTL, in quanto permette di esprimere tutte le proprietà esprimibili in esse ed anche alcune in più.

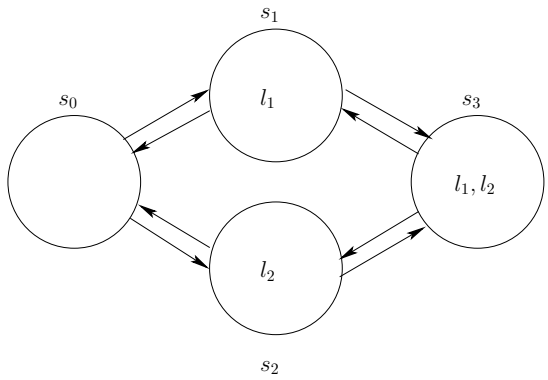

Figura 6.6 Struttura temporale per il sistema di accensione delle luci

6.4 Aspetti computazionali

In questo paragrafo ci concentriamo su alcuni aspetti computazionali delle nozioni viste, fra cui la complessità computazionale, gli algoritmi e i programmi per i problemi di ragionamento su formule di LTL.

6.4.1 Complessità computazionale

Dal punto di vista della complessità computazionale sono stati studiati principalmente due problemi:

soddisfacibilità: data una formula ϕ di LTL, verificare se ϕ è soddisfacibile (cfr. definizione 6.2.6);

model checking: data una formula ϕ di LTL, una struttura temporale $M = \langle S, R, L\rangle$ e uno stato iniziale $s \in S$, verificare se vale $M, s \models \phi$ (cfr. definizione 6.2.11).

Per LTL entrambi questi problemi sono PSPAZIO-completi. Ricordiamo che i problemi di questa classe di complessità sono decidibili e ritenuti essere i più difficili fra quelli per la cui soluzione è sufficiente spazio polinomiale. Non si ritiene che per essi esistano algoritmi di tempo polinomiale, e in pratica risultano essere più difficili di quelli NP-completi come SAT (cfr. paragrafo 3.4.1).

Ricordiamo che il problema del test di validità di una formula ϕ di LTL si riduce a quello del test di insoddisfacibilità di $\neg\phi$ (cfr. esercizio 6.3).

6.4.2 Algoritmi

Nell'ambito di LTL sono stati proposti vari algoritmi per la soluzione del problema del model checking e di quello, dati $M = \langle S, R, L\rangle$ e ϕ, del calcolo dell'insieme $\{s \mid s \in S \wedge M, s \models \phi\}$ degli stati inziali che rendono vera la valutazione della struttura temporale M.

La trattazione completa di tali algoritmi è piuttosto complessa ed esula dai nostri scopi; in questo testo ci limitiamo a descriverne alcuni aspetti generali. Notiamo innanzitutto che una prima difficoltà del problema del model checking in LTL è data dal fatto che è necessario valutare la formula ϕ rispetto a tutti i cammini di un insieme infinito (cfr. ad esempio figura 6.2). In realtà è stato dimostrato che è sufficiente prendere in considerazione solo strutture con un numero finito di stati.

Molti algoritmi per il model checking si articolano su tre passi principali:

1. Viene costruito un automa $A_{\neg\phi}$ (chiamato *automa di Büchi*) che caratterizza la formula $\neg\phi$ attraverso le *tracce* (sequenze di interpretazioni della logica proposizionale) che esso accetta. In particolare $A_{\neg\phi}$ codifica tutte le tracce che non soddisfano ϕ.

2. $A_{\neg\phi}$ viene messo in relazione con la struttura temporale M, ottenendo il cosiddetto *automa prodotto*. Questo automa caratterizza sia la formula ϕ sia la struttura temporale M.

3. La validità di $M, s \models \phi$ corrisponde ad una precisa proprietà dell'automa prodotto, che viene valutata nel terzo passo.

La complessità computazionale di questi algoritmi è $O(|S|^2 \cdot 2^{|\phi|})$, ovvero è quadratica nel numero $|S|$ di stati della struttura temporale $M = \langle S, R, L\rangle$ ed esponenziale nella dimensione della formula ϕ. In particolare il primo passo ha costo $O(2^{|\phi|})$, in quanto l'automa $A_{\neg\phi}$ contiene un numero di stati

proporzionale al numero di sottoformule di ϕ. Poiché il numero di archi di R in M non è mai superiore a $|S|^2$, è comune dire che gli algoritmi hanno costo lineare nella dimensione della struttura temporale (modello) ed esponenziale nella dimensione della formula (specifica).

È utile un commento sulla linearità degli algoritmi di model checking rispetto alla dimensione del modello. Il numero di stati $|S|$ è, nel caso peggiore, esponenziale nel numero di variabili, ovvero può essere pari a $2^{|\mathbf{LP}|}$. In pratica tale limite viene frequentemente raggiunto, e ciò implica che aggiungere una sola variabile proposizionale può raddoppiare il numero di stati. Questa tendenza dello spazio degli stati a crescere rapidamente prende il nome di *problema dell'espolosione degli stati*, ed è uno dei problemi che hanno ricevuto maggiore attenzione nello sviluppo di algoritmi e programmi per il model checking. Fra le tecniche di maggiore utilità che sono state utilizzate si trova l'uso dei cosiddetti *OBDD*, acronimo dell'inglese *ordered binary decision diagrams*, che sono strutture dati per la rappresentazione di funzioni booleane.

6.4.3 Programmi disponibili

Fra i vari programmi esistenti per il model checking delle logiche temporali proposizionali, SPIN e NUSMV sono disponibili in forma gratuita, vengono usati frequentemente in ambito accademico e hanno ottenuto importanti riconoscimenti anche in quello industriale.

SPIN, disponibile alla pagina `netlib.bell-labs.com/netlib/spin/whatispin.html`, viene usato principalmente per la verifica di sistemi software asincroni, in particolare di protocolli di comunicazione. Il programma permette il model checking di specifiche in LTL, ed usa sofisticate tecniche per la riduzione dello spazio degli stati. Le strutture temporali vengono specificate mediante un apposito linguaggio chiamato PROMELA, che permette di rappresentare in maniera agevole processi sequenziali, variabili locali, globali e canali di comunicazione.

NUSMV, disponibile con licenza *open source* alla pagina `nusmv.irst.itc.it`, è una reimplementazione ed estensione del sistema SMV (acronimo di *symbolic model verifier*), sviluppato originariamente presso la *Carneige-Mellon University*. Quest'ultimo permette la verifica di specifiche in CTL per processi cooperanti che comunicano attraverso variabili condivise. NUSMV ha un più ampio linguaggio per le specifiche e nel suo sviluppo sono state introdotte numerose funzionalità nuove.

Dal punto di vista linguistico, NUSMV fornisce la possibilità di specificare *moduli*, che rappresentano strutture temporali mediante variabili appartenenti a domini finiti (ad es., variabili booleane) e relazioni di transizione mediante opportuni costrutti. Inoltre è possibile dichiarare lo stato iniziale delle variabili. È possibile specificare formule in CTL, LTL e LTL con operatori per il passato.

Fra le innovazioni più interessanti di NUSMV esiste la possibilità di effettuare il cosiddetto *bounded model checking*, ovvero model checking limitato. La caratteristica fondamentale del bounded model checking risiede nella perdita di correttezza della verifica, dovuta al fatto che viene considerata una lunghezza limitata (e non più infinita come nella definizione 6.2.11) per i cammini nelle strutture temporali. La semantica delle formule temporali cambia di conseguenza: ad esempio, specificando un limite pari a 10, la formula $M, s \models \phi$ è vera se per ogni cammino $\pi = s_i^0, \ldots, s_i^b$ (con $0 \leq b \leq 10$) di M di origine $s = s_i^0$ e lunghezza non superiore a 10 vale $M, \pi \models \phi$.

Esempio 6.4.1 Detta M la struttura temporale di figura 6.1, abbiamo:

- $M, s_3 \not\models G\ \neg(p \wedge r)$, rispetto alla definizione 6.2.11, come testimoniato dal cammino infinito $s_3, s_1, s_2, s_3, \ldots$;
- $M, s_3 \models G\ \neg(p \wedge r)$, se si considera bounded model checking con limite alla lunghezza dei cammini pari a 1;
- $M, s_3 \not\models G\ \neg(p \wedge r)$, se si considera bounded model checking con limite alla lunghezza dei cammini pari a 2.

$\circ$

Il vantaggio dell'uso del bounded model checking risiede nella possibilità di usare altri algoritmi oltre a quelli menzionati nel paragrafo 6.4.2. In particolare, è possibile ridurre un'istanza del problema del bounded model checking ad un'istanza di SAT, e usare uno dei programmi menzionati nel paragrafo 3.4.3 per la soluzione di quest'ultima. Se il limite alla lunghezza del cammino non è molto elevato, spesso questo metodo è considerevolmente più efficiente del metodo corretto e completo. Per quanto riguarda la scelta del limite, essa è dettata sia da considerazioni relative all'efficienza, sia dal tipo di informazioni che si desidera ottenere con il model checking. Infatti l'uso del model checking è spesso orientato alla

PROPRIETÀ	def.	VAR	DEFINE	ASSIGN	TRANS	LTLSPEC
validità	6.2.6	**LP**	-	-	-	ϕ
soddisfacibilità	6.2.6	**LP**	-	-	-	$\neg\phi$
model checking senza stato iniziale	6.2.9	**LP**	S	-	R	$s \in S \rightarrow \phi$
model checking con stato iniziale s	6.2.11	**LP**	S	s	R	ϕ

Tabella 6.3 Uso delle sezioni di NuSMV per la verifica di proprietà.

scoperta di errori nelle implementazioni, e non alla dimostrazione della loro inesistenza. Di conseguenza, l'uso del bounded model checking può portare ad evidenziare errori significativi in tempi ragionevoli.

Nel paragrafo 7.4.2.2 considereremo un programma basato su un simile approccio, ovvero la limitazione della dimensione delle strutture per la verifica, per un problema completamente diverso.

6.5 Esercitazione pratica

Il primo scopo di questa esercitazione (cfr. pararafo 6.5.1) è mostrare come sia possibile utilizzare il calcolatore per risolvere i problemi computazionali indotti dalle definizioni del paragrafo 6.2. Allo scopo useremo NuSMV, versione 2.4.0. Chiariamo subito che, anche se esso offre un ricco linguaggio per la descrizione di processi e costrutti linguistici che permettono di rappresentare in maniera compatta una struttura temporale, in questo capitolo useremo solamente un sottoinsieme del linguaggio molto simile a LTL. In vari capitoli successivi vedremo che NuSMV può essere usato anche con altri obiettivi, più vicini alle problematiche tradizionali dell'ingegneria del software.

Il secondo scopo è esercitarsi sulla rappresentazione in LTL di frasi in italiano come quelle viste nel paragrafo 6.3. In particolare (cfr. pararafo 6.5.2) tradurremo in LTL i requisiti per il progetto di sistemi quali un forno a microonde e un protocollo per l'elezione del leader in un sistema distribuito. Nel paragrafo 6.5.3 ci occuperemo dell'implementazione di tali sistemi tramite strutture temporali ed useremo le tecniche acquisite nel paragrafo 6.5.1 per la verifica automatica tramite NuSMV della correttezza dell'implementazione rispetto alla specifica.

6.5.1 Prima parte

In questo paragrafo ci occuperemo della verifica tramite NuSMV delle definizioni del paragrafo 6.2, in particolare:

- verifica di validità, soddisfacibilità e insoddisfacibilità di formule (cfr. definizione 6.2.6);
- valutazione di una formula rispetto a una struttura temporale, con o senza stato iniziale (model checking, cfr. definizioni 6.2.9 e 6.2.11).

I file NuSMV consentono la definizione di uno o più moduli; quello obbligatorio (`MODULE main`) sarà l'unico necessario in questo capitolo. Nel modulo `main` verranno sempre dichiarate le variabili (`VAR`) necessarie per rappresentare le formule di LTL di interesse, ovvero l'alfabeto **LP**. La formula di LTL viene rappresentata nella sezione `LTLSPEC`. Il modulo può contenere altri elementi, che saranno chiariti di volta in volta. La tabella 6.3 contiene le indicazioni di interesse in maniera schematica.

La sintassi di NuSMV non ammette alcuni connettivi temporali (R e W), che vanno sostituiti con quelli forniti (X, F, G e U) tramite opportune equivalenze, ad esempio (cfr. esercizio 6.5):

- $(p\ R\ q) \equiv \neg\ (\neg p\ U\ \neg q)$,
- $(p\ W\ q) \equiv ((p\ U\ q) \vee G\ p)$

6.5.1.1 Verifica di validità e soddisfacibilità

Per la verifica di tautologie scriviamo un file NuSMV in cui la sezione `LTLSPEC` contiene la formula la cui validità si desidera verificare o confutare. Il seguente file è utile per verificare che le tutte formule di LTL dell'esercizio 6.2 sono valide.

```
-- Time-stamp: "2006-10-12 09:51:26 cadoli"
-- File: 4tautologie.smv
-- Descrizione: alcune tautologie di LTL

MODULE main
VAR
  p: boolean;
  q: boolean;
-- fine MODULE main
LTLSPEC
-- (1)
-- (G p) -> p
-- (2)
-- p -> q U p
-- (3)
-- p -> F p
-- (4)
-- (!p W q) -> G(F p -> F G q)
-- NuSMV non ha il connettivo temporale W, quindi dobbiamo riscrivere
-- la formula precedente utilizzando la seguente tautologia:
-- (p W q) <-> ((p U q) | G p)
-- ottenendo così:
((!p U G q) | G !p) -> (G(F p -> F G q))
```

Il sistema riconosce la validità delle formule con un opportuno messaggio. Ad esempio per la quarta formula otteniamo il messaggio:

```
-- specification  ((((!p) U  G q) |  G (!p)) ->  G ( F p ->  F  G q))  is true
```

Esercizio 6.9 [Soluzione a pagina 86] Scrivere un file NuSMV per dimostrare che le formule dell'esercizio 6.5 sono valide. ◦

Esercizio 6.10 Trovare almeno altre due formule valide e due non valide. Per quanto riguarda le formule non valide, fornire un controesempio opportuno.

Rappresentare tutte le formule mediante la sintassi di NuSMV e usare il sistema per convalidare le proprie ipotesi. ◦

Poiché la soddisfacibilità di una formula coincide con la non validità della sua negazione (cfr. esercizio 6.3), il metodo appena visto può essere usato anche per la verifica della soddisfacibilità. Con riferimento all'esempio 6.3.2, la soddisfacibilità della formula ϕ che rappresenta la congiunzione dei requisiti del sistema per l'accensione dei semafori può essere mostrata inserendo $\neg\phi$ nella sezione `LTLSPEC`, ottenendo il seguente file.

```
-- Time-stamp: "2006-10-12 10:18:04 cadoli"
-- File: semaforo.smv
-- Descrizione: verifica soddisfacibilità delle specifiche in LTL per un
-- sistema che regola l'accensione di 2 semafori

MODULE main
VAR
  r1: boolean;
  v1: boolean;
  r2: boolean;
  v2: boolean;
LTLSPEC
!(
     G(r1 <-> ! v1) & -- a)
     G(r2 <-> ! v2) & -- b)
     G(v1 -> r2)    & -- c)
     G(v2 -> r1)    & -- d)
     G F v1         & -- e)
     G F v2           -- f)
)
```

Poiché $\neg\phi$ non è valida, NuSMV restituisce un controesempio (o traccia) che lo testimonia, ovvero una struttura temporale M e un cammino π in essa tale che $M, \pi \models \phi$, secondo la definizione 6.2.6.

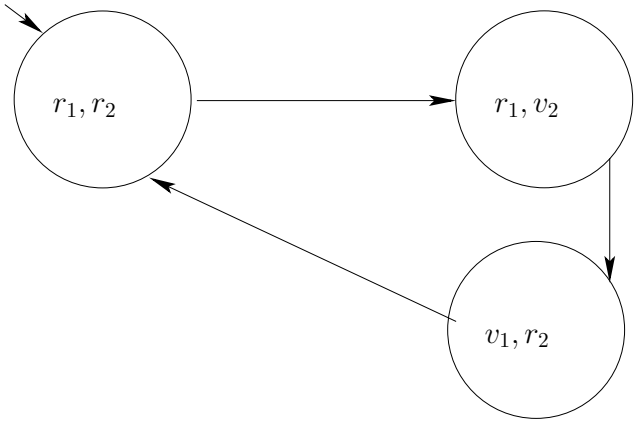

Figura 6.7 Cammino che prova la soddisfacibilità dei requisiti dell'esempio 6.3.2

```
-- specification  !((((( G (r1 <-> !v1) &  G (r2 <-> !v2)) &  G (v1 -> r2)) &
--                  G (v2 -> r1)) &  G  F v1) &  G  F v2)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
-- Loop starts here
-> State: 1.1 <-
  r1 = 1
  v1 = 0
  r2 = 1
  v2 = 0
-> State: 1.2 <-
  r2 = 0
  v2 = 1
-> State: 1.3 <-
  r1 = 0
  v1 = 1
  r2 = 1
  v2 = 0
-> State: 1.4 <-
  r1 = 1
  v1 = 0
```

Si noti che ad ogni istante temporale vengono rappresentate solamente le variabili di stato che cambiano. Poiché il cammino deve essere infinito (cfr. definizione 6.2.3), esso deve essere ciclico. Un estremo del ciclo viene indicato dalla dicitura "`Loop starts here`". La figura 6.7 mostra la traccia generata da NuSMV, con lo stato iniziale evidenziato da una freccia entrante "dal nulla". Rispetto alla soluzione di figura 6.5, questa soluzione obbliga a commutare a rosso entrambi i semafori ad ogni iterazione.

Esercizio 6.11 [Soluzione a pagina 87] Scrivere un file NuSMV per la verifica della soddisfacibilità dei requisiti a'-b') in LTL dell'esempio 6.3.3. Confrontare la soluzione ottenuta con la struttura temporale di figura 6.6. ○

6.5.1.2 Model checking

NuSMV può essere usato anche per rappresentare una struttura temporale $M = (S, R, L)$ e verificare quali formule siano valutate a T in esso, secondo la definizione 6.2.9 o 6.2.11.

Il file NuSMV adatto a tale scopo ha il seguente formato:

- seguendo le indicazioni generali:
  - viene dichiarato solo il modulo obbligatorio (`MODULE main`),
  - in tale modulo vengono dichiarate le variabili necessarie (`VAR`) per rappresentare la formula da valutare;

  - la specifica LTL (`LTLSPEC`) contiene la formula da valutare; se la formula va verificata a partire da un certo stato (cfr. definizione 6.2.11), questo va specificato ponendolo a sinistra di un'implicazione logica "`->`"; altrimenti, va specificata la disgiunzione degli stati ammissibili;

- viene dichiarata la relazione di transizione (`TRANS`), specificando vari "casi" e tenendo conto che in NuSMV viene eseguito solamente il primo caso il cui antecedente è vero:
  - per ogni stato s di S va specificato un caso (`case`);
  - nell'antecedente del caso (a sinistra di "`:`") va specificato $L(s)$, ovvero l'insieme dei valori di verità di tutte le variabili (sia quelle vere che quelle false);
  - nel conseguente del caso (a destra di "`:`") vanno specificati tutti gli stati s' tali che $R(s, s')$; in particolare occorre:
    * specificare il valore delle variabili all'istante successivo (`next`);
    * separare i vari stati s' mediante il connettivo "`|`" (ovvero "or", per esprimere il non determinismo);
  - le "non-transizioni" di M vengono catturate tutte insieme con un "caso di default"; tale caso ("`1`", cioè "vero") specifica che, all'istante successivo lo stato non cambia;

- è possibile dare un nome esplicito agli stati, mediante la sezione opzionale `DEFINE`, che definisce nuove variabili facendo uso di quelle della sezione `VAR`;

- se lo stato iniziale è determinato (cfr. definizione 6.2.11) lo si può specificare mediante la sezione opzionale `ASSIGN`, che specifica lo stato iniziale (`init`) delle variabili.

Il seguente file fa riferimento alla definizione 6.2.9, all'esempio 6.2.10 e all'esercizio 6.6: lo stato iniziale non è specificato.

```
-- Time-stamp: "2006-10-12 10:20:17 cadoli"
-- File: modelChecking.smv

MODULE main
VAR
  p: boolean;
  q: boolean;
  r: boolean;
TRANS
-- descrive la relazione di transizione
  case
-- s1: s2 | s3;
    s1: (next(p) = 1 & next(q) = 0 & next(r) = 1) |
        (next(p) = 0 & next(q) = 1 & next(r) = 0);
-- s2: s2 | s3;
    s2: (next(p) = 1 & next(q) = 0 & next(r) = 1) |
        (next(p) = 0 & next(q) = 1 & next(r) = 0);
-- s3: s1
    s3: (next(p) = 1 & next(q) = 1 & next(r) = 0);
-- s4: s4
    s4: (next(p) = 1 & next(q) = 1 & next(r) = 1);
-- non-transizioni
     1: next(p) = p & next(q) = q & next(r) = r;
  esac
DEFINE
-- i possibili stati del sistema, definiti tramite le variabili
s1 := p & q & !r;
s2 := p & !q & r;
s3 := !p & q & !r;
s4 := p & q & r;
s  := s1 | s2 | s3 | s4;
-- fine MODULE main

LTLSPEC
s -> (
-- esempio
  (G(p | q)) &          --T
```

```
  (F p) &                --T
  (G r -> G p) &         --T
-- esercizio
  (F X p) &              --T
--  (F q) &              --F
  (G ((p & !q) -> r)) & --T
--  (G p -> G r) &       --F
  (G (q U p)) &          --T
--  (G (p U q)) &        --F
  (G p -> (p U r))       --T
)
```

Il seguente frammento di file fa riferimento alla definizione 6.2.11 e all'esempio 6.2.12: lo stato iniziale (s_4) è specificato.

```
ASSIGN
-- s4
  init(p) := 1;
  init(q) := 1;
  init(r) := 1;
LTLSPEC
G p
```

Esercizio 6.12 Trovare almeno altre due formule vere e due non vere nella struttura temporale di cui sopra. Per quanto riguarda le formule non vere, fornire un controesempio opportuno.

Rappresentare tutte le formule mediante la sintassi di NuSMV e usare il sistema per convalidare le proprie ipotesi. ○

6.5.2 Seconda parte

In questo paragrafo ci esercitiamo sulla rappresentazione in LTL di frasi in italiano, in particolare tradurremo in LTL i requisiti per opportune semplificazioni di sistemi del mondo reale.

6.5.2.1 Forno a microonde

In un forno a microonde ci interessano i seguenti aspetti:

- l'emettitore (E) di microonde, che può essere attivo (ovvero sta riscaldando) oppure no;
- lo sportello (S), che può essere aperto o chiuso;
- il timer (T), che può indicare un tempo maggiore di zero oppure no.

Per modellare ciascuno degli aspetti usiamo una variabile proposizionale:

- r, vera se E è attivo;
- a, vera se S è aperto;
- t, vera se T indica un tempo maggiore di zero.

La nostra analisi riguarda un intervallo di tempo fra due settaggi di T da parte dell'operatore. In questo intervallo (in cui inizialmente è vero t) lo sportello può essere aperto e richiuso un numero arbitrario di volte; quando lo sportello è chiuso (è vero $\neg a$), il timer si scarica e l'emettitore è attivo (è vero r); quando è aperto (è vero a), il timer non si scarica l'emettitore non è attivo (è vero $\neg r$). La fine dell'intervallo è data da un ulteriore settaggio di T; prima di ciò, T può arrivare a indicare un tempo pari a zero e t sarà falsa (cfr. figura 6.8).

Esercizio 6.13 [Soluzione a pagina 87] Questo esercizio si articola su vari passi:

1. Scrivere in italiano un insieme di requisiti da rispettare nella progettazione del forno. I requisiti devono, fra l'altro, garantire l'incolumità dell'operatore e la terminazione del riscaldamento.
2. Tradurre i requisiti in una formula ϕ di LTL.
3. Progettare una struttura temporale M e uno stato iniziale s_0 che rappresenti il funzionamento del forno.
4. Verificare che valga $M, s_0 \models \phi$, secondo la definizione 6.2.11.

○

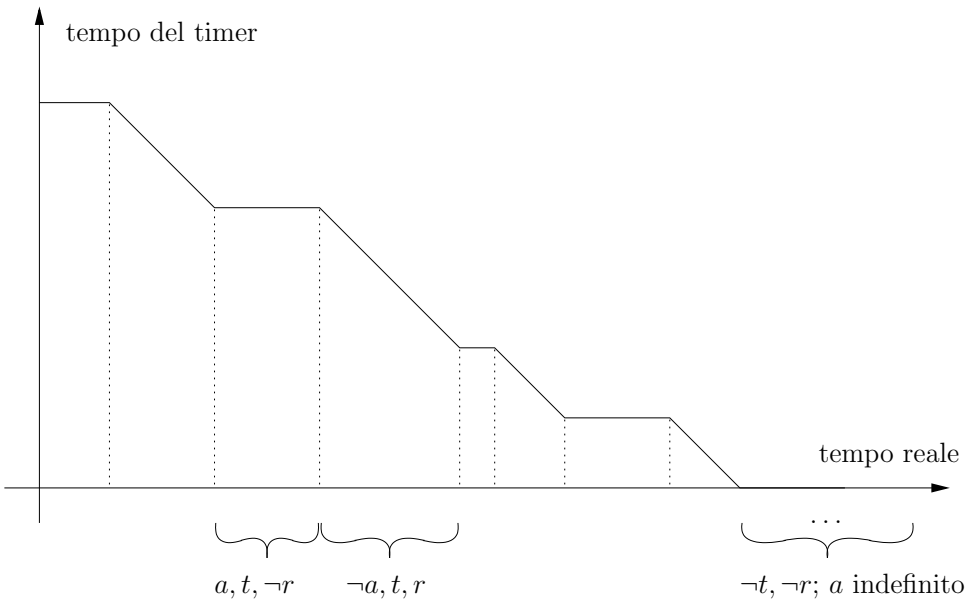

Figura 6.8 Andamento del tempo indicato dal timer rispetto al tempo reale

6.5.2.2 Elezione del leader in un sistema distribuito[1]

Nei sistemi distribuiti esistono servizi (ad esempio il bilanciamento del carico o il coordinamento di una query in un DBMS distribuito) che possono essere effettuati da vari processi del sistema. Tuttavia, per ragioni di consistenza, in ogni istante solamente un processo ha il permesso di erogare servizi di questo tipo. Di conseguenza un processo (chiamato il "leader") deve essere eletto fra tutti i processi, tenendo conto della loro efficacia nell'erogare il servizio. Nel seguito di questo paragrafo assumeremo che 1) i processi abbiano un identificatore preso da un dominio ordinato (ad es., gli interi) e 2) l'efficacia di un candidato venga rappresentata semplicemente dal suo identificatore. Ci occuperemo dell'elezione del candidato con le migliori qualità, in particolare vogliamo che diventi leader il processo con l'identificatore più alto.

Le regole che determinano l'elezione del leader sono le seguenti:

1. il numero di processi partecipanti è finito;
2. i processi comunicano attraverso un canale in maniera asincrona;
3. ogni processo ha un identificatore;
4. esiste un ordinamento totale fra gli identificatori;
5. ogni processo può essere attivo o non attivo;
6. un processo può essere eletto solo se è attivo;
7. inizialmente nessun processo è attivo;
8. prima o poi ogni processo diventa attivo;
9. un processo attivo non può mai diventare non attivo;
10. in ogni istante c'è sempre al più un leader;
11. prima o poi c'è almeno un leader;
12. se c'è un processo attivo con identificatore maggiore, prima o poi il leader dà le dimissioni;
13. un nuovo leader deve migliorare il leader precedente.

[1] Questo paragrafo è una rielaborazione del materiale contenuto in [24, par. 2.5].

Esercizio 6.14 [Soluzione a pagina 88] Questo esercizio si articola su vari passi:

1. Basandosi sulle regole di cui sopra, scegliere un numero intero positivo che rappresenta il numero di processi del sistema e formulare i requisiti da rispettare nella progettazione del metodo che gestisce l'elezione del leader tramite una formula ϕ di LTL.
2. Progettare una struttura temporale M e uno stato iniziale s_0 che rappresenti il funzionamento del meccanismo di elezione.
3. Verificare che valga $M, s_0 \models \phi$, secondo la definizione 6.2.11.

◦

6.5.3 Terza parte

In questo paragrafo ci occupiamo della verifica automatica tramite NuSMV dei sistemi visti nel paragrafo 6.5.2.

Esercizio 6.15 [Soluzione a pagina 89] Tradurre in NuSMV i requisiti scritti per l'esercizio 6.13 e verificare in maniera automatica il punto 4. ◦

Esercizio 6.16 [Soluzione a pagina 90] Tradurre in NuSMV i requisiti scritti per l'esercizio 6.14 e verificare in maniera automatica il punto 3. ◦

6.6 Nota bibliografica

La logica LTL è stata proposta originalmente in [32, 33]. Per una trattazione più estesa della logica temporale, del model checking e dei relativi algoritmi si rimanda a testi specifici, come [15, 25, 11, 24, 21]. La complessità computazionale di LTL è stata studiata in [40]. Oltre alla documentazione disponibile nel suo sito Web, Spin viene trattato in numerosi libri e articoli, fra cui [19, 20]. La pagina Web di SMV è `www.cs.cmu.edu/~modelcheck`, e un testo che lo tratta è [28]. La pagina Web di NuSMV possiede un'ampia documentazione, ed ulteriori aspetti tecnici sono sviluppati in varie pubblicazioni fra cui [4, 10].

6.7 Esercizi proposti

Esercizio 6.17 [Soluzione a pagina 91] Con riferimento alla struttura temporale M di figura 6.9, calcolare eval^M per ognuna delle seguenti formule:

1. $G\ \neg(a\ \wedge\ c)$
2. $F(\neg b\ \wedge\ c)\ \rightarrow\ FG\ c$
3. $((\neg b\ \wedge\ c)\ U\ c)\ \vee\ FG(\neg a\ \wedge\ c)$

Inoltre calcolare le seguenti espressioni:

4. $\mathsf{eval}^{M,s_1}(a\ \wedge\ b)$
5. $\mathsf{eval}^{M,s_1}(X\ c)$
6. $\mathsf{eval}^{M,s_1}(b\ \wedge\ c)$
7. $\mathsf{eval}^{M,s_3}(G\ c)$
8. $\mathsf{eval}^{M,s_1}((GF\ a)\ \rightarrow\ GF\ c)$
9. $\mathsf{eval}^{M,s_1}((GF\ c)\ \rightarrow\ GF\ a)$

Per tutte le formule che vengono valutate a F esibire un opportuno controesempio. ◦

Esercizio 6.18 [Soluzione a pagina 91] Con riferimento alla struttura temporale M di figura 6.1, considerare la formula $\phi = F\ \neg q$ e dire:

1. se vale $M, s_3 \models \phi$ oppure no, rispetto alla definizione 6.2.11;

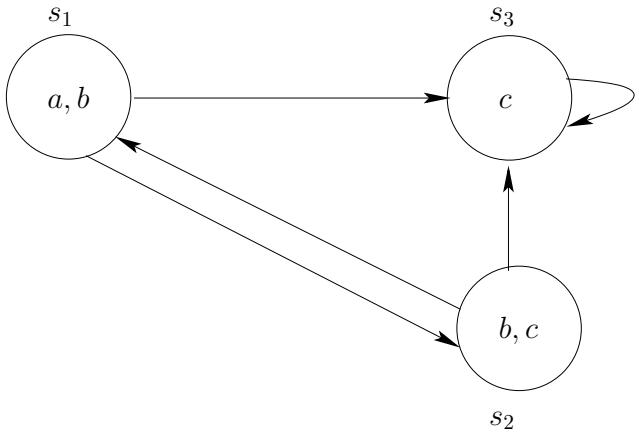

Figura 6.9 Una struttura temporale

2. per quali valori b del limite, se esistono, vale $M, s_3 \models \phi$ per il bounded model checking.

○

Esercizio 6.19 [Soluzione a pagina 91] In maniera simile a quanto effettuato per il concetto di bounded model checking, è possibile considerare cammini di lunghezza limitata anche nel contesto di altre definizioni. Ad esempio, possiamo considerare un ampliamento della definizione 6.2.6 in cui una formula ϕ è definita essere una *b-tautologia* se in ogni struttura temporale M e in ogni cammino π di lunghezza inferiore o uguale ad un certo limite b ($b \geq 1$) vale $M, \pi \models \phi$.

Per le seguenti formule:

1. $p \wedge X\ p \wedge X\ X\ p \rightarrow G\ p$,
2. $F\ p \rightarrow p$,
3. $q\ U\ p \rightarrow p$,
4. $F\ p \rightarrow (p \vee X\ p)$,

dire se esiste un b ($b \geq 1$) tale ch esse sono b-tautologie. ○

Esercizio 6.20 [Soluzione a pagina 92] Con riferimento all'esercizio 6.17, verificare tramite NuSMV le risposte date. ○

6.8 Soluzione di alcuni esercizi

Soluzione esercizio 6.1. La valutazione delle formule nei cammini specificati fornisce la seguente tabella.

	p	$X\ r$	Gp	$F\ \neg q$	$q\ U\ r$	$p\ W(q \wedge r)$	$p\ R\ r$
$\pi_3 = s_4, s_4, \ldots$	T	T	T	F	T	T	T
$\pi_4 = s_2, s_3, s_1, s_2, s_3, s_1, \ldots$	T	F	F	T	T	F	T

○

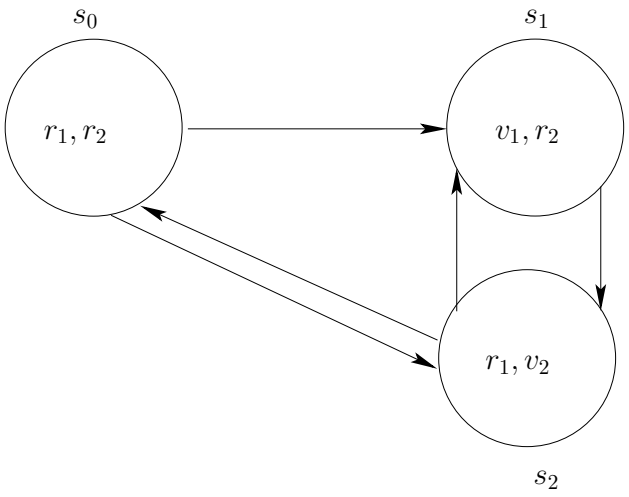

Figura 6.10 Struttura temporale per l'esercizio 6.8

Soluzione esercizio 6.6.

- $F\ X\ p$ T
- $F\ q$ F: $s_2, s_2, \ldots$
- $G(p \wedge \neg q \rightarrow r)$ T
- $G\ p \rightarrow G\ r$ F: $s_1, s_2, s_2, \ldots$
- $G(q\ U\ p)$ T
- $G(p\ U\ q)$ F: $s_1, s_2, s_2, \ldots$
- $G\ p \rightarrow p\ U\ r$ T

∘

Soluzione esercizio 6.7. Detta M la struttura temporale di figura 6.1, abbiamo:

- $M, s_4 \models F\ q$
- $M, s_4 \models G\ p \rightarrow G\ r$
- $M, s_4 \models G(p\ U\ q)$

∘

Soluzione esercizio 6.8. La struttura temporale N richiesta è mostrata in figura 6.10: rispetto a quella di figura 6.5 si noti l'aggiunta della transizione fra s_2 ed s_0. Tale transizione aggiuntiva impedisce che esista uno stato s tale che $N, s \models \phi$: la formula e') è violata nel cammino $s_0, s_2, \ldots$, in quanto v_1 è sempre falsa.

∘

Soluzione esercizio 6.9. Tralasciando le formule che coinvolgono connettivi temporali non supportati da NuSMV(cioè 4, 5, 13, 17, 26, 27, 28, 29, 30), il file NuSMV richiesto è il seguente.

```
-- Time-stamp: "2006-10-12 10:08:02 cadoli"
-- File: tanteTautologie.smv
-- Descrizione: alcune tautologie di LTL

MODULE main
VAR
  p: boolean;
  q: boolean;
LTLSPEC
```

```
(! G p <-> F ! p) &                 --1
(! F p <-> G ! p) &                 --2
(! X p <-> X ! p) &                 --3
(X (p | q) <-> (X p | X q)) &       --6
(X (p & q) <-> (X p & X q)) &       --7
(X (p -> q) <-> (X p -> X q)) &     --8
(X (p <-> q) <-> (X p <-> X q)) &   --9
(F(p | q) <-> (F p | F q) ) &       --10
(G(p & q) <-> (G p & G q)) &        --11
(X(p U q) <-> (X p U X q)) &        --12
(G G p <-> G p) &                   --14
(F F p <-> F p) &                   --15
(((p U q) U q) <-> (p U q)) &       --16
(F G F p <-> G F p) &               --18
(G F G p <-> F G p) &               --19
(F p <-> p | X F p) &               --20
(G p <-> p & X G p) &               --21
(p U q <-> q | (p & X (p U q))) &   --22
(X G p <-> G X p) &                 --23
(X F p <-> F X p) &                 --24
(F p <-> (1 U p))                   --25
```

○

Soluzione esercizio 6.11. Il file NuSMV richiesto è il seguente.

```
-- Time-stamp: "2006-10-12 10:17:40 cadoli"
-- File: luci.smv

MODULE main
VAR
  l1: boolean;
  l2: boolean;
LTLSPEC
!(
   F (l1 & l2) & -- prima o poi entrambe le luci saranno accese
   G (l1 & l2)   -- entrambe le luci sono e rimarranno accese
)
```

La traccia prodotta dal sistema (mostrata nel seguito) evidenzia che i requisiti vengono soddisfatti semplicemente lasciando sempre entrambe le luci accese.

```
-- specification  !( F (l1 & l2) &  G (l1 & l2))  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
-- Loop starts here
-> State: 1.1 <-
  l1 = 1
  l2 = 1
-> State: 1.2 <-
```

○

Soluzione esercizio 6.13.

1. Un insieme ragionevole di requisiti da rispettare nel progetto del sistema di controllo del forno è:

   a) in ogni istante, E è attivo se e solo se S è chiuso e T sta indicando un tempo maggiore di zero;

   b) in ogni istante, se S non verrà mai più aperto, prima o poi T indicherà tempo pari a zero;

   c) inizialmente S è chiuso, T è disattivo, ed E non sta riscaldando.

   Il requisito a) è essenziale per l'incolumità dell'operatore; b) ci assicura che il riscaldamento terminerà; c) dà all'operatore la possibilità di usare il forno.

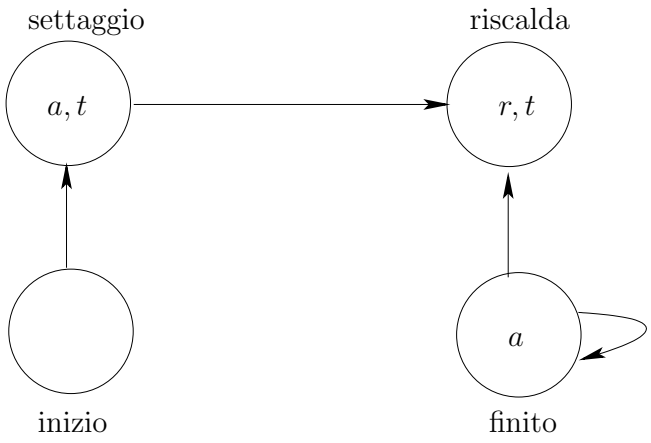

Figura 6.11 Struttura temporale per il forno a microonde

2. Innanzitutto definiamo l'insieme **LP** delle lettere proposizionali di interesse come $\{a, r, t\}$, come indicato nel testo dell'esercizio. I requisiti a-c) possono essere rappresentati mediante la congiunzione ϕ delle seguenti formule di LTL:

   a') $G((t \wedge \neg a) \equiv r)$

   b') $G(G\neg a \rightarrow F\neg t)$

   c') $(\neg a \wedge \neg t \wedge \neg r)$

3. La struttura temporale M è illustrata nella figura 6.11 e lo stato iniziale è quello etichettato con "inizio".

4. Vale $M, s_0 \models \phi$. Notiamo inoltre che abbiamo verificato la soddisfacibilità di ϕ secondo la definizione 6.2.6.

Notiamo che il riscaldamento avviene solo per un istante di tempo. Fasi di riscaldamento più lunghe possono essere ottenute aggiungendo in M un arco dal nodo etichettato con "riscaldamento" a se stesso. In tal modo tuttavia non viene più rispettato il requisito b'), in quanto il riscaldamento può continuare all'infinito anche con lo sportello chiuso. ◦

Soluzione esercizio 6.14.

1. Scegliamo un sistema distribuito di due processi. Definiamo l'insieme **LP** delle lettere proposizionali di interesse come $\{a_1, a_2, l_1, l_2\}$ che denotano, rispettivamente, lo stato di attività e di leadership dei due processi. La formula ϕ richiesta è data dalla congiunzione delle seguenti formule (è riportato un riferimento alle regole enunciate nella traccia dell'esercizio):

   a) $G(l_1 \rightarrow a_1 \ \wedge \ l_2 \rightarrow a_2)$
   6. i leader sono attivi

   b) $\neg a_1 \ \wedge \ \neg a_2$
   7. inizialmente i processi non sono attivi

   c) $F \ a_1 \ \wedge \ F \ a_2$
   8. un processo non può rimanere non attivo indefinitamente

   d) $(\neg a_1 \ U \ G \ a_1) \wedge (\neg a_2 \ U \ G \ a_2)$
   9. quando diviene attivo, un processo rimane tale

   e) $G(l_1 \rightarrow \neg l_2)$
   10. in ogni istante c'è sempre al più un leader

   f) $GF(l_1 \ \vee \ l_2)$
   11. prima o poi c'è almeno un leader

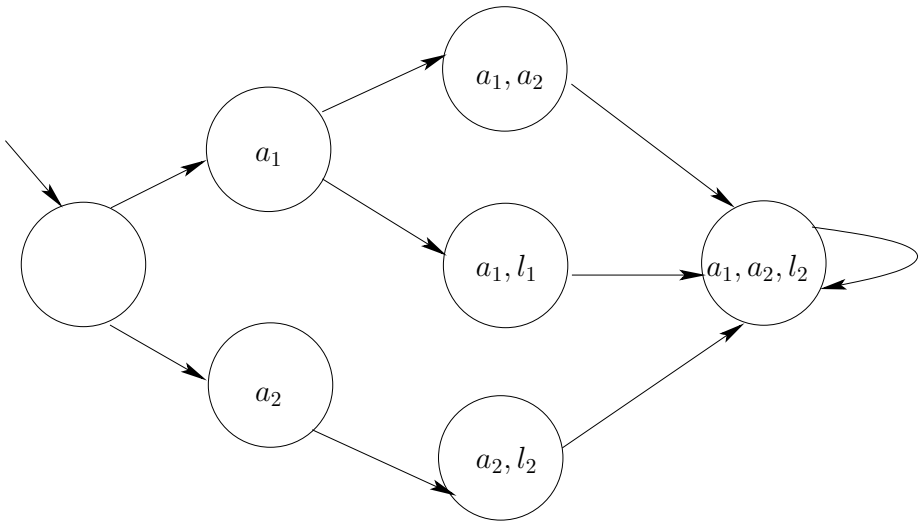

Figura 6.12 Struttura temporale che implementa il metodo di elezione del leader in un sistema con due processi

g) $G((l_1 \wedge \neg l_2 \wedge a_2) \rightarrow F \neg l_1)$
12. se c'è un processo attivo con identificatore maggiore, prima o poi il leader dà le dimissioni

h) $G(l_1 \wedge \neg X\ l_1 \rightarrow XF\ l_2)$
13. un nuovo leader deve migliorare il leader precedente

2. La struttura temporale M è illustrata nella figura 6.12 e lo stato iniziale s_0 è evidenziato con una freccia entrante "dal nulla".

3. Vale $M, s_0 \models \phi$.

$\circ$

Soluzione esercizio 6.15. Il file NuSMV richiesto è il seguente.

```
-- Time-stamp: "2006-10-12 11:26:37 cadoli"
-- File: microLTL.smv

MODULE main
VAR
  r: boolean; -- sta Riscaldando
  a: boolean; -- sportello Aperto
  t: boolean; -- Timer settato (<-> tempo richiesto > 0)
ASSIGN
  init(a) := 0;
  init(r) := 0;
  init(t) := 0;
DEFINE
  inizio   := !a & !r & !t;
  settaggio:=  a & !r &  t;
  riscalda := !a &  r &  t;
  finito   :=  a & !r & !t;
TRANS
 case
-- inizio     : settaggio
   inizio     : (next(a) = 1 & next(r) = 0 & next(t) = 1);
-- settaggio  : riscalda
   settaggio  : (next(a) = 0 & next(r) = 1 & next(t) = 1);
-- riscalda   : finito
   riscalda   : (next(a) = 1 & next(r) = 0 & next(t) = 0);
```

```
-- finito     : finito
   finito     : (next(a) = 1 & next(r) = 0 & next(t) = 0);
  1           : (next(a) = a & next(r) = r & next(t) = t);
 esac
LTLSPEC
G ((t & !a) <-> r)   -- A
&
G (G!a -> F!t)       -- B
&
(!a & !t & !r)       -- C
```

Come abbiamo notato nella soluzione dell'esercizio 6.13, se sostituiamo la regola per il riscaldamento con la seguente:

```
-- riscalda   : finito | riscalda
   riscalda   : (next(a) = 1 & next(r) = 0 & next(t) = 0) |
                (next(a) = a & next(r) = r & next(t) = t);
```

i requisiti non sono più rispettati, e NuSMV trova il controesempio in cui si continua all'infinito a riscaldare. ◦

Soluzione esercizio 6.16. Il file NuSMV richiesto è il seguente.

```
-- Time-stamp: "2006-10-16 16:32:32 cadoli"
-- File: leader2.smv
-- Scopo: elezione dinamnica del leader in sistema di due individui
MODULE main
VAR
  a1: boolean; -- 1 è attivo
  l1: boolean; -- 1 è leader
  a2: boolean; -- 2 è attivo
  l2: boolean; -- 2 è leader
ASSIGN
  init(a1):= 0;
  init(a2):= 0;
  init(l1):= 0;
  init(l2):= 0;
TRANS
 case
!a1 & !a2 & !l1 & !l2: -- {} -> {a1} | {a2}
   (next(a1) =  1 & next(a2) =  0 & next(l1) =  0 & next(l2) =  0) |
   (next(a1) =  0 & next(a2) =  1 & next(l1) =  0 & next(l2) =  0);
 a1 & !a2 & !l1 & !l2: -- {a1} -> {a1 a2} | {a1 l1}
   (next(a1) =  1 & next(a2) =  1 & next(l1) =  0 & next(l2) =  0) |
   (next(a1) =  1 & next(a2) =  0 & next(l1) =  1 & next(l2) =  0);
!a1 &  a2 & !l1 & !l2: -- {a2} -> {a2 l2}
   (next(a1) =  0 & next(a2) =  1 & next(l1) =  0 & next(l2) =  1);
 a1 &  a2 & !l1 & !l2: -- {a1 a2} -> {a1 a2 l2}
   (next(a1) =  1 & next(a2) =  1 & next(l1) =  0 & next(l2) =  1);
 a1 & !a2 &  l1 & !l2: -- {a1 l1} -> {a1 a2 l2}
   (next(a1) =  1 & next(a2) =  1 & next(l1) =  0 & next(l2) =  1);
!a1 &  a2 & !l1 &  l2: -- {a2 l2} -> {a1 a2 l2}
   (next(a1) =  1 & next(a2) =  1 & next(l1) =  0 & next(l2) =  1);
 a1 &  a2 & !l1 &  l2: -- {a1 a2 l2} -> {a1 a2 l2}
   (next(a1) =  1 & next(a2) =  1 & next(l1) =  0 & next(l2) =  1);
  1                  :
   (next(a1) = a1 & next(a2) = a2 & next(l1) = l1 & next(l2) = l2);
 esac
LTLSPEC
G(l1 -> a1 & l2 -> a2)       & -- A: i leader sono attivi
(!a1 & !a2)                  & -- B: inizialmente i processi non sono attivi
(F a1 & F a2)                & -- C: un processo non può rimanere non attivo
                               --    indefinitamente
(!a1 U G a1) & (!a2 U G a2)  & -- D: quando diviene attivo, un processo rimane
                               --    tale
G(l1 -> !l2)                 & -- E: in ogni istante c'è sempre al più un
```

```
                                --    leader
G F(l1 | l2)                & -- F: prima o poi c'è almeno un leader
G((l1 & !l2 & a2) -> F ! l1) & -- G: se c'è un processo attivo con
                                --    identificatore maggiore, prima o poi il
                                --    leader dà le dimissioni
G(l1 & !X l1 -> X F l2)         -- H: un nuovo leader deve migliorare il leader
                                --    precedente
```

○

Soluzione esercizio 6.17. I valori richiesti sono i seguenti:

1. T,
2. T,
3. T,
4. T,
5. T,
6. F, controesempio: $s_1, s_3, s_3, \ldots$,
7. T,
8. T,
9. F, controesempio: $s_1, s_3, s_3, \ldots$.

○

Soluzione esercizio 6.18.

1. Vale $M, s_3 \not\models F\ \neg q$, come testimoniato dal cammino infinito $s_3, s_1, s_3, \ldots$.
2. Per $b \in \{0, 1\}$ nel bounded model cecking vale $M, s_3 \models F\ \neg q$. Per tutti gli altri valori vale $M, s_3 \not\models F\ \neg q$.

○

Soluzione esercizio 6.19. Tutte le formule sono b-tautologie per i seguenti valori di b:

1. $b = 3$,
2. $b = 1$,
3. $b = 1$,
4. $b = 2$.

○

Soluzione esercizio 6.20. Il file NuSMV richiesto è il seguente.

```
-- Time-stamp: "2006-10-05 16:49:53 cadoli"
-- File: struttTemp.smv
-- Descrizione: model checking in una struttura temporale

MODULE main
VAR
  a: boolean;
  b: boolean;
  c: boolean;
TRANS
  case
-- s1: s2 | s3
    a & b & !c: (next(a) = 0 & next(b) = 1 & next(c) = 1) |
                (next(a) = 0 & next(b) = 0 & next(c) = 1);
-- s2: s1 | s3
    !a & b & c: (next(a) = 0 & next(b) = 0 & next(c) = 1) |
                (next(a) = 0 & next(b) = 0 & next(c) = 1);
-- s3: s3
    !a & !b & c: (next(a) = 0 & next(b) = 0 & next(c) = 1);
-- non-transizioni
    1: next(a) = a & next(b) = b & next(c) = c;
  esac
DEFINE
s1 := a & b & !c;
s2 := !a & b & c;
s3 := !a & !b & c;
s  := s1 | s2 | s3;
LTLSPEC
-- senza stato iniziale
-- s -> G !(a & c)                    -- 1: T
-- s -> F(!b & c) -> F G c            -- 2: T
-- s -> ((!b & c) U c) | F G(!a & c) -- 3: T
-- con stato iniziale
-- s1 -> a & b                        -- 4: T
-- s1 -> X c                          -- 5: T
-- s1 -> b & c                        -- 6: F
-- s3 -> G c                          -- 7: T
-- s1 -> ((G F a) -> G F c)           -- 8: T
-- s1 -> ((G F c) -> G F a)           -- 9: F
```

○