

LOGICA TEMPORALE
E VERIFICA DI PROPRIETÀ DI PROGRAMMI

MARTA CIALDEA MAYER

Indice

1	Logica Temporale Lineare	3
1.1	Introduzione	3
1.2	Le logiche modali	3
1.3	Sintassi e semantica della logica temporale lineare	8
1.3.1	Sintassi	8
1.3.2	Semantica	9
1.3.3	Alcune proprietà degli operatori temporali	12
1.4	Sistemi di inferenza per la logica temporale lineare	15
1.4.1	Sistemi assiomatici per LTL	15
1.4.2	Tableaux per LTL	16
1.4.3	Terminazione	18
1.4.4	Tableaux chiusi	19
1.4.5	Cammini aperti	21
1.5	Esercizi	28
2	Verifica di Proprietà di Programmi	30
2.1	Introduzione	30
2.2	Automi su parole infinite	31
2.2.1	Automi di Büchi	32
2.2.2	Automi e formule temporali	33
2.2.3	Automi di Büchi generalizzati	35
2.2.4	Operazioni su automi di Büchi	37
2.3	Verifica di sistemi basata su model checking in LTL	40
2.3.1	Traduzione di formule LTL in GBA	42
2.3.2	Esempi	43
2.3.3	Un algoritmo per la costruzione dell'automata corrispondente a una formula	46
2.3.4	Verifica in teoria degli automi	49
2.4	Esercizi	51

Capitolo 1

Logica Temporale Lineare

1.1 Introduzione

In questo capitolo viene introdotta la logica temporale lineare (LTL), uno dei formalismi logici per la specifica di comportamenti di sistemi soggetti ad evoluzione nel tempo; tali formalismi si sono rivelati di importanza cruciale nelle applicazioni orientate alla verifica di programmi e protocolli.

Si noti che la logica classica è concepita per rappresentare realtà statiche: una interpretazione di un linguaggio classico corrisponde a uno “stato” del mondo. È naturalmente possibile fare riferimento al tempo anche in logica dei predicati (si consideri, ad esempio, il Calcolo delle Situazioni); tuttavia, come vedremo in queste dispense, l’uso di operatori logici specifici per rappresentare realtà che evolvono nel tempo consente di ottenere un notevole potere espressivo anche restando nel caso proposizionale (cioè senza l’uso di quantificatori) e, cosa ancora importante, in una logica *decidibile*.

1.2 Le logiche modali

LTL è un caso particolare di *logica modale*. Le logiche modali descrivono un insieme di stati e le relazioni tra di essi, estendendo la logica classica mediante un insieme di *operatori modali* (o *modalità*). La logica modale nasce con l’analisi delle proposizioni contenenti le espressioni “necessario” e “possibile” fatta da Aristotele. I termini “modale” e “modalità” derivano dalla tradizione della logica scolastica, secondo la quale le espressioni “necessario” e “possibile” designavano “modi d’essere” o di “presentarsi” degli enunciati ai quali si riferiscono. In questo senso si può distinguere tra proposizioni vere “necessariamente”, “possibilmente” o “contingentemente”. Queste tre modalità (chiamate *aletiche*) nella logica moderna sono rispettivamente distinte da:

- l’operatore \Box (necessità);
- l’operatore \Diamond (possibilità);
- l’assenza di operatori modali (contingenza).

I due operatori modali \Box e \Diamond sono operatori “unari”; si applicano cioè a una singola formula. In altri termini, se A è una formula, lo sono anche $\Box A$ (A è necessaria) e $\Diamond A$ (A è possibile).

La differenza fondamentale tra gli operatori modali e i connettivi della logica classica consiste nel fatto che i primi non rappresentano funzioni booleane dei propri argomenti, non sono cioè *vero-funzionali*. Ad esempio, A può essere vera (in una data interpretazione), e $\Box A$ può essere vera oppure falsa (nella stessa interpretazione). Al contrario, gli operatori della logica classica sono vero-funzionali, cioè la verità di una formula costruita a partire da formule più semplici per applicazione di un operatore classico dipende soltanto dalla verità delle formule a cui è applicato.

Ai concetti di necessità e possibilità è possibile tuttavia dare interpretazioni diverse: ad esempio, potrebbe trattarsi di necessità/possibilità logica oppure fisica. Gli stessi operatori modali possono inoltre essere utilizzati per rappresentare concetti diversi, come:

- l’obbligo (morale, sociale, normativo,...); $\Box A$ significa che A è obbligatorio, e $\Diamond A$ che A è permesso. \Box e \Diamond sono in questo caso interpretati come *modalità deontiche*.
- La conoscenza o l’opinione di un agente; $\Box A$ significa che l’agente sa (o crede) che A sia vero, e $\Diamond A$ che A è compatibile con le conoscenze (o opinioni) dell’agente. \Box e \Diamond sono in questo caso interpretati come *modalità epistemiche* (conoscenza) o *doxastiche* (opinione).
- la verità/falsità di una proposizione rispetto al tempo. $\Box A$ significa che A sarà sempre vero nel futuro (o nel passato), e $\Diamond A$ che A si verificherà in qualche istante futuro (o passato). \Box e \Diamond sono in questo caso interpretati come *modalità temporali*.

La definizione di una semantica formale per gli operatori modali, senza la quale il loro significato rimane vago, è abbastanza recente. Anche se anticipata da lavori precedenti, il suo compimento è da attribuire al filosofo e logico americano S. Kripke che, alla fine degli anni 50, definì la *semantica dei mondi possibili* per le logiche modali.¹ In questo approccio, le formule modali sono da interpretare in un insieme di “mondi possibili”; ogni mondo di tale struttura è un’interpretazione classica e ciascuna modalità corrisponde a una quantificazione (universale o esistenziale) sull’insieme dei mondi. Il significato di una formula (la sua verità/falsità) può variare da un mondo all’altro. Per cui, ad esempio, si può avere che una formula A è vera in un mondo w e falsa in un altro mondo w' . L’idea della necessità come “verità in tutti i mondi possibili” era stata già proposta precedentemente da Carnap,² ma la semplice interpretazione di $\Box A$ come “ A è vera in tutti i mondi possibili” rende indistinguibili formule della forma $\Box A$, $\Box\Box A$, $\Box\Box\Box A$,... (in generale, l’iterazione di un prefisso modale non ha alcun effetto). Inoltre, una formula della forma $\Box A$ non potrebbe essere vera in un mondo e falsa in un altro, perché la verità di $\Box A$ in un mondo w non dipende in alcun modo da w stesso.

¹S. Kripke, ‘A Completeness Theorem in Modal Logic’, *Journal of Symbolic Logic* **24** (1959): 1–14; ‘Semantical Considerations on Modal Logic’, *Acta Philosophica Fennica* **16** (1963): 83–94.

²R. Carnap, *Introduction to Semantics*, Cambridge, MA, 1942; *Meaning and Necessity*, University of Chicago Press, 1947.

Kripke introduce invece una *relazione di accessibilità* (binaria) sui mondi possibili, e definisce:

$\Box A$ è vero in un mondo w se e solo se A è vero in ogni mondo w' accessibile da w .

I mondi “accessibili” da un mondo w rappresentano stati di cose “possibili” dal punto di vista di w . Di conseguenza, $\Box A$ è vera in w se e solo se A è vera in tutti i mondi che sono *possibili secondo* w . E se qualche formula A è vera in un mondo w' e tuttavia $\Diamond A$ non vale in w , questo è perché w' rappresenta uno stato di cose che non è possibile dal punto di vista di w . In altri termini, si può dire che i mondi accessibili da w sono per lui “visibili”, mentre gli altri non lo sono. Un’interpretazione di un linguaggio modale è dunque un grafo orientato, dove i nodi sono i mondi possibili e gli archi rappresentano la relazione di accessibilità.

Si consideri ad esempio la struttura rappresentata nella figura 1.2. Qui ogni nodo rappresenta un mondo ed è etichettato dal suo nome (w, w_1, \dots) – utile per riferirci ad esso nel seguito – e dall’insieme di variabili proposizionali vere nel mondo stesso. Come si vede, a ogni mondo è associata un’interpretazione proposizionale. Nel mondo w , ad esempio, sono veri gli atomi p e q , mentre r è falso. Tuttavia $\Diamond r$ è vero in w , perché r è vero in un mondo accessibile da w (w_2). In w è anche vero $\Box p$, perché p è vero in ogni mondo accessibile da w (w stesso, w_1, w_2 e w_3); il fatto che sia falso in w_4 non influisce sulla verità di $\Box p$ in w perché w non “vede” w_4 . $\Box p$ è invece falso in w_3 (che vede w_4). Dato poi che p è falso in w_4 e w vede un mondo che vede a sua volta w_4 , in w è vero $\Diamond \Diamond \neg p$. Consideriamo ora il mondo w_2 , che non vede alcun mondo. In w_2 tutto è necessario, perché qualsiasi formula è vera in tutti i mondi accessibili da w_2 . Quindi in w è vero ad esempio $\Diamond \Box p$.

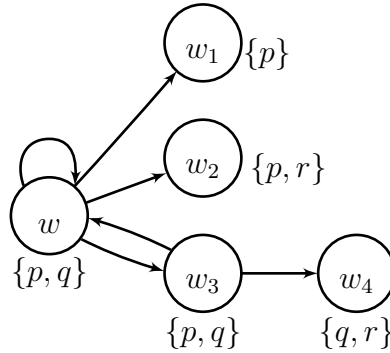


Figura 1.1: Una struttura di Kripke

Formalmente, secondo la semantica di Kripke:

Definizione 1 Un’interpretazione modale \mathcal{M} del linguaggio proposizionale basato sulle variabili dell’insieme P è una quadrupla $\langle W, w_0, R, v \rangle$, dove:

1. W è un insieme non vuoto (l’insieme dei mondi possibili);
2. w_0 è un elemento di W (il “mondo iniziale”);

3. R è una relazione binaria su W : $R \subseteq W \times W$ (la relazione di accessibilità; scriveremo wRw' per indicare che w' è accessibile da w);
4. v è una funzione $v : W \rightarrow 2^P$ che associa a ogni elemento $w \in W$ un sottoinsieme di P .

Per definire la nozione di verità di una formula in un'interpretazione, occorre definire prima la nozione di verità di una formula A in un mondo w di un'interpretazione:

$$\mathcal{M}_w \models A$$

Definizione 2 Sia $\mathcal{M} = \langle W, w_0, R, v \rangle$ un'interpretazione modale. La relazione $\mathcal{M}_w \models A$ è definita induttivamente come segue:

1. se p è una variabile proposizionale, allora $\mathcal{M}_w \models p$ se e solo se $p \in v(w)$;
2. per ogni \mathcal{M} e w , $\mathcal{M}_w \models \top$ e $\mathcal{M}_w \not\models \perp$;
3. $\mathcal{M}_w \models \neg A$ sse $\mathcal{M}_w \not\models A$;
4. $\mathcal{M}_w \models A \wedge B$ sse $\mathcal{M}_w \models A$ e $\mathcal{M}_w \models B$;
5. $\mathcal{M}_w \models A \vee B$ sse $\mathcal{M}_w \models A$ oppure $\mathcal{M}_w \models B$;
6. $\mathcal{M}_w \models A \rightarrow B$ sse $\mathcal{M}_w \not\models A$ oppure $\mathcal{M}_w \models B$;
7. $\mathcal{M}_w \models \Box A$ sse per ogni $w' \in W$, se wRw' allora $\mathcal{M}_{w'} \models A$;
8. $\mathcal{M}_w \models \Diamond A$ sse esiste $w' \in W$ tale che wRw' e $\mathcal{M}_{w'} \models A$.

Una formula A è vera in $\mathcal{M} = \langle W, w_0, R, v \rangle$:

$$\mathcal{M} \models A$$

sse $\mathcal{M}_{w_0} \models A$, cioè se è vera nel "mondo iniziale". Una formula è valida sse è vera in tutte le sue interpretazioni.

Come si vede, l'operatore di necessità ha un significato universale (corrisponde a una quantificazione universale sui mondi accessibili), mentre quello di possibilità ha un significato esistenziale. A differenza dei due quantificatori classici, però, il dominio di quantificazione (i mondi accessibili da un dato mondo) non è necessariamente non vuoto. Di conseguenza, mentre la formula $\forall x A \rightarrow \exists x A$ è valida in logica classica, la sua controparte modale, $\Box A \rightarrow \Diamond A$, non lo è.

La semantica dei mondi possibili, come definita sopra, costituisce il nucleo di diverse logiche modali. Le formule valide secondo tale semantica sono dunque valide in tutte le logiche modali che si basano su di essa. Ad esempio, le formule seguenti sono logicamente valide:

$$\begin{aligned} \Diamond A &\equiv \neg \Box \neg A \\ \Box A &\equiv \neg \Diamond \neg A \\ \Box(A \wedge B) &\rightarrow \Box A \wedge \Box B \\ \Diamond(A \vee B) &\equiv \Diamond A \vee \Diamond B \\ \Box(A \rightarrow B) &\rightarrow (\Box A \rightarrow \Box B) \\ \Box A &\text{ se } A \text{ è classicamente valida} \end{aligned}$$

Altre formule, come quelle seguenti, non sono valide:

- 1) $\Box A \rightarrow \Diamond A$
- 2) $\Box A \rightarrow A$
- 3) $\Box A \rightarrow \Box \Box A$
- 4) $\Box A \rightarrow \Box \Diamond A$

Eppure, a seconda dell'interpretazione che si vuole dare agli operatori modali, si vorrebbe avere la validità di alcune delle formule precedenti. Ad esempio, supponiamo di voler dare al \Box un'interpretazione *epistemica* ($\Box A$: l'agente conosce la verità di A ; $\Diamond A$: A è compatibile con le conoscenze dell'agente). In questa interpretazione, l'insieme dei mondi accessibili da un mondo dato rappresenta l'insieme degli stati di cose compatibili con le conoscenze dell'agente. Le quattro formule sopra riportate si interpretano allora, rispettivamente, come:

- 1) Se l'agente conosce A , allora A è consistente con le sue conoscenze;
- 2) Quel che l'agente conosce è vero (la conoscenza è opinione vera);
- 3) Se l'agente conosce A allora sa di conoscerlo;
- 4) Se l'agente non conosce A allora sa di non conoscerlo.

(le proprietà 3 e 4 caratterizzano la capacità introspettiva dell'agente). Secondo l'interpretazione epistemica le quattro formule dovrebbero essere tutte valide.

La semantica di molte logiche modali (chiamate *logiche modali normali*), tra cui quella epistemica, si ottiene dal nucleo della definizione della semantica dei mondi possibili (definizioni 1 e 2) imponendo opportune restrizioni sulla relazione di accessibilità.

In altri termini, vi sono formule che pur non essendo valide secondo la semantica kripkeana generale, sono vere in tutte le interpretazioni appartenenti ad una determinata classe. Molte di tali classi si possono caratterizzare mediante condizioni interessanti che la relazione di accessibilità deve soddisfare.

Ad esempio si ha che:

1. la formula 1) è vera in tutte le interpretazioni in cui la relazione R è *seriale* (per ogni $w \in W$ esiste $w' \in W$ tale che wRw');
2. la formula 2) è vera in tutte le interpretazioni in cui la relazione R è *riflessiva*;
3. la formula 3) è vera in tutte le interpretazioni in cui la relazione R è *transitiva*;
4. la formula 4) è vera in tutte le interpretazioni in cui la relazione R è *euclidea* (per ogni $w, w', w'' \in W$, se wRw' e wRw'' , allora $w'Rw''$).

Inoltre, le quattro formule sono tutte vere nelle interpretazioni in cui R è una relazione di equivalenza.

1.3 Sintassi e semantica della logica temporale lineare

La logica temporale lineare (LTL) è una logica modale le cui interpretazioni sono strutture lineari e discrete di stati. In queste dispense ci limitiamo a considerare strutture temporali infinite, e la logica temporale con i soli operatori “del futuro” (omettiamo cioè di considerare gli operatori che consentono di riferirsi a istanti di tempo passati).

In LTL, agli operatori \Box e \Diamond viene data un'interpretazione temporale:

$\Box A$: A sarà sempre vera in futuro;

$\Diamond A$: A sarà vera in qualche istante futuro.

Secondo l'interpretazione temporale, ad esempio, la formula $\Box\Diamond A$ si interpreta come “ A si verifica infinite volte in futuro”, e $\Diamond\Box A$ come “da un certo istante futuro in poi, A sarà sempre vera”.

Sono inoltre presenti altri operatori “temporali”: \bigcirc (“next”) e \mathcal{U} (“until”), dove:

$\bigcirc A$: A è vero nello stato successivo a quello attuale.

$A\mathcal{U}B$: B sarà vero in qualche istante futuro e A vale da adesso fino al momento in cui sarà vero B (escludendo tale stato).

Come si vede, l'operatore \mathcal{U} è *binario*, cioè si applica a due formule (si comporta, sintatticamente, come i connettivi \wedge , \vee , ...).

Inoltre, per consentire la possibilità di trasformare qualsiasi formula in *forma normale negativa* (dove le negazioni dominano solo atomi), viene incluso anche l'operatore binario \mathcal{R} (“release”), dove:

$A\mathcal{R}B$: o B sarà sempre vero, oppure sarà vero fino a che il verificarsi di A non “libera” B dal vincolo di essere vero (ma B deve essere ancora vero quando vale A).

L'operatore \mathcal{R} è infatti il *duale* di \mathcal{U} , nel senso che la seguente equivalenza è valida:

$$(A\mathcal{R}B) \equiv \neg((\neg A)\mathcal{U}(\neg B))$$

1.3.1 Sintassi

L'insieme delle formule di LTL su un insieme di variabili proposizionali P è definito come segue:

Definizione 3

1. \top e \perp sono formule;
2. se $p \in P$ allora p è una formula;
3. se A è una formula allora $(\neg A)$, $(\Box A)$, $(\Diamond A)$, $(\bigcirc A)$ sono formule;

4. se A e B sono formule, allora $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \mathcal{U} B)$, $(A \mathcal{R} B)$ sono formule.

Le parentesi più esterne sono di regola omesse e si conviene che gli operatori unari abbiano precedenza maggiore di quelli binari.

1.3.2 Semantica

In queste dispense rappresenteremo le interpretazioni proposizionali classiche mediante insiemi di atomi, convenendo che un atomo p è vero in un'interpretazione I se e solo se $p \in I$.

Un'interpretazione temporale non è altro che una sequenza infinita numerabile di interpretazioni classiche. Formalmente, se \mathbb{N} è l'insieme dei numeri naturali, ordinati secondo l'abituale relazione $<$, allora:

Definizione 4 Un'interpretazione temporale \mathcal{M} del linguaggio P è una funzione che associa a ogni $k \in \mathbb{N}$ un sottoinsieme di P .³

$$\mathcal{M} : \mathbb{N} \rightarrow 2^P$$

Ogni $k \in \mathbb{N}$ rappresenta un istante di tempo (o *stato*) e il sottoinsieme di P ad esso associato l'insieme delle lettere proposizionali vere nello stato k . 0 è lo stato iniziale.

La verità di una formula A in uno stato k di un'interpretazione \mathcal{M} ($\mathcal{M}_k \models A$) è definita come segue:

Definizione 5 La relazione $\mathcal{M}_k \models A$ è definita induttivamente come segue

1. $\mathcal{M}_k \models \top$ e $\mathcal{M}_k \not\models \perp$;
2. $\mathcal{M}_k \models p$ sse $p \in \mathcal{M}(k)$;
3. $\mathcal{M}_k \models \neg A$ sse $\mathcal{M}_k \not\models A$;
4. $\mathcal{M}_k \models A \wedge B$ sse $\mathcal{M}_k \models A$ e $\mathcal{M}_k \models B$;
5. $\mathcal{M}_k \models A \vee B$ sse $\mathcal{M}_k \models A$ oppure $\mathcal{M}_k \models B$;
6. $\mathcal{M}_k \models A \rightarrow B$ sse $\mathcal{M}_k \not\models A$ oppure $\mathcal{M}_k \models B$;
7. $\mathcal{M}_k \models \bigcirc A$ sse $\mathcal{M}_{k+1} \models A$;
8. $\mathcal{M}_k \models \square A$ sse per ogni $i \geq k$ $\mathcal{M}_i \models A$;
9. $\mathcal{M}_k \models \diamond A$ sse esiste $i \geq k$ tale che $\mathcal{M}_i \models A$;
10. $\mathcal{M}_k \models A \mathcal{U} B$ sse esiste $i \geq k$ tale che $\mathcal{M}_i \models B$ e per ogni $j = k, \dots, i-1$ ($k \leq j < i$), $\mathcal{M}_j \models A$;
11. $\mathcal{M}_k \models A \mathcal{R} B$ sse vale uno di questi due casi:
 - per ogni $i \geq k$, $\mathcal{M}_i \models B$;

³Se S è un qualsiasi insieme, usiamo la notazione 2^S per indicare l'insieme delle parti di S .

- esiste $i \geq k$ tale che $\mathcal{M}_i \models A$ e per ogni $j = k, \dots, i$ ($k \leq j \leq i$) si ha $\mathcal{M}_j \models B$;

Una formula A è vera in un'interpretazione \mathcal{M} sse è vera nel suo stato iniziale:

$$\mathcal{M} \models A \text{ sse } \mathcal{M}_0 \models A$$

Se S è un insieme di formule e \mathcal{M} un'interpretazione:

- $\mathcal{M}_k \models S$ sse $\mathcal{M}_k \models A$ per ogni formula $A \in S$;
- \mathcal{M} è un modello di S ($\mathcal{M} \models S$) sse $\mathcal{M} \models A$ per ogni formula $A \in S$.

Una formula A è valida ($\models A$) sse per ogni interpretazione \mathcal{M} e ogni $k \in \mathbb{N}$, $\mathcal{M}_k \models A$.

Se S è un insieme di formule e A una formula, A è una conseguenza logica di S ($S \models A$) sse per ogni interpretazione \mathcal{M} e $k \in \mathbb{N}$: se $\mathcal{M}_k \models S$ allora $\mathcal{M}_k \models A$. Se A è una conseguenza logica di S diciamo anche che S implica logicamente A .

Due formule A e B sono logicamente equivalenti ($A \leftrightarrow B$) se per ogni interpretazione \mathcal{M} e stato $k \in \mathbb{N}$, $\mathcal{M}_k \models A$ se e solo se $\mathcal{M}_k \models B$.

Dalle prime 6 clausole della definizione precedente, segue immediatamente che se A è una formula proposizionale classica, allora, per ogni interpretazione \mathcal{M} e $k \in \mathbb{N}$, $\mathcal{M}_k \models A$ se e solo se A è vera nell'interpretazione classica rappresentata da $\mathcal{M}(k)$.

La definizione del significato dell'operatore \circ (clausola 7) formalizza l'idea intuitiva che $\circ A$ vale in un istante k sse A vale nell'istante immediatamente successivo.

Per quel che riguarda le clausole che definiscono il significato degli altri operatori temporali, osserviamo innanzitutto (clausole 8 e 9) che *il futuro include lo stato attuale*: infatti se $\Box A$ è vera nello stato k , allora A è vera in k ; e se A è vera in k , allora $\Diamond A$ è vera in k .

Per quel che riguarda l'operatore \mathcal{U} (clausola 10): $A\mathcal{U}B$ è vera nello stato k sse esiste uno stato $i \geq k$ in cui è vero B , ed A vale da k (incluso) fino a i (escluso). Dato che lo stato i potrebbe essere uguale a k stesso, ed in tal caso l'intervallo $[k, \dots, i - 1]$ è vuoto, ne segue che se B vale in k , allora $A\mathcal{U}B$ è vera in k , per qualsiasi formula A .

La definizione dell'operatore \mathcal{R} (clausola 11) si può riformulare come segue. $A\mathcal{R}B$ vale nello stato k se vale uno dei due casi seguenti:

- B vale da k (incluso) in poi;
- esiste uno stato $i \geq k$ in cui A è vera e B vale in ogni stato compreso tra k e i (estremi inclusi); cioè B vale da k incluso fino a un istante in cui valgono sia A che B .

Nel secondo caso, il verificarsi di A "libera" B dalla necessità di continuare a verificarsi in seguito. Si noti che, anche in questo caso, poiché potrebbe essere $i = k$, se in k valgono sia A che B , allora in k vale $A\mathcal{R}B$.

Dalla definizione 5 seguono le seguenti proprietà degli operatori temporali (falsità di una formula in uno stato):

1. $\mathcal{M}_k \not\models \bigcirc A$ sse $\mathcal{M}_{k+1} \not\models A$
2. $\mathcal{M}_k \not\models \Box A$ sse esiste $i \geq k$ tale che $\mathcal{M}_i \not\models A$
3. $\mathcal{M}_k \not\models \Diamond A$ sse per ogni $i \geq k$, $\mathcal{M}_i \not\models A$
4. $\mathcal{M}_k \not\models AU B$ sse per ogni $i \geq k$, se $\mathcal{M}_i \models B$, allora esiste j tale che $k \leq j < i$ e $\mathcal{M}_j \not\models A$.⁴
5. $\mathcal{M}_k \not\models A\mathcal{R}B$ sse valgono entrambe le condizioni seguenti:
 - esiste $i \geq k$ tale che $\mathcal{M}_i \not\models B$, e
 - per ogni $i \geq k$, se $\mathcal{M}_i \models A$, allora esiste j tale che $k \leq j \leq i$ e $\mathcal{M}_j \not\models B$.

A conclusione di questo paragrafo, osserviamo che la validità è stata definita come verità *in ogni stato di ogni interpretazione*, e non come verità in ogni interpretazione (che equivale a dire verità nello stato iniziale di ogni interpretazione). Tuttavia, il risultato che segue stabilisce che in realtà le due nozioni sono equivalenti.⁵

Teorema 1 *Una formula è valida se e solo se essa è vera in ogni interpretazione.*

Dimostrazione. L'implicazione da sinistra a destra è ovvia: se A è vera in ogni stato di ogni interpretazione, allora è vera nello stato iniziale di ogni interpretazione.

Per dimostrare l'inverso, supponiamo che la formula A sia vera nello stato iniziale di ogni interpretazione ed esista tuttavia uno stato k di un'interpretazione \mathcal{M} tale che $\mathcal{M}_k \not\models A$. Consideriamo l'interpretazione \mathcal{M}' tale che per ogni $n \in \mathbb{N}$:

$$\mathcal{M}'(n) = \mathcal{M}(k + n)$$

In altri termini, se vediamo le interpretazioni temporali come sequenze di interpretazioni classiche, \mathcal{M}' è il "suffisso" di \mathcal{M} che inizia allo stato k .

Si può dimostrare facilmente (per induzione sulle formule) che, per ogni formula B e $n \in \mathbb{N}$: $\mathcal{M}'_n \models B$ se e solo se $\mathcal{M}_{n+k} \models B$. Questo vale in particolare per $n = 0$ e $B = A$. Quindi si avrebbe $\mathcal{M}'_0 \not\models A$, contraddicendo il fatto che A è vera in ogni interpretazione.

Come conseguenza, abbiamo che due formule sono logicamente equivalenti sse per ogni interpretazione \mathcal{M} esse sono entrambe vere o entrambe false nello stato iniziale di \mathcal{M} .

⁴Equivalentemente, si può dire che $\mathcal{M}_k \not\models AU B$ sse vale uno di questi due casi:

- per ogni $i \geq k$, $\mathcal{M}_i \not\models B$, oppure
- esiste $i \geq k$ tale che $\mathcal{M}_i \models B$; ma, se i_0 è il minimo $i \geq k$ tale che $\mathcal{M}_i \models B$, cioè:
$$i_0 \geq k, \mathcal{M}_{i_0} \models B$$
 e per ogni j compreso tra k e $i_0 - 1$ (inclusi), $\mathcal{M}_j \not\models B$, allora esiste j tale che $k \leq j < i_0$ e $\mathcal{M}_j \not\models A$.

⁵Questo vale per le logiche modali in generale. Per questo motivo la definizione 2 dava la definizione diretta di validità come verità in tutte le interpretazioni.

1.3.3 Alcune proprietà degli operatori temporali

Dalla semantica della logica temporale seguono alcune importanti equivalenze logiche:

1. $\diamond A \leftrightarrow \top \mathcal{U} A$ (quindi \diamond è definibile in termini di \mathcal{U});

2. \diamond e \square sono duali:

$$\begin{aligned}\square A &\leftrightarrow \neg \diamond \neg A \\ \diamond A &\leftrightarrow \neg \square \neg A\end{aligned}$$

3. \mathcal{U} e \mathcal{R} sono duali:

$$\begin{aligned}A \mathcal{R} B &\leftrightarrow \neg(\neg A \mathcal{U} \neg B) \\ A \mathcal{U} B &\leftrightarrow \neg(\neg A \mathcal{R} \neg B)\end{aligned}$$

4. $\bigcirc A \leftrightarrow \neg \bigcirc \neg A$ (quindi \bigcirc è duale di se stesso)

5. $\square A \leftrightarrow A \wedge \bigcirc \square A$

6. $\diamond A \leftrightarrow A \vee \bigcirc \diamond A$

7. $A \mathcal{U} B \leftrightarrow B \vee (A \wedge \bigcirc(A \mathcal{U} B))$

8. $A \mathcal{R} B \leftrightarrow B \wedge (A \vee \bigcirc(A \mathcal{R} B))$

9. $A \mathcal{R} B \leftrightarrow \square B \vee (B \mathcal{U}(A \wedge B))$

Le equivalenze 5–8 sono molto importanti: costituiscono una sorta di “equazioni di punto fisso” per gli operatori \square , \diamond , \mathcal{U} , \mathcal{R} .

Inoltre, le formule seguenti sono valide:

1. $\square A \rightarrow A$

2. $A \rightarrow \diamond A$

3. $\square A \rightarrow \square \square A$

4. $\square A \rightarrow \bigcirc A$

5. $\bigcirc A \rightarrow \diamond A$

6. $\bigcirc(A \rightarrow B) \rightarrow (\bigcirc A \rightarrow \bigcirc B)$

7. $\square(A \rightarrow B) \rightarrow (\square A \rightarrow \square B)$

8. $A \wedge \square(A \rightarrow \bigcirc A) \rightarrow \square A$

9. $A \mathcal{U} B \rightarrow \diamond B$

10. $A \mathcal{R} B \rightarrow B$

Le prime due formule sono valide perché \leq è riflessiva; la 3 perché \leq è transitiva. La validità delle formule 4 e 5 (relazione tra \circ e \square o \diamond) è dovuta al fatto che $k + 1 \geq k$. La 6 e la 7 sono il corrispondente modale delle analoghe proprietà del quantificatore universale. La 8 è la riscrittura, in LTL, del principio di induzione matematica. L'ultima formula infine stabilisce una relazione tra \mathcal{U} e \diamond .

Dimostriamo ora, come esempio, alcune delle proprietà sopra riportate.

Le formule $\diamond A$ e $A \vee \circ \diamond A$ sono logicamente equivalenti.

Per dimostrare che $\diamond A \models A \vee \circ \diamond A$, assumiamo che \mathcal{M} sia qualsiasi interpretazione e $n \in \mathbb{N}$ tali che $\mathcal{M}_n \models \diamond A$. Quindi esiste $k \geq n$ tale che $\mathcal{M}_k \models A$. Se $k = n$ allora $\mathcal{M}_n \models A$, quindi $\mathcal{M}_n \models A \vee \circ \diamond A$. Altrimenti, se $k > n$, allora $k \geq n + 1$, quindi $\mathcal{M}_{n+1} \models \diamond A$ e $\mathcal{M}_n \models \circ \diamond A$.

Per dimostrare che $A \vee \circ \diamond A \models \diamond A$, assumiamo che \mathcal{M} sia qualsiasi interpretazione e $n \in \mathbb{N}$ tali che $\mathcal{M}_n \models A \vee \circ \diamond A$. Se $\mathcal{M}_n \models A$, allora, poiché $n \geq n$, si ha che $\mathcal{M}_n \models \diamond A$. Se invece $\mathcal{M}_n \models \circ \diamond A$, allora $\mathcal{M}_{n+1} \models \diamond A$, quindi esiste $k \geq n + 1$ tale che $\mathcal{M}_k \models A$; se $k \geq n + 1$ allora $k \geq n$, quindi anche $\mathcal{M}_n \models \diamond A$. In entrambi i casi dunque $\mathcal{M}_n \models \diamond A$.

La formula $A \wedge \square(A \rightarrow \circ A) \rightarrow \square A$ è valida.

Per dimostrare la validità della formula data, mostriamo che per ogni interpretazione \mathcal{M} e $n \in \mathbb{N}$, se $\mathcal{M}_n \models A \wedge \square(A \rightarrow \circ A)$ allora $\mathcal{M}_n \models \square A$. Supponiamo dunque che \mathcal{M} sia qualsiasi interpretazione e $n \in \mathbb{N}$ uno stato tali che:

1. $\mathcal{M}_n \models A$,
2. $\mathcal{M}_n \models \square(A \rightarrow \circ A)$: quindi, per ogni $k \geq n$, se $\mathcal{M}_k \models A$, allora $\mathcal{M}_{k+1} \models A$.

Possiamo riscrivere le due proprietà come segue:

1. $\mathcal{M}_{n+0} \models A$,
2. per ogni $k \in \mathbb{N}$, se $\mathcal{M}_{n+k} \models A$, allora $\mathcal{M}_{n+k+1} \models A$.

Per il principio di induzione matematica, per ogni $k \in \mathbb{N}$, si ha che $\mathcal{M}_{n+k} \models A$, cioè per ogni $k \geq n$, $\mathcal{M}_k \models A$. Dunque $\mathcal{M}_n \models \square A$.

Le formule $A \mathcal{R} B$ e $\square B \vee (BU(A \wedge B))$ sono logicamente equivalenti.

Per dimostrare che $A \mathcal{R} B \models \square B \vee (BU(A \wedge B))$, assumiamo che \mathcal{M} sia qualsiasi interpretazione e $n \in \mathbb{N}$ tali che $\mathcal{M}_n \models A \mathcal{R} B$ e $\mathcal{M}_n \not\models \square B$, e mostriamo che $\mathcal{M}_n \models BU(A \wedge B)$. Sia k_0 il più piccolo $k \geq n$ tale che $\mathcal{M}_{k_0} \not\models B$ (che esiste dato che $\mathcal{M}_n \not\models \square B$). Poiché $\mathcal{M}_n \models A \mathcal{R} B$, $\mathcal{M}_n \models B$ (vedi punto 10 a pagina 12) e quindi deve essere $k_0 > n$. Poiché k_0 è stato scelto minimo, sia ha $\mathcal{M}_n \models B$, $\mathcal{M}_{n+1} \models B, \dots, \mathcal{M}_{k_0-1} \models B$. Ma poiché $\mathcal{M}_0 \models A \mathcal{R} B$, deve esistere k_1 , $n \leq k_1 \leq k_0 - 1$ tale che $\mathcal{M}_{k_1} \models A$. Si ha allora: $\mathcal{M}_n \models B, \dots, \mathcal{M}_{k_1-1} \models B, \mathcal{M}_{k_1} \models A \wedge B$. Dunque $\mathcal{M}_n \models BU(A \wedge B)$.

Per dimostrare l'inverso assumiamo che \mathcal{M} sia qualsiasi interpretazione e $n \in \mathbb{N}$ tali che $\mathcal{M}_n \models \Box B \vee (BU(A \wedge B))$. Se $\mathcal{M}_n \models \Box B$, allora per ogni $k \geq n$ $\mathcal{M}_k \models B$, dunque $\mathcal{M}_n \models A \mathcal{R} B$. Altrimenti deve essere $\mathcal{M}_n \models BU(A \wedge B)$, quindi esiste $k_0 \geq n$ tale che $\mathcal{M}_{k_0} \models A \wedge B$ e per ogni i , se $n \leq i < k_0$, $\mathcal{M}_i \models B$: $\mathcal{M}_n \models B, \dots, \mathcal{M}_{k_0-1} \models B, \mathcal{M}_{k_0} \models A \wedge B$. Cioè esiste $k_0 \geq n$ tale che $\mathcal{M}_{k_0} \models A$ e per ogni i , se $n \leq i \leq k_0$, $\mathcal{M}_i \models B$. Dunque $\mathcal{M}_n \models A \mathcal{R} B$.

Le formule $A \mathcal{R} B$ e $\neg(\neg AU \neg B)$ sono logicamente equivalenti.

Per dimostrare che $A \mathcal{R} B \models \neg(\neg AU \neg B)$, assumiamo che \mathcal{M} sia un'interpretazione e $k \in \mathbb{N}$ tali che $\mathcal{M}_k \models A \mathcal{R} B$. Allora vale uno dei due casi seguenti, entrambi i quali implicano che $\mathcal{M}_k \models \neg(\neg AU \neg B)$:

- 1) per ogni $i \geq k$, $\mathcal{M}_i \models B$. Quindi non esiste $i \geq k$ tale che $\mathcal{M}_i \models \neg B$, dunque $\mathcal{M}_k \not\models \neg AU \neg B$, cioè $\mathcal{M}_k \models \neg(\neg AU \neg B)$.
- 2) esiste $i_0 \geq k$ tale che:
 1. $\mathcal{M}_{i_0} \models A$ e
 2. per ogni $j = k, \dots, i_0$ si ha $\mathcal{M}_j \models B$.

Supponiamo ora, per assurdo, che $\mathcal{M}_k \not\models \neg(\neg AU \neg B)$, cioè $\mathcal{M}_k \models \neg AU \neg B$: esiste allora $i_1 \geq k$ tale che:

3. $\mathcal{M}_{i_1} \models \neg B$, e
4. per ogni $j = k, \dots, i_1 - 1$, $\mathcal{M}_j \models \neg A$.

Si dovrebbe allora avere $i_1 > i_0$, per (2) e (3). Dato che $k \geq i_0 < i_1$, si deve avere (per 4) che $\mathcal{M}_{i_0} \models \neg A$, contraddicendo (1).

Per dimostrare l'inverso, cioè che $\neg(\neg AU \neg B) \models A \mathcal{R} B$, ragioniamo per contrapposizione, mostrando che per qualsiasi \mathcal{M} e k , se $\mathcal{M}_k \not\models A \mathcal{R} B$, allora $\mathcal{M}_k \not\models \neg(\neg AU \neg B)$ (cioè $\mathcal{M}_k \models \neg AU \neg B$). Assumiamo dunque che $\mathcal{M}_k \not\models A \mathcal{R} B$, cioè che valgano entrambe le condizioni seguenti (vedi pagina 11):

- (a) esiste $i \geq k$ tale che $\mathcal{M}_i \not\models B$, e
- (b) per ogni $i \geq k$, se $\mathcal{M}_i \models A$, allora esiste j tale che $k \leq j \leq i$ e $\mathcal{M}_j \not\models B$.

Sia ora i_0 il più piccolo $i \geq k$ che soddisfa la condizione (a), cioè:

- (b) $i_0 \geq k$ e $\mathcal{M}_{i_0} \models \neg B$, e
- (c) per ogni i , se $k \leq i < i_0$, $\mathcal{M}_i \models B$.

Dimostriamo allora che per ogni j , se $k \leq j < i_0$, allora $\mathcal{M}_j \models \neg A$. Supponiamo per assurdo che esista j_0 , tale che $k \leq j_0 < i_0$ e $\mathcal{M}_{j_0} \not\models \neg A$, cioè $\mathcal{M}_{j_0} \models A$. Per la (b), esiste j_1 tale che $k \leq j_1 \leq j_0$ e $\mathcal{M}_{j_1} \not\models B$. Poiché $k \leq j_1 \leq j_0$ e $j_0 < i_0$, ne segue $k \leq j_1 \leq i_0$. Ma questo, assieme al fatto che $\mathcal{M}_{j_1} \not\models B$, contraddice (c). Dunque:

- (d) per ogni j , se $k \leq j < i_0$ e $\mathcal{M}_j \models \neg A$.

Questo, assieme alla (b), porta a concludere che $\mathcal{M}_k \models \neg A \mathcal{U} \neg B$.

Il prossimo esempio dimostra la tecnica da utilizzare quando si vuole mostrare che una formula A non è valida: si deve definire un'interpretazione \mathcal{M} tale che $\mathcal{M}_0 \not\models A$.

La formula $\diamond A \rightarrow \square A$ non è valida.

Si consideri qualsiasi interpretazione \mathcal{M} in cui $\mathcal{M}_0 \models A$ e $\mathcal{M}_1 \not\models A$. In tale interpretazione, ovviamente, $\mathcal{M}_0 \models \diamond A$, ma $\mathcal{M}_0 \not\models \square A$.

A conclusione di questo paragrafo osserviamo che qualsiasi formula è logicamente equivalente a una formula in forma normale negativa (nnf) – cioè in cui le negazioni dominano soltanto formule atomiche. Per trovare la nnf di una formula basta applicare le seguenti regole di riscrittura (oltre a quelle analoghe per la logica classica):⁶

$$\begin{aligned} \neg \bigcirc A &\Rightarrow \bigcirc \neg A \\ \neg \diamond A &\Rightarrow \square \neg A \\ \neg \square A &\Rightarrow \diamond \neg A \\ \neg(A \mathcal{U} B) &\Rightarrow \neg A \mathcal{R} \neg B \\ \neg(A \mathcal{R} B) &\Rightarrow \neg A \mathcal{U} \neg B \end{aligned}$$

Le formule sinistra e destra di ciascuna regola sono logicamente equivalenti.

1.4 Sistemi di inferenza per la logica temporale lineare

1.4.1 Sistemi assiomatici per LTL

Nelle trattazioni assiomatiche di LTL si considerano primitivi soltanto gli operatori temporali $\bigcirc, \square, \diamond, \mathcal{U}$ (gli altri, come si è visto, si possono definire in termini di questi).

Un sistema assiomatico per LTL si può ottenere aggiungendo a qualsiasi sistema assiomatico per la logica proposizionale classica (ad esempio il sistema di inferenza hilbertiano):

(a) i seguenti (schemi di) assiomi:

$$\mathbf{A1.} \quad \neg \diamond A \equiv \square \neg A$$

$$\mathbf{A2.} \quad \square(A \rightarrow B) \rightarrow (\square A \rightarrow \square B)$$

⁶Ricordiamo che per trasformare una formula proposizionale in nnf si eliminano innanzitutto le implicazioni e le doppie implicazioni applicando le seguenti regole di riscrittura:

$$\begin{aligned} A \rightarrow B &\Rightarrow \neg A \vee B \\ \neg(A \rightarrow B) &\Rightarrow A \wedge \neg B \\ A \equiv B &\Rightarrow (A \wedge B) \vee (\neg A \wedge \neg B) \\ \neg(A \equiv B) &\Rightarrow (A \wedge \neg B) \vee (\neg A \wedge B) \end{aligned}$$

In seguito si applicano le regole:

$$\begin{aligned} \neg(A \wedge B) &\Rightarrow \neg A \vee \neg B \\ \neg(A \vee B) &\Rightarrow \neg A \wedge \neg B \\ \neg \neg A &\Rightarrow A \end{aligned}$$

- A3.** $\Box A \rightarrow (A \wedge \bigcirc \Box A)$
A4. $\bigcirc \neg A \equiv \neg \bigcirc A$
A5. $\bigcirc(A \rightarrow B) \rightarrow (\bigcirc A \rightarrow \bigcirc B)$
A6. $\Box(A \rightarrow \bigcirc A) \rightarrow (A \rightarrow \Box A)$
A7. $A \mathcal{U} B \equiv B \vee (A \wedge \bigcirc(A \mathcal{U} B))$
A8. $A \mathcal{U} B \rightarrow \diamond B$

(b) La regola di necessitazione:

$$\frac{A}{\Box A}$$

Tale sistema è corretto e completo per LTL.

1.4.2 Tableaux per LTL

Il metodo dei tableaux semantici è stato esteso a LTL da Wolper.⁷ La chiave del metodo è la natura ricorsiva delle equivalenze già viste:

$$\begin{aligned} \Box A &\leftrightarrow A \wedge \bigcirc \Box A \\ \diamond A &\leftrightarrow A \vee \bigcirc \diamond A \\ A \mathcal{U} B &\leftrightarrow B \vee (A \wedge \bigcirc(A \mathcal{U} B)) \\ A \mathcal{R} B &\leftrightarrow (A \wedge B) \vee (B \wedge \bigcirc(A \mathcal{R} B)) \end{aligned}$$

In un tableau per LTL i nodi non sono etichettati da singole formule, ma da insiemi di formule. Essenzialmente, un nodo rappresenta quel che è vero in uno stato; questo viene “analizzato” fin dove possibile, cioè finché esso non contiene soltanto letterali e formule prefisse dall’operatore \bigcirc : a questo punto si sa tutto quel che deve essere vero nello stato in esame, e si passa ad analizzare lo stato successivo, generando un nuovo nodo che sarà inizialmente etichettato dalle formule A che apparivano prefisse con \bigcirc nel nodo precedente.

Il tableau iniziale per un insieme S di formule è costituito da un unico nodo, etichettato da S stesso. L’espansione di un tableau avviene mediante l’applicazione di una regola ad uno dei suoi nodi “foglia” (cioè un nodo senza figli). Questa porta ad aggiungere uno o due figli al nodo stesso.

Per ridurre il numero delle regole di espansione, assumiamo che l’insieme iniziale S sia costituito da formule in forma normale negativa. L’applicazione di qualsiasi regola conserverà questa proprietà. Le regole di espansione dei tableaux per LTL sono riportate nella tabella 1.1, dove i simboli S e Λ denotano insiemi di formule, A e B sono formule, e la virgola è l’unione insiemistica.

Le due regole classiche sono la riscrittura, in termini di nodi etichettati da insiemi di formule, delle note regole per i tableaux proposizionali classici. Rispetto a questi, è come se in un nodo di un tableau temporale si conservassero tutte le formule “attive” in un ramo (di un tableau con nodi etichettati da singole formule) – dove le formule attive sono i letterali e le formule che possono ancora essere espansive.

Le regole temporali per \Box , \diamond , \mathcal{U} e \mathcal{R} si basano sulle “equazioni di punto fisso” per questi operatori (vedi pagina 12).

⁷P. Wolper, ‘The tableau method for temporal logic: an overview’, *Logique et Analyse* **28** (1985): 119–152.

Regole classiche	
$\frac{A \wedge B, S}{A, B, S} (\wedge)$	$\frac{A \vee B, S}{A, S \quad B, S} (\vee)$
Regole temporali	
$\frac{\Box A, S}{A, \Box A, S} (\Box)$	$\frac{\Diamond A, S}{A, S \quad \Box \Diamond A, S} (\Diamond)$
$\frac{A U B, S}{B, S \quad A, \Box(A U B), S} (U)$	$\frac{A \mathcal{R} B, S}{A, B, S \quad B, \Box(A \mathcal{R} B), S} (\mathcal{R})$
$\frac{\Lambda, \Box A_1, \dots, \Box A_n}{A_1, \dots, A_n} (\Box)$ <p>se Λ è un insieme di letterali</p>	

Tabella 1.1: Le regole di espansione dei tableaux per LTL

Infine, la regola per \Box è quella in cui è terminata l'analisi di uno stato (quello in cui sono veri tutti i letterali in Λ , che non possono più essere ulteriormente analizzati). Il figlio del nodo espanso rappresenta allora un nuovo stato, quello successivo, in cui saranno vere tutte le formule A tali che $\Box A$ è richiesto nello stato precedente.

Possiamo dire che, mentre le regole classiche e quelle per \Box , \Diamond , U e \mathcal{R} sono *statiche* – riguardano cioè l'analisi di un singolo stato –, la regola per \Box è *dinamica*: è conclusa l'analisi di uno stato (rappresentato dai letterali in Λ) e si passa ad analizzare lo stato successivo.

Come nel caso della logica proposizionale classica, quando un nodo contiene un atomo e la sua negazione, esso non viene espanso. Un nodo che contenga una coppia di letterali complementari⁸ sarà chiamato *contraddittorio*.

Formalmente, possiamo definire i tableaux come segue.

Definizione 6 *Un tableau per un insieme S di formule LTL è definito induttivamente:*

1. *Se T è costituito da un unico nodo etichettato da S , allora T è un tableau per S (il tableau iniziale per S).*
2. *Se T è un tableau per S , n è un nodo non contraddittorio di T a cui è pos-*

⁸Ricordiamo che una coppia di letterali complementari è costituita da un atomo e la sua negazione.

sibile applicare una regola di espansione R e T' si ottiene da T espandendo il nodo n mediante la regola R , allora T' è un tableau per S .

1.4.3 Terminazione

Si consideri il tableau della figura 1.2 per l'insieme $S = \{\Box p \wedge \Diamond \neg p\}$. Il nodo 4 è chiuso (contiene due letterali complementari), ma la costruzione del sottoalbero destro può proseguire all'infinito, in quanto ad ogni nodo aperto si può sempre applicare una regola di espansione. Infatti il nodo 6 è etichettato dallo stesso insieme di formule del nodo 2, quindi la costruzione del tableau risulta un processo ciclico infinito. Ciò è dovuto al fatto che – a differenza dei tableaux proposizionali classici – non tutte le regole sostituiscono la formula espansa con formule più semplici (con un minor numero di operatori logici).

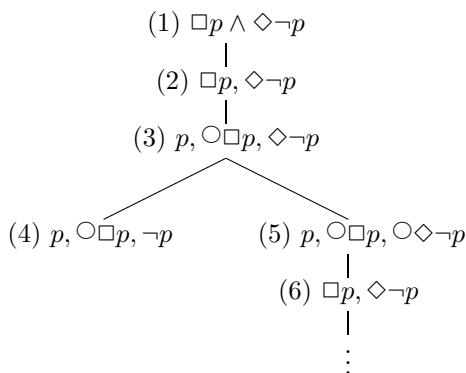


Figura 1.2: Un tableau infinito

Per evitare cicli di questo tipo nella costruzione di un tableau temporale, si introduce un meccanismo di *loop checking* che interrompe la costruzione di un sottoalbero non appena viene generato un nodo con etichetta uguale a quella di un nodo già esistente. Dobbiamo quindi rivedere il modo in cui un nodo viene espanso.

Definizione 7 (Espansione di un nodo.) *Sia n un nodo non contraddittorio di un tableau T etichettato dall'insieme S di formule. Sia inoltre*

$$\frac{S}{S_0} (R) \qquad \text{oppure} \qquad \frac{S}{S_0 \quad S_1} (R)$$

una regola di espansione. Allora il tableau T' che risulta da T espandendo il nodo n mediante la regola R si ottiene da T aggiungendo, per $i = 0$ o $i = 0, 1$:

1. *un nuovo nodo n_i come figlio di T se in T non occorre alcun nodo con etichetta S_i ;*
2. *un arco che connette n a m_i se m_i è un nodo di T etichettato da S_i .*

In altri termini, se un figlio di un nodo n avrebbe etichetta S_i e già esiste un nodo m_i con etichetta S_i , il nuovo nodo non viene creato, ma si aggiunge un arco da n a m_i . Un tableau temporale non è dunque un albero, ma un grafo.

Ad esempio, in figura 1.3 è rappresentato il tableau (finito) corrispondente a quello illustrato in figura 1.2. Come si vede, quando si espande il nodo 5, non viene creato alcun nuovo nodo, ma viene aggiunto un arco dal nodo 5 al nodo 2.

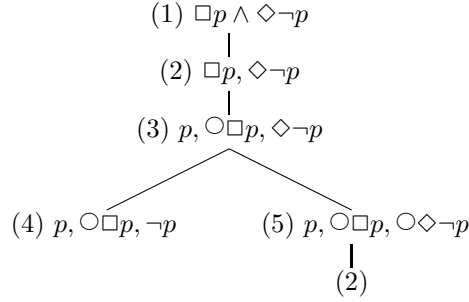


Figura 1.3: Tableau per l'insieme $\{\Box p \wedge \Diamond \neg p\}$

In questo modo si garantisce che la costruzione di qualsiasi tableau termini. Infatti, sia S l'insieme iniziale di formule e

$$subf(S) = \{A \mid A \text{ è una sottoformula di una formula in } S\}$$

Esaminando le regole di espansione, si riconosce facilmente che ogni formula appartenente ad un'etichetta di un qualsiasi tableau per S appartiene all'insieme finito

$$\Sigma = subf(S) \cup \{\bigcirc A \mid A \in subf(S)\}$$

Cioè qualsiasi etichetta di un nodo in un tableau per S è un sottoinsieme di Σ . Se k è la cardinalità di Σ , non possono esserci più di 2^k etichette distinte in un qualsiasi tableau per S . Quindi un tableau per S non può avere più di 2^k nodi.

Ciò significa che per ogni insieme S di formule esiste un tableau finito e completo per S , dove:

Definizione 8 *Un tableau T è completo se nessun nodo di T può essere espanso.*

1.4.4 Tableaux chiusi

Nei tableaux per la logica proposizionale classica, ottenuti considerando soltanto le regole classiche della tabella 1.1, i nodi chiusi sono i nodi contraddittori, un ramo chiuso è un ramo la cui foglia è chiusa e un tableau chiuso è un tableau i cui rami sono tutti chiusi. I nodi/rami/tableaux che non sono chiusi si dicono aperti. Nella logica proposizionale classica, inoltre, ogni ramo aperto in un tableau completo per un insieme S di formule rappresenta un modello di S – quello in cui sono veri esattamente tutti gli atomi che occorrono nella foglia del ramo.

Consideriamo ora i tableaux temporali. Anche qui un cammino⁹ aperto rappresenta un'interpretazione temporale, quindi una sequenza di stati. Ricordando quel che si è detto a proposito delle regole statiche e dinamiche (pagina 17),

⁹Non parliamo più di rami, ma di cammini, dato che un tableau è un grafo e non un albero.

l'analisi di uno stato termina quando viene applicata la regola per \bigcirc , che determina il passaggio ad un altro stato. Quindi i letterali presenti in un nodo a cui viene applicata la regola dinamica (o in un nodo non espandibile) rappresentano uno stato dell'interpretazione. Il successivo si troverà seguendo il cammino fino allo "stato" successivo.¹⁰

Osservando il tableau della figura 1.3, si vede che il nodo 4 è chiuso, dunque lo è il ramo 1,2,3,4, ed anche qualsiasi ramo della forma 1,2,3,5,2,...2,3,4. Ma il cammino (infinito) 1,2,3,5,2,3,5,... non ha alcun nodo contraddittorio. Esso dovrebbe dunque rappresentare un modello della formula $\Box p \wedge \Diamond \neg p$. Osserviamo che fino al nodo 5 vengono applicate regole statiche, mentre nel passaggio da 5 a 2 viene applicata la regola per \bigcirc , mediante la quale si passa ad analizzare un nuovo stato, la cui analisi termina di nuovo nel nodo 5, e così via. Dunque l'interpretazione determinata da questo ramo è la sequenza di stati in ciascuno dei quali sono veri esattamente gli atomi del nodo 5, cioè l'interpretazione \mathcal{M} tale che per ogni $i \in \mathbb{N}$, $\mathcal{M}(i) = \{p\}$; quindi per ogni i , $\mathcal{M}_i \models p$. Ma questo non è chiaramente un modello di $\Box p \wedge \Diamond \neg p$ – che in realtà è insoddisfacibile. Ci aspetteremmo dunque che anche il ramo 1,2,3,5,2,3,5,... sia chiuso. Dovrebbe esserlo perché la formula iniziale "richiede" che prima o poi si verifichi $\neg p$ (con $\Diamond \neg p$) e invece nel cammino considerato $\neg p$ non si verifica mai.

Quel che segue serve appunto a definire una nozione appropriata di tableau chiuso per LTL. Per far ciò occorre la definizione seguente.

Definizione 9 *Una eventuality è una formula della forma $\Diamond A$ oppure $BU A$.*

Nel citato articolo di Wolper, i tableaux chiusi vengono caratterizzati come segue. Definiamo innanzitutto la nozione di *eventuality soddisfatta in un nodo*.

Definizione 10 *Se n è un nodo di un tableau, una eventuality $\Diamond A$ o $BU A$ è soddisfatta in n se esiste un cammino nel tableau da n a un nodo la cui etichetta contiene A .*

Precisiamo che un'etichetta S "contiene" una formula A significa che $A \in S$ e non che A occorre come sottoformula (propria) di qualche formula in S .

Dopo aver terminato la costruzione di un tableau, si procede alla *cancellazione di nodi*:

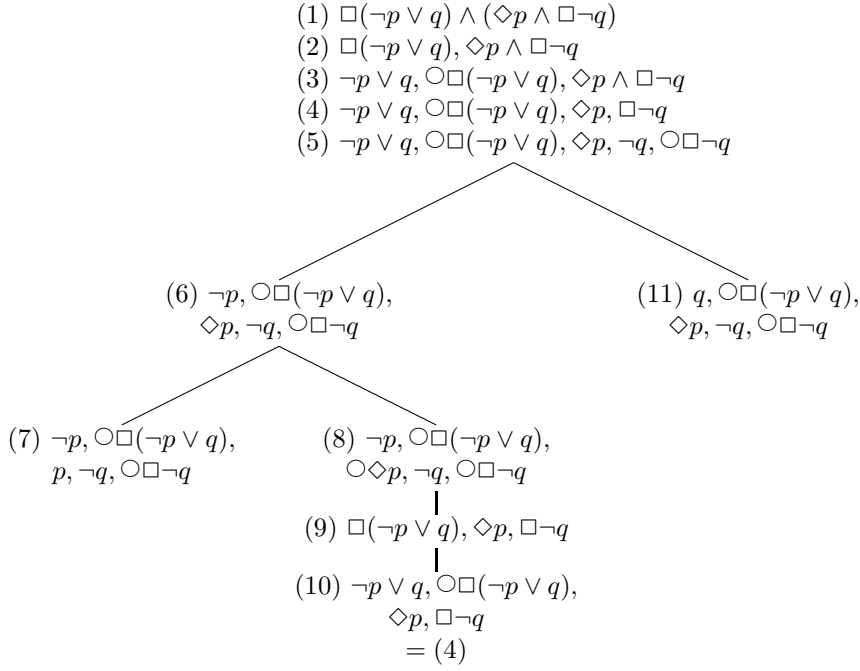
Definizione 11 (Cancellazione di nodi in un tableau completo)

1. se un nodo è contraddittorio, viene cancellato.
2. se un nodo contiene una eventuality che non è soddisfatta nel nodo, viene cancellato;
3. se tutti i figli di un nodo sono cancellati, il nodo stesso viene cancellato.

Definizione 12 *Un tableau è chiuso se la sua radice viene cancellata.*

Nel caso del tableau rappresentato nella figura 1.3, il nodo 4 viene cancellato perché contiene p e $\neg p$. Il nodo 2 viene cancellato perché contiene l'eventuality $\Diamond \neg p$ che non è soddisfatta in 2 (nessun cammino porta da 2 a un nodo che contiene $\neg p$, perché il nodo 4 è stato cancellato). Tutti i figli della radice sono cancellati, dunque la radice è cancellata e il tableau è chiuso.

¹⁰Il modo in cui un cammino rappresenta un'interpretazione temporale sarà precisato nel paragrafo 1.4.5.

Figura 1.4: Un tableau per $\Box(\neg p \vee q) \wedge (\Diamond p \wedge \Box\neg q)$.

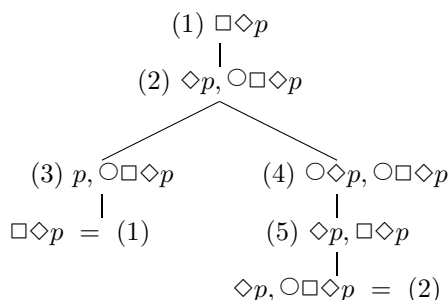
Consideriamo l'esempio più complesso rappresentato in figura 1.4. Si noti che il nodo 10 non viene in realtà creato, ma viene aggiunto un arco dal nodo 9 al nodo 4. Quando la costruzione del tableau è terminata, i nodi 7 e 11 vengono cancellati perché sono contraddittori. Consideriamo ora il nodo 4, che contiene l'eventuality $\Diamond p$. L'unico cammino che parte dal nodo 4 è ora il cammino 4,5,6,8,9,4,... (i nodi 7 e 11 sono stati cancellati). Nessun nodo di questo cammino contiene p , quindi l'eventuality $\Diamond p$ non è soddisfatta in 4, e il nodo 4 viene cancellato. Di conseguenza sono cancellati anche i nodi 3, 2 e 1: il tableau è chiuso.

1.4.5 Cammini aperti

La definizione di tableaux chiusi data nel paragrafo precedente non fornisce una definizione "locale" di chiusura (di nodi o cammini), e non può quindi aiutare a caratterizzare i cammini aperti in un tableau completo (rappresentanti modelli dell'insieme iniziale).

Consideriamo ad esempio il tableau completo della figura 1.5. Nessun nodo in tale tableau viene cancellato. Infatti l'eventuality $\Diamond p$ contenuta nei nodi 2 e 5 è soddisfatta in entrambi i nodi, perché entrambi sono connessi da un cammino al nodo 3 che contiene p . Eppure il cammino 1,2,4,5,2,4,5,... che non passa mai per il nodo 3 non rappresenta un modello della formula iniziale: l'interpretazione determinata da tale cammino è infatti costituita dalla sequenza di interpretazioni classiche rappresentate dal nodo 4 (quello a cui è applicata la regola \bigcirc), in cui p è falso (nel nodo 4 non occorre p).

Nel seguito, per rappresentare cammini utilizzeremo le notazioni abituali per

Figura 1.5: Un tableau aperto per $\Box \Diamond p$

denotare parole di linguaggi omega-regolari (la generalizzazione a parole infinite dei linguaggi regolari), in particolare $(n_1, \dots, n_k)^*$ denoterà una qualsiasi sequenza costituita da zero o più ripetizioni della sequenza n_1, \dots, n_k e $(n_1, \dots, n_k)^\omega$ la sequenza costituita da infinite ripetizioni di n_1, \dots, n_k .

Per poter caratterizzare facilmente i cammini aperti in un tableau, modificheremo le regole di espansione. Innanzitutto precisiamo i casi di applicabilità della regola \bigcirc , generalizzandola rispetto all'uso che ne è stato fatto fin'ora. Precisiamo cioè che essa è applicabile anche a nodi contenenti soltanto letterali:¹¹

$$\frac{\Lambda, \bigcirc A_1, \dots, \bigcirc A_n}{A_1, \dots, A_n} (\bigcirc)$$

dove Λ è un insieme di letterali e $n \geq 0$

Quando $n = 0$, l'espansione è l'insieme vuoto, che può essere rappresentato da \top (la congiunzione di un insieme vuoto di formule è sempre vera).

In tal modo qualsiasi nodo non contraddittorio è espandibile. Anche un nodo n etichettato da \top è espandibile mediante la regola \bigcirc : la sua espansione genererà un arco dal nodo n a n stesso.

Di conseguenza, gli unici cammini finiti in un tableau completo sono quelli che terminano con un nodo contraddittorio. Questo ci consentirà di restringere la ricerca di cammini aperti in un tableau ai soli cammini infiniti.

Ad esempio, la figura 1.6 rappresenta, a sinistra, un tableau costruito senza applicare la regola \bigcirc a nodi contenenti solo letterali, mentre il tableau a destra è il corrispondente tableau ottenuto applicando la regola generale.

Cammini e interpretazioni temporali

Possiamo ora caratterizzare le interpretazioni temporali che sono rappresentate da un cammino infinito di un tableau:

Definizione 13

1. Uno stato di un tableau è un nodo a cui è applicata la regola \bigcirc , cioè un nodo che contiene solo letterali ed eventualmente formule della forma $\bigcirc A$.

¹¹La formulazione precedente in realtà non escludeva questo caso, ma non è in ogni caso necessario applicarla a nodi etichettati da insiemi di letterali perché la caratterizzazione dei tableaux chiusi della definizione 12 sia corretta.

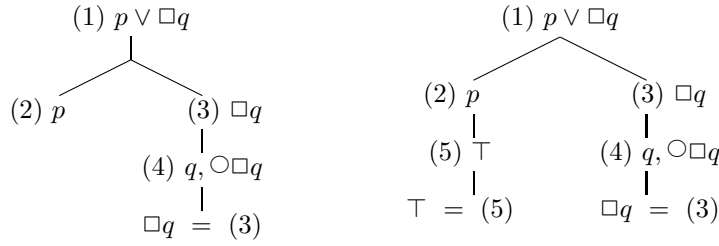


Figura 1.6: Due tableaux equivalenti

2. Se n è uno stato etichettato da S , diciamo che un'interpretazione proposizionale classica I è rappresentata da n sse per ogni atomo p del linguaggio:

$$\begin{aligned} & \text{se } p \in S \text{ allora } p \in I \\ & \text{se } \neg p \in S \text{ allora } p \notin I \end{aligned}$$

Se n è uno stato di un tableau, $\mathcal{I}(n)$ è l'insieme di tutte le interpretazioni classiche rappresentate da n .

3. Sia $\mathcal{C} = n_0, n_1, n_2, \dots$ un cammino infinito in un tableau. Allora la sequenza degli stati del cammino è la sottosequenza massimale di nodi del cammino costituita soltanto da stati, cioè la sequenza di nodi s_0, s_1, s_2, \dots tali che per ogni $i \geq 0$:
- (a) $s_i = n_j$ per qualche j ;
 - (b) s_i è uno stato;
 - (c) se $s_i = n_j$, allora $s_{i+1} = n_{j+k}$ per qualche $k > 0$ (cioè s_{i+1} è successivo s_i nella sequenza n_0, n_1, n_2, \dots);
 - (d) se $s_i = n_j$ e $s_{i+1} = n_{j+k}$, allora nessuno dei nodi n_{j+h} per $0 < h < k$ è uno stato (cioè nel cammino \mathcal{C} non vi sono altri stati tra n_j e n_{j+k}).
4. Sia \mathcal{M} un'interpretazione temporale, $\mathcal{C} = n_0, n_1, n_2, \dots$ un cammino infinito di un tableau e s_0, s_1, s_2, \dots la sequenza degli stati di \mathcal{C} . \mathcal{M} è rappresentata dal cammino \mathcal{C} sse per ogni $i \in \mathbb{N}$, $\mathcal{M}(i) \in \mathcal{I}(s_i)$ (cioè l'interpretazione classica associata allo stato i di \mathcal{M} è rappresentata da s_i).

Si noti che uno stato può rappresentare diverse interpretazioni classiche. Ad esempio se l'insieme di atomi del linguaggio è $\{p, q, r\}$ e gli unici letterali nell'etichetta di n sono p e $\neg q$, allora entrambe le interpretazioni $\{p\}$ e $\{p, r\}$ sono rappresentate da n .

In particolare, uno stato etichettato da \top rappresenta tutte le interpretazioni classiche.

Di conseguenza un cammino di un tableau può rappresentare diverse interpretazioni. Consideriamo, ad esempio, il cammino $\mathcal{C} = 1, 2, 3, 1, (2, 4, 5)^\omega$ del tableau rappresentato in figura 1.5. La corrispondente sequenza di stati è $3, 4^\omega$. Dato che il linguaggio è costituito dal solo atomo p , l'unica interpretazione rappresentata da 3 è $I_1 = \{p\}$. Lo stato 4 rappresenta invece entrambe le interpretazioni I_1 e $I_0 = \emptyset$. Di conseguenza il cammino \mathcal{C} rappresenta tutte le

interpretazioni \mathcal{M} tali che $\mathcal{M}(0) = I_1$ – nessun vincolo è imposto su $\mathcal{M}(i)$ per $i > 0$: $\mathcal{M}(i)$ può essere uguale a I_0 o I_1 , indifferentemente.

Come ulteriore esempio, consideriamo il cammino $\mathcal{C} = 1, 2, 5^\omega$ del tableau di destra della figura 1.6. La corrispondente sequenza di stati è $2, 5^\omega$. \mathcal{C} rappresenta qualsiasi interpretazione temporale \mathcal{M} tale che $p \in \mathcal{M}(0)$ – cioè $\mathcal{M}_0 \models p$.

Si noti che un cammino finito di un *tableau completo* non rappresenta alcuna interpretazione. Infatti l'ultimo nodo del cammino è necessariamente un nodo contraddittorio (altrimenti il nodo sarebbe ancora espandibile e il tableau non sarebbe completo), che non può rappresentare alcuna interpretazione classica.

Cammini aperti in un tableau completo

Per dare una caratterizzazione locale dei cammini aperti, modifichiamo anche le regole di espansione statiche, facendo in modo che i nodi ottenuti tengano traccia delle formule espanse. Nell'espansione, tuttavia, le formule già espanse – che scompaiono nelle regole della tabella 1.1 – vengono marcate in modo da evitare che esse vengano espanse più volte. Quando si applica la regola \circ le formule marcate vengono eliminate.

La tabella 1.2 riporta le nuove regole. Le formule già espanse sono marcate con un asterisco. Si intende che le regole possono essere applicate soltanto per espandere formule non marcate. L'insieme S può contenere sia formule marcate che formule non marcate.

Definizione 14 *In un tableau ottenuto mediante l'applicazione delle regole della tabella 1.2, diciamo che una formula A occorre in un nodo n etichettato da S (e n contiene A) se A appartiene (marcata o non marcata) all'insieme S .*

Si noti che se A occorre solo come *sottoformula propria* di una formula $B \in S$, allora A non occorre in n .

Prima di proseguire, occorre precisare come avviene il loop checking nel nuovo calcolo. La prossima definizione introduce la nozione di “nodo di accettazione di una eventuality”; la caratterizzazione di tali nodi tiene conto di tutte le formule del nodo, marcate o no, di conseguenza le formule non marcate non possono essere ignorate quando si confrontano le etichette di due nodi. Dunque, quando si esegue il controllo di cicli nel sistema con memoria delle formule espanse si utilizza la nozione seguente di uguaglianza delle etichette:

Definizione 15 *Se n_0 e n_1 sono due nodi di un tableau, S_i è l'insieme delle formule non marcate nell'etichetta di n_i e S_i^* l'insieme delle formule marcate nell'etichetta di n_i (per $i = 0, 1$), allora le etichette di n_0 e n_1 sono uguali se $S_0 = S_1$ e $S_0^* = S_1^*$.*

Ciò può portare a volte a proseguire la costruzione del tableau quando, applicando le regole senza memoria delle formule espanse, si potrebbe terminare.

Possiamo ora definire il concetto chiave che ci permetterà di caratterizzare i cammini aperti di un tableau.

Definizione 16 (Nodo di accettazione di una eventuality) *Sia $E = \diamond A$ o $E = BU A$ una eventuality. Un nodo n di un tableau etichettato da S è un nodo di accettazione di E sse $E \notin S$ oppure $A \in S$.*

L'insieme dei nodi di accettazione di E viene indicato con f_E :

$$f_E = \{S \mid E \notin S \text{ oppure } A \in S\}$$

Regole classiche
$\frac{A \wedge B, S}{A, B, (A \wedge B)^*, S} (\wedge)$
$\frac{A \vee B, S}{A, (A \vee B)^*, S \quad B, (A \vee B)^*, S} (\vee)$
Regole temporali
$\frac{\Box A, S}{A, \Box A, (\Box A)^*, S} (\Box)$
$\frac{\Diamond A, S}{A, (\Diamond A)^*, S \quad \Box \Diamond A, (\Diamond A)^*, S} (\Diamond)$
$\frac{A U B, S}{B, (A U B)^*, S \quad A, \Box(A U B), (A U B)^*, S} (U)$
$\frac{A \mathcal{R} B, S}{A, B, (A \mathcal{R} B)^*, S \quad B, \Box(A \mathcal{R} B), (A \mathcal{R} B)^*, S} (\mathcal{R})$
$\frac{\Lambda, \Box A_1, \dots, \Box A_n, B_1^*, \dots, B_k^*}{A_1, \dots, A_n} (\circ)$ <p style="text-align: center; margin-top: 5px;">se Λ è un insieme di letterali e $n \geq 0$</p>

Tabella 1.2: Regole di espansione con memoria delle formule espanse

In altri termini, se un nodo n contiene l'eventuality $E = \diamond A$ o $E = BU A$, allora $n \in f_E$ sse n contiene anche A . Se n non contiene E , allora comunque $n \in f_E$. In particolare, dunque, un nodo etichettato da \top o da un qualsiasi insieme di letterali è un nodo di accettazione per qualsiasi eventuality.

Come esempio, consideriamo ora il tableau per $\square\diamond p$ della figura 1.7 costruito applicando le nuove regole (nella figura 1.5 è illustrato il tableau per la stessa formula costruito applicando le regole senza memoria delle formule espanse).

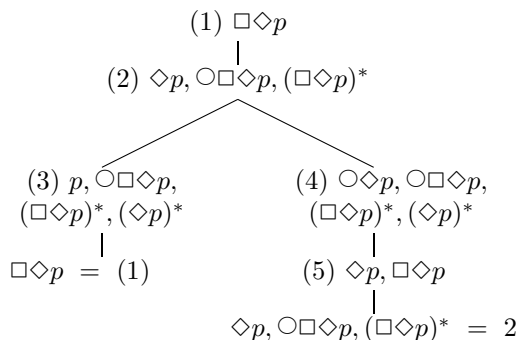


Figura 1.7: Un tableau aperto per $\square\diamond p$

L'unica eventuality che occorre in qualche nodo del tableau è $\diamond p$. I nodi di accettazione di questa eventuality sono

$$f_{\diamond p} = \{1, 3\}$$

Il nodo 1 è d'accettazione perché non contiene $\diamond p$, 3 lo è perché contiene p . Tutti gli altri nodi non lo sono perché contengono $\diamond p$ (marcata o no) e non contengono p .

Possiamo ora finalmente caratterizzare i cammini aperti.

Definizione 17

1. Un cammino infinito di un tableau soddisfa un'eventuality E sse contiene nodi di f_E infinitamente spesso (cioè almeno un nodo di f_E - che è un insieme finito - occorre nel cammino infinite volte).
2. Un cammino \mathcal{C} di un tableau è aperto sse è infinito e soddisfa tutte le eventualities occorrenti in qualche nodo del tableau. \mathcal{C} è chiuso se non è aperto (dunque se è finito oppure non soddisfa qualche eventuality).
3. Un tableau è chiuso se tutti i suoi cammini sono chiusi.

Si noti, di nuovo, che i cammini finiti in un tableau completo sono sempre chiusi. Ciò è coerente con il fatto che un nodo di un tableau completo non ha successori solo se contiene un atomo e la sua negazione, quindi l'ultimo nodo di un cammino finito in un tableau completo è sempre contraddittorio.

Tornando all'esempio della figura 1.7, i cammini aperti sono tutti e solo quelli che contengono infinite volte il nodo 1 o il nodo 3. Poiché tutti i cammini che contengono il nodo 1 infinite volte contengono anche il nodo 3 infinite volte (e viceversa), possiamo equivalentemente caratterizzare i cammini aperti come quelli che contengono infinite occorrenze del nodo 3. Utilizzando la

notazione per parole infinite, i cammini aperti sono tutti quelli della forma $1, 2, ((4, 5, 6)^*, 3, 1, 2, (3, 1, 2)^*)^\omega$. Cioè i cammini che, dopo il “prefisso” $1, 2$, possono contenere qualsiasi numero di ripetizioni dei nodi $4, 5, 6$, ma dopo un numero finito di tali cicli toccano almeno una volta i nodi $3, 1, 2$. Possono poi restare sul ciclo $3, 1, 2$ all’infinito, oppure tornare a ripetere la sequenza $4, 5, 2$, ma di nuovo solo un numero finito di volte, per poi tornare a $3, 1, 2$, ecc. Quindi ad esempio, il cammino $1, 2, 3, 1, 2, 3, 1, (4, 5, 2)^\omega$ è chiuso, così come tutti i cammini della forma $1, 2, n_1, \dots, n_k, (4, 5, 2)^\omega$ (per $k \geq 0$).

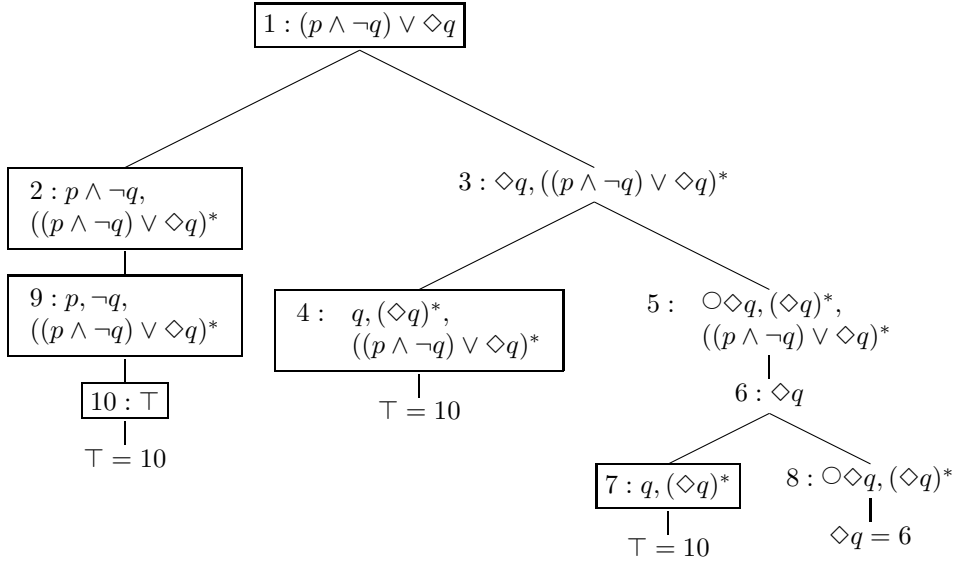


Figura 1.8: Un tableau completo per $(p \wedge \neg q) \vee \diamond q$

Come ulteriore esempio, consideriamo il tableau della figura 1.8. Si noti che la costruzione del tableau utilizzando le regole senza memoria delle formule espanse non avrebbe generato il nodo 6, ma avrebbe aggiunto un arco dal nodo 5 al nodo 3 (il nodo 6 e il nodo 3 hanno le stesse formule non marcate). Qui invece le etichette del nodo 3 e del nodo 6 sono da considerarsi diverse. I nodi di accettazione dell’eventuality $\diamond q$ sono racchiusi in un quadrato: i nodi 1, 2, 9 e 10 non contengono $\diamond q$, i nodi 4 e 7 contengono q . Gli altri, pur contenendo $\diamond q$ non contengono q quindi non sono d’accettazione. I cammini aperti sono tutti e solo quelli che passano per uno dei nodi di accettazione infinite volte. Questa caratterizzazione si può semplificare: sono aperti tutti i cammini che contengono il nodo 10. L’unico cammino chiuso è $1\ 3\ 5\ (6\ 8)^\omega$.

I cammini aperti hanno la proprietà enunciata nel teorema seguente.

Teorema 2 *Se C è un cammino aperto in un tableau completo per un insieme di formule S , allora qualsiasi interpretazione rappresentata da C è un modello di S .*

Viceversa, se un cammino C di un tableau completo è chiuso, non rappresenta alcun modello di S .

La completezza del metodo dei tableaux per la logica temporale è una conseguenza di questo fatto. Infatti, se esiste un tableau completo per S con un cammino aperto, allora S è soddisfacibile, per il teorema precedente. Dunque se S è insoddisfacibile tutti i tableaux completi per S sono chiusi. Poché esiste almeno un tableau completo per S (terminazione), se S è insoddisfacibile esiste un tableau chiuso per S (completezza).

Vale inoltre anche l'inverso del teorema 2: se T è un tableau completo per un insieme S di formule, allora ogni modello di S è rappresentato da un cammino aperto in T (si veda la definizione 13 di interpretazioni rappresentate da cammini). In altri termini si ha che l'insieme dei cammini aperti in un tableau completo per S rappresenta *esattamente* l'insieme dei modelli di S .

1.5 Esercizi

1. Si verifichi che nell'interpretazione \mathcal{M} illustrata nella figura 1.2 di pagina 5:

- (a) $\mathcal{M}_{w_3} \models \Box q$;
- (b) $\mathcal{M}_w \models \Box(q \rightarrow \neg r)$;
- (c) $\mathcal{M}_w \models \Box(q \vee \Box r)$;
- (d) $\mathcal{M}_{w_3} \models \Diamond\Diamond(p \wedge r)$.

2. Verificare la validità delle formule modali riportate a pagina 6.
3. Trovare un contromodello di ciascuna delle formule modali non valide riportate a pagina 7.
4. Dimostrare le proprietà degli operatori temporali riportate nel paragrafo 1.3.3.
5. Dimostrare che:

- (a) $\models \Diamond true$
- (b) $\models \Box true$
- (c) $\Diamond A \not\models \Box\Diamond A$
- (d) $\not\models \Box A \vee \Box\neg A$
- (e) $\models \Diamond A \vee \Diamond\neg A$
- (f) $\Diamond(A \vee B) \leftrightarrow \Diamond A \vee \Diamond B$
- (g) $\Diamond A \wedge \Diamond B \not\models \Diamond(A \wedge B)$
- (h) $\Diamond(A \wedge B) \models \Diamond A \wedge \Diamond B$
- (i) $\Box(A \wedge B) \leftrightarrow \Box A \wedge \Box B$
- (j) $\Box(A \vee B) \not\models \Box A \vee \Box B$
- (k) $\Box A \vee \Box B \models \Box(A \vee B)$
- (l) $\Box A \leftrightarrow \perp \mathcal{R} A$
- (m) $\neg(A \mathcal{U} B) \leftrightarrow \neg B \wedge (\neg A \vee \neg\bigcirc(A \mathcal{U} B))$
- (n) $\neg(A \mathcal{U} B) \leftrightarrow \Box\neg B \vee (\neg B \mathcal{U} (\neg A \wedge \neg B))$

- 6.
7. Si considerino gli enunciati seguenti:
- (A) È sempre vero che se il pollo è nel forno e il forno è acceso, allora prima o poi il pollo sarà cotto.
 - (B) È sempre vero che se il pollo è nel forno e il forno è acceso, allora il forno resta acceso finché il pollo è bruciato oppure il pollo è cotto e fuori dal forno.
- (a) Scrivere due formule A e B di LTL che rappresentino adeguatamente gli enunciati A e B, utilizzando le lettere proposizionali *in_forno*, *acceso*, *cotto*, *bruciato*.
- (b) Definire un'interpretazione \mathcal{M}_1 del linguaggio che sia un modello di A ma non un modello di B , e un'interpretazione \mathcal{M}_2 che sia un modello di B ma non un modello di A . Rappresentare graficamente le due interpretazioni e spiegare perché non sono, rispettivamente, modelli di B e di A .
8. Dimostrare mediante il metodo dei tableaux la validità delle formule riportate nel paragrafo 1.3.3 e le equivalenze logiche dello stesso paragrafo (incluso l'esercizio 5).
9. Per ciascuna delle formule A non valide riportate nell'esercizio 5, costruire un tableau completo per $\neg A$, identificarne i cammini aperti e definire un'interpretazione rappresentata da uno di essi. Verificare che la formula A è falsa in tale interpretazione.
10. Costruire un tableau completo per ciascuna delle formule seguenti, identificarne i cammini aperti e definire un'interpretazione rappresentata da uno di essi.
- (a) $\neg \diamond \Box p$
 - (b) $\diamond \Box p$
 - (c) $\Box (\bigcirc p \wedge \diamond q)$
 - (d) $\neg \Box (p \wedge \bigcirc q \rightarrow \diamond r)$
 - (e) $\Box (p \rightarrow \diamond q)$

Capitolo 2

Verifica di Proprietà di Programmi

2.1 Introduzione

Una delle applicazioni più interessanti della logica temporale lineare è la verifica di sistemi dinamici (software o hardware). Il problema consiste nel verificare se un sistema soddisfa una specifica data. Gli approcci che si basano su LTL sono essenzialmente due: quelli *basati sulla deduzione*, in cui il comportamento del sistema è modellato mediante un insieme S di formule LTL e la specifica da una formula F . La verifica viene ridotta a un problema di deduzione: verificare se $S \models F$. Il secondo approccio è quello basato su *model checking*.

In questa dispensa non parleremo degli approcci basati sulla deduzione, ma affronteremo il problema del *model checking* in LTL. Per definire il problema stesso, occorrerà introdurre in concetto di *automa su parole infinite*. Potremo allora definire automi le cui esecuzioni accettano interpretazioni temporali. Il linguaggio accettato da un tale automa (o semplicemente il linguaggio dell'automa) è dunque un insieme (anche infinito) di interpretazioni temporali: un automa è una rappresentazione finita di tale insieme. Il problema del *model checking* in LTL è allora il seguente: dato un automa il cui linguaggio è costituito dall'insieme T di interpretazioni temporali e una formula F di LTL, determinare se F è vera in ogni interpretazione $\mathcal{M} \in T$. Come vedremo il problema risulta decidibile.

Le tecniche utilizzate per risolvere il problema descritto sopra sono alla base della metodologia di verifica di sistemi a stati finiti mediante *model checking* in LTL. Infatti un sistema a stati finiti si può rappresentare mediante un automa: l'insieme delle sue esecuzioni è costituito dall'insieme di esecuzioni dell'automa che accettano qualche parola. Il problema di verificare se il sistema rappresentato da un automa \mathcal{A} soddisfa una specifica F formulata in LTL viene allora ridotto al problema di verificare se F è vera in tutte le interpretazioni che costituiscono il linguaggio di \mathcal{A} .

2.2 Automi su parole infinite

Nell'accezione più generale, un automa è un grafo con stati iniziali:

Definizione 18 (Automa) *Un automa \mathcal{A} è una tripla $\langle S, \Delta, I \rangle$ dove:*

1. S è un insieme finito (l'insieme degli stati di \mathcal{A});
2. $\Delta \subseteq S \times S$ è una relazione su S (la relazione di transizione: $(s, s') \in \Delta$ se esiste una transizione atomica da s a s');
3. $I \subseteq S$ è l'insieme degli stati iniziali.

A questa caratterizzazione di base si possono aggiungere diverse componenti, quali etichette (sugli stati o sugli archi), insiemi di stati "finali", ecc. Ad esempio gli *automi a stati finiti* utilizzati per descrivere linguaggi formali sono quintuple $\langle S, \Delta, s_0, \Sigma, F \rangle$, dove Σ è l'alfabeto del linguaggio da riconoscere, s_0 è lo stato iniziale, $F \subseteq S$ è l'insieme degli stati finali e Δ è ora una *funzione* $S \times \Sigma \rightarrow S$, cioè una relazione tra stati con etichette sugli archi che rappresentano la relazione (negli automi a stati finiti non deterministici, Δ è una relazione).

In questo contesto ci interesseranno automi con etichettatura degli stati:

Definizione 19 *Un automa con etichettatura degli stati \mathcal{A} è una quintupla $\langle S, \Delta, I, \Sigma, L \rangle$, dove:*

1. S è un insieme finito (l'insieme degli stati di \mathcal{A});
2. $\Delta \subseteq S \times S$ è una relazione su S (la relazione di transizione di \mathcal{A});
3. $I \subseteq S$ è l'insieme degli stati iniziali di \mathcal{A} ;
4. Σ è un insieme finito (l'alfabeto di \mathcal{A});
5. $L : S \rightarrow \Sigma$ è una funzione che associa un'etichetta a ciascuno stato di \mathcal{A} (L è la funzione di etichettatura).

Come esempio, consideriamo le strutture di Kripke introdotte nel paragrafo 2.2 della Prima Parte delle dispense del corso: una struttura di Kripke \mathcal{M} è un automa con etichettatura degli stati, in cui i simboli dell'alfabeto sono insiemi di lettere proposizionali:

$$\mathcal{M} = \langle S, \Delta, I, L, 2^P \rangle$$

(dove P è l'insieme degli atomi del linguaggio).

La differenza più rilevante tra gli automi a stati finiti classici e quelli che utilizzeremo in questo contesto consiste comunque nel fatto che le parole da riconoscere sono infinite, quindi l'accettazione non può fare riferimento allo stato dell'automata al momento della terminazione, perché ora non c'è terminazione. Gli automi su parole infinite sono utili per specificare il comportamento di sistemi che non terminano, come un sistema hardware o un sistema operativo. Tali automi possono anche rappresentare esecuzioni finite, convenendo che l'ultimo stato di un'esecuzione si ripete all'infinito.

Nel seguito, se Σ è un alfabeto (un insieme finito di simboli), indicheremo con Σ^ω l'insieme delle parole infinite su Σ . Ciascuna parola $v \in \Sigma^\omega$ è una funzione $v : \mathbb{N} \rightarrow \Sigma$: per ogni $i \in \mathbb{N}$, $v(i)$ è il simbolo di v in posizione i – in altri termini, la parola è costituita dalla sequenza $v(0), v(1), v(2), \dots$. Un ω -linguaggio sull'alfabeto Σ è un sottoinsieme di Σ^ω .

2.2.1 Automi di Büchi

Gli automi più semplici che lavorano su parole infinite sono gli *automi di Büchi*, che estendono appunto gli automi a stati finiti a input infiniti.¹ Gli automi di Büchi riconoscono i linguaggi chiamati *omega-regolari*, che costituiscono la generalizzazione alle parole infinite dei linguaggi regolari.

Definizione 20 (Automa di Büchi (BA)) *Un automa di Büchi \mathcal{A} è una tupla $\langle S, \Delta, I, L, \Sigma, F \rangle$ dove:*

1. S è un insieme finito di stati;
2. $\Delta \subseteq S \times S$ è la relazione di transizione;
3. $I \subseteq S$ è l'insieme degli stati iniziali di \mathcal{A} ;
4. Σ è un insieme finito (l'alfabeto di \mathcal{A});
5. $L : S \rightarrow \Sigma$ è la funzione di etichettatura;
6. $F \subseteq S$ è l'insieme degli stati di accettazione di \mathcal{A} .

Come si vede, un BA è costituito da un automa con etichettatura sugli stati con una componente aggiuntiva: l'insieme di stati di accettazione.²

L'insieme F servirà a distinguere, all'interno di tutte le possibili esecuzioni dell'automata, quelle che accettano una parola del linguaggio, le *esecuzioni accettanti*. Un'esecuzione, in generale, è un cammino infinito che inizia in uno stato iniziale. Le esecuzioni accettanti sono soltanto quelle in cui qualche stato di accettazione si ripete infinite volte.

Come esempio, consideriamo il BA in cui:

$$\begin{array}{lll} S = \{s_1, s_2\} & \Delta = S \times S & I = S \\ \Sigma = \{A, B\} & L(s_1) = A, L(s_2) = B & F = \{s_1\} \end{array}$$

Esso può essere rappresentato graficamente come nella figura 2.1. Nella rappresentazione grafica di BA, evidenzieremo gli stati di accettazione con un doppio margine.

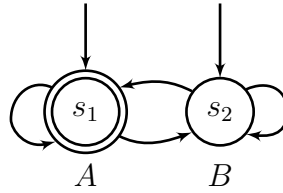


Figura 2.1: Un automa di Büchi

Introduciamo ora i concetti fondamentali per definire il linguaggio riconosciuto da un automa.

¹Julius Richard Büchi (1924—1984) è il logico e matematico svizzero che ha sviluppato la teoria degli automi per il riconoscimento dei linguaggi omega-regolari.

²Gli stati di accettazione sono a volte chiamati anche "stati finali", ma noi eviteremo questa terminologia per evitare confusione con gli stati finali degli automi classici.

Definizione 21 Sia $\mathcal{A} = \langle S, \Delta, I, L, \Sigma, F \rangle$ un automa di Büchi.

1. Un'esecuzione ρ di \mathcal{A} è una funzione $\rho: \mathbb{N} \rightarrow S$ tale che:

- (a) $\rho(0) \in I$;
- (b) $\langle \rho(i), \rho(i+1) \rangle \in \Delta$, per ogni $i \geq 0$.

2. Se $v \in \Sigma^\omega$, un'esecuzione ρ di \mathcal{A} legge v sse per ogni $i \in \mathbb{N}$, $L(\rho(i)) = v(i)$ (l'etichetta dell' i -esimo stato di ρ è uguale all' i -esimo simbolo di v).

3. Un'esecuzione ρ di \mathcal{A} accetta una parola $v \in \Sigma^\omega$ sse ρ legge v e esiste uno stato $s \in F$ che occorre infinite volte in ρ : se $\text{inf}(\rho)$ è l'insieme degli stati che compaiono infinitamente spesso in ρ , si deve avere che

$$\text{inf}(\rho) \cap F \neq \emptyset$$

Se esiste un'esecuzione di \mathcal{A} che accetta v , diciamo che \mathcal{A} accetta v .

4. Il linguaggio di $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^\omega$ di \mathcal{A} è l'insieme delle parole accettate da \mathcal{A} .

Consideriamo ancora come esempio l'automata rappresentato nella figura 2.1, che ha un'unico stato di accettazione. Perché un'esecuzione accetti una parola di Σ^ω essa deve contenere lo stato s_1 un numero infinito di volte. Quindi le parole accettate sono quelle che contengono infinite occorrenze del simbolo A ; ad esempio, A^ω o $(AB)^\omega$. Non sono accettate ad esempio le parole B^ω e AB^ω . Il linguaggio dell'automata si può denotare mediante l'espressione ω -regolare $(B^*A)^\omega$.

2.2.2 Automi e formule temporali

Sia P un insieme di lettere proposizionali e consideriamo un automa con etichette in 2^P :

$$\mathcal{A} = (S, \Delta, I, L, 2^P, F)$$

Le parole lette da \mathcal{A} sono *sequenze infinite di sottoinsiemi di P* : X_1, X_2, X_3, \dots con $X_i \subseteq P$. Ogni insieme $X_i \subseteq P$ rappresenta un'interpretazione proposizionale classica, quindi le parole lette da \mathcal{A} sono interpretazioni del linguaggio di LTL su P : ogni esecuzione ρ dell'automata legge l'interpretazione temporale \mathcal{M} tale che, per ogni $i \in \mathbb{N}$, $\mathcal{M}(i) = L(\rho(i))$.

Consideriamo ora una formula proposizionale F . Essa determina un insieme di interpretazioni proposizionali, tutte quelle in cui F è vera. Analogamente, una formula di LTL determina l'insieme di interpretazioni temporali in cui è vera. Si può in un certo senso identificare una formula con un insieme di interpretazioni. Come vedremo nel resto di queste dispense, da una formula F di LTL è possibile costruire un automa \mathcal{A} ad essa "equivalente", nel senso che il linguaggio di \mathcal{A} è costituito esattamente dai modelli di F .

Automi etichettati da formule proposizionali

Per arrivare a costruire un automa di Büchi che rappresenta una formula LTL, introduciamo innanzitutto una "notazione compatta" per rappresentare un BA etichettato da interpretazioni proposizionali. Consideriamo ad esempio l'automata rappresentato a sinistra nella figura 2.2, sul linguaggio proposizionale costituito

dagli atomi A e B . Tre stati di questo automa, r_1 , r_2 e r_3 , hanno gli stessi archi entranti e uscenti e nessuno dei tre è uno stato di accettazione. Questi tre stati si possono far collapsare in un unico stato, etichettato dall'insieme costituito dalle tre etichette di r_1 , r_2 e r_3 . Ma un insieme di interpretazioni si può identificare con una formula. L'automata di destra della figura 2.2 corrisponde appunto alla rappresentazione compatta dell'automata di sinistra: gli stati sono tutti etichettati da formule; uno stato etichettato dalla formula F rappresenta un insieme di stati, etichettati ciascuno da un'interpretazione in cui F è vera. In altri termini, in uno stato etichettato da F può essere letto uno qualsiasi dei "simboli" in cui F è vera.

Nell'esempio, lo stato q' dell'automata di destra rappresenta soltanto uno stato, lo stato q dell'automata di sinistra; la formula che lo etichetta è vera soltanto nell'interpretazione $\{B\}$ (l'etichetta dello stato q). Lo stato r invece corrisponde ai tre stati r_1 , r_2 e r_3 ; la sua etichetta, $A \vee \neg B$ è una formula che è vera nelle tre interpretazioni che etichettano r_1 , r_2 e r_3 , e falsa in $\{B\}$.³

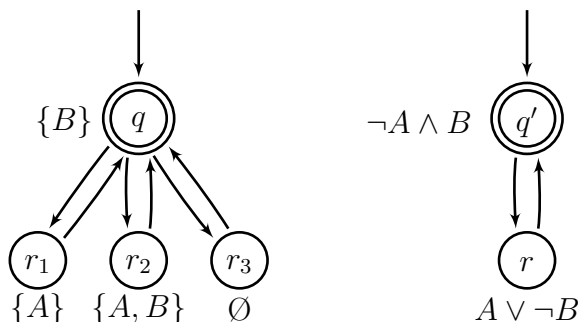


Figura 2.2: Rappresentazione compatta di BA etichettati da insiemi di atomi

In generale, un *automa etichettato da formule* è una rappresentazione compatta di un automa etichettato da insiemi di atomi. Se F è la formula proposizionale che etichetta lo stato s , s rappresenta un insieme di stati, le cui etichette sono costituite da tutte e solo le interpretazioni in cui F è vera. In questo modo si ottiene una rappresentazione che può essere trattata in maniera più efficiente. Infatti, la complessità di molti algoritmi su automi dipende dal numero di stati degli automi.

Si noti che il linguaggio accettato da un automa etichettato da formule non cambia, si tratta soltanto di una differente rappresentazione: le parole lette da un automa etichettato da formule sono sempre interpretazioni temporali \mathcal{M} . Nel caso degli automi della figura 2.2, entrambi leggono parole \mathcal{M} tali che $\mathcal{M}_i \models \neg A \wedge B$ se i è pari, e $\mathcal{M}_i \models A \vee \neg B$ se i è dispari.

³Dato un insieme I_1, \dots, I_n di interpretazioni proposizionali, si può facilmente costruire una formula proposizionale che è vera esattamente in I_1, \dots, I_n come segue. Per ogni interpretazione I_k , con $1 \leq k \leq n$, si costruisce la congiunzione F_k di letterali tale che se $p \in I$ allora p è un letterale della congiunzione, altrimenti lo è $\neg p$. Ovviamente I_k è l'unico modello di F_k . Una formula i cui modelli sono esattamente I_1, \dots, I_n è la disgiunzione $F_1 \vee \dots \vee F_n$. Tale formula può eventualmente essere semplificata.

Ad esempio, dalle tre interpretazioni $\{A\}$, $\{A, B\}$ e \emptyset si ottiene la formula $F = (A \wedge \neg B) \vee (A \wedge B) \vee (\neg A \wedge \neg B)$. La formula indicata come etichetta del nodo r nella figura 2.2 è equivalente a F .

In generale, in un automa sul linguaggio P etichettato da formule, la funzione di etichettatura è

$$L : S \rightarrow 2^{2^P}$$

In altri termini, $L(s)$ è un insieme di assegnazioni proposizionali, rappresentato da una formula proposizionale classica con atomi in P .

Un automa etichettato da formule legge parole $v : \mathbb{N} \rightarrow 2^P$, costituite cioè da sequenze di assegnazioni proposizionali classiche, esattamente come gli automi con funzione di etichettatura del tipo $L : S \rightarrow 2^P$. Una parola $v : \mathbb{N} \rightarrow 2^P$ letta dall'automato corrisponde all'interpretazione temporale \mathcal{M} tale che $\mathcal{M}(i) = v(i)$.

Per comodità definiamo in modo indipendente la nozione di esecuzione di un automa etichettato da formule:

Definizione 22 Sia $\mathcal{A} = \langle S, \Delta, I, L, \Sigma, F \rangle$ un automa di Büchi etichettato da formule proposizionali. Un'esecuzione ρ di \mathcal{A} che legge la parola v è un mapping $\mathbb{N} \rightarrow S$ tale che:

- $\rho(0) \in I$;
- $\langle \rho(i), \rho(i+1) \rangle \in \Delta$, per ogni $i \geq 0$;
- $v(i) \models L(\rho(i))$ per ogni $i \geq 0$.

Si osservi che se uno stato $\rho(i)$ è etichettato da \top , allora per ogni “simbolo” $v(i)$:

$$v(i) \models L(\rho(i))$$

E se uno stato $\rho(i)$ è etichettato da \perp , allora per nessun simbolo $v(i)$ si avrà che $v(i) \models L(\rho(i))$. In altri termini, nessuna parola verrà letta da un'esecuzione che passa per uno stato etichettato da \perp (che è la rappresentazione compatta dell'insieme vuoto di stati etichettati da interpretazioni proposizionali). Di conseguenza, gli stati etichettati di \perp possono essere eliminati (con tutti gli archi entranti e uscenti) da un automa, senza modificare il suo linguaggio.

2.2.3 Automi di Büchi generalizzati

Sempre per raggiungere lo scopo di costruire BA rappresentanti formule LTL, abbiamo bisogno di considerare una versione generalizzata degli automi di Büchi:

Definizione 23 (Automa di Büchi generalizzato (GBA)) Un GBA è una tupla $\mathcal{A} = \langle S, \Delta, I, L, \Sigma, F \rangle$ dove S , Δ , I , Σ e L sono come nella definizione 20, e F è un insieme di insiemi di stati di accettazione:

$$F = \{f_1, f_2, \dots, f_m\} \text{ con } f_i \subseteq S$$

La nozione di esecuzione (run) e parola letta da un'esecuzione sono come nella definizione 21 (o la definizione 22 se si tratta di un automa etichettato da formule).

Un'esecuzione ρ di un GBA $\mathcal{A} = \langle S, \Delta, I, L, \Sigma, F \rangle$ accetta una parola $v \in \Sigma^\omega$ sse ρ legge v e

$$\text{inf}(\rho) \cap f_i \neq \emptyset \text{ per ogni } f_i \in F$$

(ricordiamo che $\text{inf}(\rho)$ è l'insieme degli stati che compaiono infinitamente spesso in ρ).

In altri termini, in un GBA ci possono essere più insiemi di stati di accettazione f_1, f_2, \dots . Le esecuzioni accettanti sono quelle che passano per *ciascuno* degli insiemi f_i infinite volte.

Consideriamo, per esempio, l'automa rappresentato nella figura 2.3, che ha due insiemi di stati di accettazione. Gli elementi del primo sono rappresentati in figura da un doppio cerchio, quelli del secondo da un doppio quadrato. L'automa ha cioè l'insieme $F = \{f_1, f_2\}$ di insiemi di stati di accettazione, dove $f_1 = \{s_0\}$ e $f_2 = \{s_1\}$. L'esecuzione $s_1 s_0 (s_1 s_2)^\omega$ non accetta alcuna parola, perchè, sebbene contenga stati di f_2 infinite volte, non contiene stati di f_1 infinitamente spesso (s_0 occorre un numero finito di volte – una sola). Il linguaggio accettato dall'automa è infatti costituito dalle parole della forma $(B(CB)^*A)^\omega$.

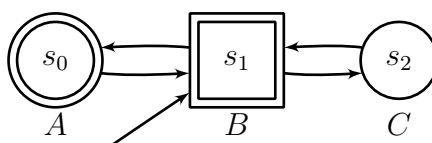


Figura 2.3: Un automa di Büchi generalizzato

Gli automi di Büchi generalizzati non aggiungono comunque potere espressivo agli automi semplici, nel senso che l'insieme dei linguaggi accettati da GBA è uguale a quello dei linguaggi accettati da BA. È infatti possibile tradurre un GBA in un BA “equivalente”, che accetta cioè lo stesso linguaggio. Nel resto di questo paragrafo mostriamo come costruire, a partire da un GBA, un BA ad esso equivalente.

Sia $\mathcal{A} = (S, \Delta, I, L, \Sigma, \{f_1, \dots, f_m\})$ un GBA con m insiemi di stati di accettazione. La traduzione $\mathcal{A}^* = (S^*, \Delta^*, I^*, L^*, \Sigma, F)$ di \mathcal{A} è il BA così definito:

Stati. Ciascuno stato di \mathcal{A}^* è identificato da una coppia della forma (s, i) , dove s è uno stato di \mathcal{A} e $i = 1, \dots, m$. In altri termini, S^* contiene m copie distinte di S (dove m è il numero di insiemi di stati di accettazione di \mathcal{A}):

$$S^* = S \times \{1, \dots, m\} = \{(s, i) \mid s \in S, 1 \leq i \leq m\}$$

Stati iniziali. Gli stati iniziali di \mathcal{A}^* sono quelli che corrispondono agli stati iniziali della “prima copia”:

$$I^* = I \times \{1\} = \{(s, 1) \mid s \in I\}$$

Alfabeto. L'alfabeto di \mathcal{A}^* è lo stesso di \mathcal{A} .

Etichette. L'etichetta di qualsiasi copia di uno stato s è uguale all'etichetta di s : per ogni i , $L^*((s, i)) = L(s)$

Relazione di transizione. Per definire la relazione di transizione di \mathcal{A}^* , si definisce innanzitutto l'operazione $i \oplus_m 1$ sull'insieme $\{1, \dots, m\}$ (una specie di “somma modulo m ”):

$$i \oplus_m 1 = \begin{cases} i + 1 & \text{se } i < m \\ 1 & \text{se } i = m \end{cases} \quad \text{oppure: } i \oplus_m 1 = (i \bmod m) + 1$$

Le transizioni Δ^* dell'automa \mathcal{A}^* sono allora definite come segue: per ogni transizione (s, s') in Δ e per ogni $i = 1, \dots, m$, Δ^* contiene

$$\begin{aligned} &((s, i), (s', i \oplus_m 1)) && \text{se } s \in f_i \\ &((s, i), (s', i)) && \text{altrimenti} \end{aligned}$$

Cioè, nelle esecuzioni di \mathcal{A}^* , dalla i -esima copia dello stato s (cioè da (s, i)) si passa a $(s', i \oplus_m 1)$ – cioè si passa alla “copia” successiva – se s è nell' i -esimo insieme di accettazione f_i ; altrimenti si resta nella stessa “copia”.

In questo modo, se un'esecuzione di \mathcal{A} passa per ogni insieme di accettazione di \mathcal{A} infinite volte, la corrispondente esecuzione di \mathcal{A}^* cicla sulle m copie di S .

Stati di accettazione: sono gli stati di f_1 nella prima copia:

$$F = f_1 \times \{1\} = \{(s, 1) \mid s \in f_1\}$$

Se un'esecuzione di \mathcal{A}^* passa per uno stato di F infinite volte, vuol dire che ogni volta l'esecuzione è passata per tutte le copie di S ; quindi, per ogni $i = 1, \dots, m$ l'esecuzione passa per una copia di uno stato in f_i (per passare dall' i -esima copia di S alla successiva, deve passare per l' i -esima copia di uno stato in f_i). Viceversa, se un'esecuzione di \mathcal{A} passa per ogni insieme di accettazione di \mathcal{A} infinite volte, la corrispondente esecuzione di \mathcal{A}^* cicla sulle m copie di S , quindi torna infinite volte sulla prima copia: ogni volta, per passare alla seconda copia, passerà per uno stato di F .

Ad esempio, l'automa generalizzato rappresentato in figura 2.3, dove $F = \{f_1, f_2\}$ con $f_1 = \{s_0\}$, $f_2 = \{s_1\}$, si “traduce” nell'automa rappresentato nella figura 2.4. Nella figura gli stati di una stessa copia sono rappresentati allo stesso livello. Come si vede, gli archi uscenti dagli stati $(s_0, 1)$ e $(s_1, 2)$ “cambiano copia”, dato che $(s_0, 1) \in f_1$ e $(s_1, 2) \in f_2$. Gli archi uscenti dagli altri stati restano invece nella stessa copia. L'insieme degli stati di accettazione contiene solo la prima copia dell'unico stato di accettazione di f_1 , cioè lo stato $(s_0, 1)$. Ovviamente, l'automa della figura 2.4 si può semplificare eliminando gli stati $(s_0, 2)$ e $(s_1, 2)$ che non sono raggiungibili da nessuno stato iniziale.

Consideriamo anche la traduzione di GBA in BA in due casi particolari. Il primo è un caso banale: se $\mathcal{A} = (S, \Delta, I, L, \Sigma, \{f_1\})$ è un GBA con un solo insieme in F , allora la sua traduzione è il BA $(S, \Delta, I, L, \Sigma, f_1)$. Inoltre, se uno degli insiemi di accettazione coincide con S , il GBA può essere prima semplificato: se $\mathcal{A} = (S, \Delta, I, L, \Sigma, \{f_1, f_2, \dots, f_m\})$ dove $f_1 = S$ allora la sua traduzione è la traduzione di $(S, \Delta, I, L, \Sigma, \{f_2, \dots, f_m\})$.

2.2.4 Operazioni su automi di Büchi

Una prima operazione importante sui BA è verificare se il linguaggio di un automa è vuoto oppure no (*checking emptiness*). Osserviamo innanzitutto che, se \mathcal{A} è un BA, $\mathcal{L}(\mathcal{A}) \neq \emptyset$ se e solo se \mathcal{A} ha almeno un'esecuzione accettante (esiste ρ tale che $\text{inf}(\rho) \cap F \neq \emptyset$). Ciò vale se e solo se in \mathcal{A} esiste una componente fortemente connessa (SCC) raggiungibile da uno stato iniziale e contenente almeno uno stato di accettazione. Questo è equivalente a dire che esiste uno

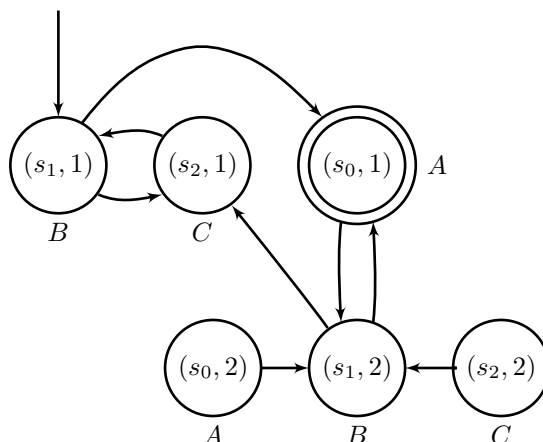


Figura 2.4: Un automa di Büchi equivalente all'automata della figura 2.3

stato di accettazione $s \in F$ raggiungibile da uno stato iniziale e raggiungibile da se stesso (cioè se esiste un ciclo contenente almeno uno stato di accettazione e raggiungibile da uno stato iniziale). Tale condizione si può facilmente verificare effettuando una visita del grafo corrispondente ad \mathcal{A} , che richiede tempo lineare nella dimensione dell'automata.

Un automa \mathcal{C} è chiamato *automa unione* di \mathcal{A} e \mathcal{B} (e può essere denotato da $\mathcal{A} \cup \mathcal{B}$) se $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{B})$. Analogamente si definisce l'automata *intersezione* e l'automata *complemento*.

Gli automi di Büchi sono chiusi rispetto a unione, intersezione, complemento. Questo significa che, se \mathcal{A} e \mathcal{B} sono BA, esistono automi:

$$\begin{aligned} \mathcal{A} \cup \mathcal{B} \text{ tale che } \mathcal{L}(\mathcal{A} \cup \mathcal{B}) &= \mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{B}) \\ \mathcal{A} \cap \mathcal{B} \text{ tale che } \mathcal{L}(\mathcal{A} \cap \mathcal{B}) &= \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}) \\ \bar{\mathcal{A}} \text{ tale che } \mathcal{L}(\bar{\mathcal{A}}) &= \overline{\mathcal{L}(\mathcal{A})} \end{aligned}$$

La costruzione dell'automata complemento è piuttosto complicata, e, quando applicata a un automata con n stati, risulta in un automata che ha un numero di stati dell'ordine di $n!$. In queste dispense non ne parleremo.

È invece molto facile costruire l'automata unione di due automi: esso si ottiene semplicemente giustapponendo i due automi. Più in dettaglio, siano

$$\begin{aligned} \mathcal{A}_1 &= (\Sigma, S_1, \Delta_1, I_1, L_1, F_1) \\ \mathcal{A}_2 &= (\Sigma, S_2, \Delta_2, I_2, L_2, F_2) \end{aligned}$$

automi sullo stesso alfabeto Σ . Possiamo assumere senza perdita di generalità che $S_1 \cap S_2 = \emptyset$ (se così non fosse, si rinominano gli stati in modo da ottenere insiemi disgiunti). L'automata unione di \mathcal{A}_1 e \mathcal{A}_2 è definito come segue:

$$\mathcal{A}_1 \cup \mathcal{A}_2 = (\Sigma, S_1 \cup S_2, \Delta_1 \cup \Delta_2, I_1 \cup I_2, L, F_1 \cup F_2)$$

dove la funzione di etichettatura L è tale che $L(s) = \begin{cases} L_1(s) & \text{se } s \in S_1 \\ L_2(s) & \text{se } s \in S_2 \end{cases}$

La costruzione dell'automa intersezione è un po' più complessa. Nel resto di questo paragrafo mostriamo come costruire l'intersezione di due automi *etichettati da formule*.

Se \mathcal{A}_1 e \mathcal{A}_2 sono BA etichettati da formule (sullo stesso alfabeto), si costruisce in primo luogo un GBA $\mathcal{A}_1 \cap \mathcal{A}_2$, che accetta il linguaggio $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$. Il GBA può poi essere trasformato in un BA che accetta lo stesso linguaggio. La costruzione è fatta in modo tale che un'esecuzione di $\mathcal{A}_1 \cap \mathcal{A}_2$ simula due esecuzioni simultanee, una su \mathcal{A}_1 e una su \mathcal{A}_2 . Un'esecuzione accettata deve passare infinite volte per stati (corrispondenti a stati) di accettazione di entrambi gli automi originari.

L'automa intersezione $\mathcal{A}_1 \cap \mathcal{A}_2$ è definito come segue.

Stati: uno stato di $\mathcal{A}_1 \cap \mathcal{A}_2$ contiene uno stato di \mathcal{A}_1 e uno di \mathcal{A}_2 :

$$S = S_1 \times S_2 = \{\langle s_1, s_2 \rangle \mid s_1 \in S_1 \text{ e } s_2 \in S_2\}$$

Alfabeto: l'alfabeto è lo stesso di \mathcal{A}_1 e \mathcal{A}_2 .

Etichette: l'etichetta di uno stato è la congiunzione delle etichette degli stati componenti:

$$L(\langle s_1, s_2 \rangle) = L_1(s_1) \wedge L_2(s_2)$$

Relazione di transizione:

$$\langle \langle q, r \rangle, \langle q', r' \rangle \rangle \in \Delta \text{ sse } (q, q') \in \Delta_1 \text{ e } (r, r') \in \Delta_2$$

Stati iniziali:

$$\langle q, r \rangle \in I \text{ sse } q \in I_1 \text{ e } r \in I_2$$

Stati di accettazione:

$$F = \{f_1, f_2\} \text{ con } f_1 = F_1 \times S_2, f_2 = S_1 \times F_2$$

Vengono visitati infinitamente spesso sia stati di F_1 che stati di F_2 (ma non necessariamente contemporaneamente)

Consideriamo come esempio i due automi rappresentati in alto nella figura 2.5. L'insieme degli stati dell'automa intersezione è $\{(s_0, t_0), (s_0, t_1), (s_1, t_0), (s_1, t_1)\}$. Consideriamo ora la funzione di etichettatura:

$$\begin{aligned} L(\langle s_0, t_0 \rangle) &= A \wedge \neg B \wedge \neg A \wedge B && \leftrightarrow \perp \\ L(\langle s_0, t_1 \rangle) &= A \wedge \neg B \wedge (A \vee \neg B) && \leftrightarrow A \wedge \neg B \\ L(\langle s_1, t_0 \rangle) &= \neg A \wedge \neg A \wedge B && \leftrightarrow \neg A \wedge B \\ L(\langle s_1, t_1 \rangle) &= \neg A \wedge (A \vee \neg B) && \leftrightarrow \neg A \wedge \neg B \end{aligned}$$

Poiché l'etichetta dello stato (s_0, t_0) è una contraddizione, tale stato può essere eliminato dall'automa. L'intersezione dei due automi è rappresentata in basso nella figura 2.5. Come si vede, ci sono due insiemi di stati di intersezione: $\{(s_0, t_1)\}$ (perché s_0 è uno d'accettazione dell'automa di sinistra – e (s_0, t_0) è stato eliminato) e $\{(s_1, t_0)\}$ (perché t_0 è uno d'accettazione dell'automa di destra – e (s_0, t_0) è stato eliminato).

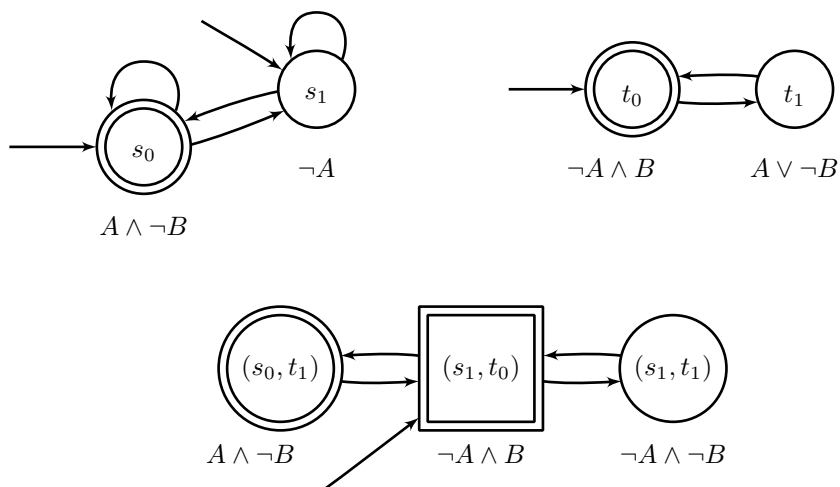


Figura 2.5: Intersezione di due automi

Unione, intersezione, complemento nei BA corrispondono agli operatori logici \wedge, \vee, \neg , nel senso che segue. Come abbiamo anticipato, e come mostreremo nel prossimo capitolo, una formula F di LTL si può rappresentare mediante un BA \mathcal{A}_F etichettato da formule (proposizionali classiche), tale che: $\mathcal{L}(\mathcal{A}_F) = \{\mathcal{M} \mid \mathcal{M} \models F\}$. Si ha allora che:

$$\begin{aligned} \mathcal{L}(\mathcal{A}_{F \wedge G}) &= \{\mathcal{M} \mid \mathcal{M} \models F \text{ e } \mathcal{M} \models G\} = \mathcal{L}(\mathcal{A}_F) \cap \mathcal{L}(\mathcal{A}_G) = \mathcal{L}(\mathcal{A}_F \cap \mathcal{A}_G) \\ \mathcal{L}(\mathcal{A}_{F \vee G}) &= \{\mathcal{M} \mid \mathcal{M} \models F \text{ o } \mathcal{M} \models G\} = \mathcal{L}(\mathcal{A}_F) \cup \mathcal{L}(\mathcal{A}_G) = \mathcal{L}(\mathcal{A}_F \cup \mathcal{A}_G) \\ \mathcal{L}(\mathcal{A}_{\neg F}) &= \{\mathcal{M} \mid \mathcal{M} \not\models F\} = \overline{\mathcal{L}(\mathcal{A}_F)} = \mathcal{L}(\overline{\mathcal{A}_F}) \end{aligned}$$

2.3 Verifica di sistemi basata su model checking in LTL

Torniamo al problema della verifica di un sistema a stati finiti. Un sistema di questo tipo si può rappresentare mediante un automa di Büchi. L'insieme degli stati di accettazione dell'automa rappresenta eventuali *vincoli di fairness* imposti sul sistema: avviene spesso che il modello del sistema non sia sufficientemente dettagliato per rappresentare adeguatamente le esecuzioni del sistema, e vi sono alcune esecuzioni che, sebbene possibili nel modello, non lo sono nel sistema. Dettagliare ulteriormente il modello potrebbe però non essere conveniente, perché porterebbe ad aumentare il numero degli stati e dunque la complessità della verifica. Si possono allora imporre dei vincoli semantici sulle esecuzioni, per escludere quelle irragionevoli. Tali vincoli sono detti “vincoli di *fairness*” e si possono modellare appunto mediante un insieme di stati di accettazione. Nel caso in cui non si vogliono imporre vincoli di *fairness*, l'insieme degli stati di accettazione dell'automa che modella il sistema contiene tutti gli stati dell'automa. Dunque, se identifichiamo un sistema a stati finiti con l'insieme delle sue esecuzioni, esso può essere modellato mediante un BA etichettato da formule proposizionali. Il linguaggio di quest'ultimo, ricordiamo, è costituito da un insieme di interpretazioni temporali.

Il problema di verificare se il sistema rappresentato da un automa \mathcal{A} soddisfa una specifica F formulata in LTL viene allora ridotto al problema di verificare se F è vera in tutte le interpretazioni che costituiscono il linguaggio di \mathcal{A} .

In questo capitolo mostreremo come il problema del *model checking* in LTL si possa risolvere mediante operazioni tra automi. Infatti il problema si può affrontare in questo modo: a partire da una formula temporale F si costruisce un automa \mathcal{A}_F il cui linguaggio è costituito esattamente dai modelli di F . Verificare se un automa \mathcal{B} soddisfa una specifica F (cioè se tutte le esecuzioni di \mathcal{B} sono modelli di F) si riduce a verificare se il linguaggio di \mathcal{B} è incluso nel linguaggio di \mathcal{A}_F :

$$\mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}(\mathcal{A}_F)$$

Questa condizione equivale a dire che

$$\mathcal{L}(\mathcal{B} \cap \overline{\mathcal{A}_F}) = \emptyset$$

cioè che l'intersezione tra \mathcal{B} e il complemento di \mathcal{A}_F è vuota: “tutte le parole accettate da \mathcal{B} sono accettate da \mathcal{A}_F ” equivale a “non esiste alcuna parola accettata da \mathcal{B} che non sia accettata da \mathcal{A}_F ”.

Per la verifica di tale condizione, non avremo bisogno di costruire il complemento di \mathcal{A}_F : infatti, come osservato a pagina 40, $\mathcal{L}(\overline{\mathcal{A}_F}) = \mathcal{L}(\mathcal{A}_{\neg F})$. Di conseguenza, controllare se il sistema modellato da \mathcal{B} soddisfa la specifica F equivale a controllare se

$$\mathcal{L}(\mathcal{B} \cap \mathcal{A}_{\neg F}) = \emptyset$$

Dunque il problema del model checking in LTL viene risolto come segue: per verificare se una formula F è vera in tutte le interpretazioni del linguaggio di un automa \mathcal{B} :

1. si traduce la formula $\neg F$ in un automa $\mathcal{A}_{\neg F}$, tale che il linguaggio di $\mathcal{A}_{\neg F}$ è costituito esattamente dai modelli di $\neg F$;
2. si costruisce l'automata intersezione $\mathcal{B} \cap \mathcal{A}_{\neg F}$ e si controlla se il suo linguaggio è vuoto oppure no.

Abbiamo già visto nel capitolo precedente come risolvere i problemi del secondo passo. Notiamo in particolare che, nel caso in cui l'automata \mathcal{B} non soddisfi la specifica F , verrà identificata un'esecuzione accettante ρ dell'automata $\mathcal{B} \cap \mathcal{A}_{\neg F}$, e questa fornirà importanti informazioni: le parole accettate da ρ sono infatti interpretazioni temporali⁴ e ciascuna di esse costituisce un esempio di esecuzione (accettante) di \mathcal{B} in cui F è falsa; è dunque un esempio di esecuzione *fair* del sistema che non soddisfa la specifica F .

In questo capitolo mostreremo l'altra procedura chiave per affrontare il problema del model checking in LTL: la costruzione di un automa che rappresenti una formula LTL.

⁴Ricordiamo che un'esecuzione di un automa etichettato da formule è una rappresentazione compatta di un insieme di esecuzioni del corrispondente automa etichettato da interpretazioni proposizionali, e può dunque “leggere” diverse interpretazioni temporali.

2.3.1 Traduzione di formule LTL in GBA

Sia F una formula LTL. La costruzione dell'automa \mathcal{A}_F equivale sostanzialmente alla costruzione di un tableau per F applicando le regole con memoria delle formule espanse, ignorando i nodi che non siano stati.

Più precisamente, si costruisce innanzitutto un tableau completo T per la formula F , e da esso si cancellano, ricorsivamente, tutti i nodi senza figli (quelli contraddittori e quelli che, a seguito della cancellazione, rimangono senza figli).

Il tableau T si può vedere come un automa $T = (N, \Delta_T, \{n_0\}, L_T, \Sigma_T, F_T)$, dove:

- l'insieme degli stati N è l'insieme dei nodi;
- lo stato iniziale è la radice $n_0 \in N$ del tableau;
- Δ_T è la relazione padre-figlio;
- Σ_T è l'alfabeto:

$$\Sigma_T = 2^{subf(F) \cup \{\circ B \mid B \in subf(F)\}}$$

le etichette dei nodi sono infatti insiemi di formule, tutti sottoinsiemi dell'insieme $subf(F) \cup \{\circ B \mid B \in subf(F)\}$;

- $L_T : N \rightarrow \Sigma$ è la funzione di etichettatura dei nodi;
- l'automa ha un insieme di stati di accettazione per ogni eventuality E che occorre in F ; ciascuno di tali insiemi contiene esattamente i nodi di accettazione di E . Cioè $F_T = \{f_{E_1}, \dots, f_{E_m}\}$, dove E_1, \dots, E_m sono tutte le eventualities in $subf(F)$ e per ogni $i = 1, \dots, m$, se $E_i = \diamond A_i$ oppure $E_i = B_i \mathcal{U} A_i$:

$$f_{E_i} = \{n \in N \mid E_i \notin L_T(n) \text{ oppure } A_i \in L_T(n)\}$$

Dal tableau T si estrae l'automa \mathcal{A}_F considerando soltanto gli *stati* del tableau (nodi a cui è applicata la regola \circ). Ciascuno stato viene etichettato dalla congiunzione dei letterali del nodo corrispondente.

Più precisamente:

$$\mathcal{A}_F = (S, \Delta, I, L, 2^{2^P}, F)$$

dove:

Stati: $S = \{n \in N \mid n \text{ è espanso mediante la regola } \circ\}$

Stati iniziali:

$$I = \{n \in S \mid \text{esiste un cammino in } T \text{ da } n_0 \text{ a } n \text{ in cui } n \text{ è l'unico elemento di } S\}$$

In altri termini ogni stato iniziale di \mathcal{A}_F è uno stato raggiungibile nel tableau dalla radice senza passare per altri stati.

Relazione di transizione:

$$\Delta = \{(n_i, n_j) \mid n_i, n_j \in S \text{ e esiste un cammino in } T \text{ da } n_i \text{ a } n_j \text{ che non contiene altri elementi di } S\}$$

Cioè esiste un arco da n_i a n_j in \mathcal{A}_F se n_j è raggiungibile da n_i nel tableau senza passare per altri stati.

Etichette: l'etichetta di uno stato n di \mathcal{A}_F è la formula che si ottiene congiungendo i letterali presenti nell'etichetta di n in T :

$$L(n) = \ell_1 \wedge \dots \wedge \ell_k \text{ se } n \in N \text{ e } \ell_1, \dots, \ell_k \text{ sono tutti i letterali che occorrono in } L_T(n)$$

Ricordiamo che la congiunzione vuota è equivalente a \top , quindi se $k = 0$, allora $L(n) = \top$.

Stati di accettazione: Ciascun insieme di stati di accettazione di \mathcal{A}_F si ottiene da un insieme di nodi di accettazione di T conservandone soltanto gli stati: $F = \{f_1, \dots, f_m\}$, dove per ogni $i = 1, \dots, m$

$$f_i = (f_{E_i} \cap S)$$

Si noti che l'automa \mathcal{A}_F è etichettato da congiunzioni di letterali, e non da formule qualsiasi.

La relazione esistente tra la formula iniziale e l'automa generato dalla traduzione è la seguente: l'automa \mathcal{A}_F generato dalla traduzione della formula F è tale che $\mathcal{L}(\mathcal{A}_F)$ è l'insieme dei modelli di F . In altri termini ogni esecuzione accettante rappresenta (accetta) uno o più modelli di F , e, viceversa, ogni modello di F è accettato dall'automa. Di conseguenza una formula F è soddisfacibile sse $\mathcal{L}(\mathcal{A}_F) \neq \emptyset$.

In particolare, ogni esecuzione $\rho = s_0, s_1, s_2, \dots$ di un automa generato dall'algoritmo rappresenta un insieme di interpretazioni temporali: se le interpretazioni lette da ρ sono tutte le interpretazioni \mathcal{M} tali che, per ogni $i \in \mathbb{N}$, se $L(s_i) \models p$ (p è una conseguenza logica dell'etichetta di s_i , cioè è uno dei letterali congiunti in $L(s_i)$), allora $p \in \mathcal{M}(i)$, e se $L(s_i) \models \neg p$ allora $p \notin \mathcal{M}(i)$.

Si noti che la traduzione di formule in automi fornisce non soltanto un metodo per controllare la soddisfacibilità di formule LTL, ma anche un metodo per costruire un modello di una formula (se esiste): data una formula F , si costruisce \mathcal{A}_F ; se il linguaggio di \mathcal{A}_F non è vuoto, si sceglie un'esecuzione accettante ρ e una parola accettata da ρ : questa è un modello di F .

Per quel che riguarda la complessità della traduzione, osserviamo che il numero di nodi costruiti ed il tempo per costruirli è esponenziale nella dimensione della formula iniziale. L'esperienza mostra comunque che, nelle applicazioni alla verifica di sistemi, generalmente l'automa costruito è relativamente piccolo.

2.3.2 Esempi

Consideriamo ad esempio la formula $\Box \Diamond p$ e costruiamo l'automa $\mathcal{A}_{\Box \Diamond p}$, etichettato da congiunzioni di letterali nel linguaggio $P = \{p\}$. Nella figura 2.6 è rappresentato, in alto, un tableau completo per tale formula. Nel tableau non ci sono nodi contraddittori, quindi nessun nodo viene cancellato. I nodi 3 e 4 sono stati. I nodi di accettazione per $\Diamond p$ sono: $f_{\Diamond p} = \{1, 3\}$.

L'automa che rappresenta $\Box \Diamond p$ è

$$\mathcal{A}_{\Box \Diamond p} = (S, \Delta, I, L, 2^{2^P}, \{f_1\})$$

dove:

$$S = \{3, 4\}, \quad \Delta = S \times S, \quad I = \{3, 4\}, \quad f_1 = \{3\}, L(3) = p, \quad L(4) = \top$$

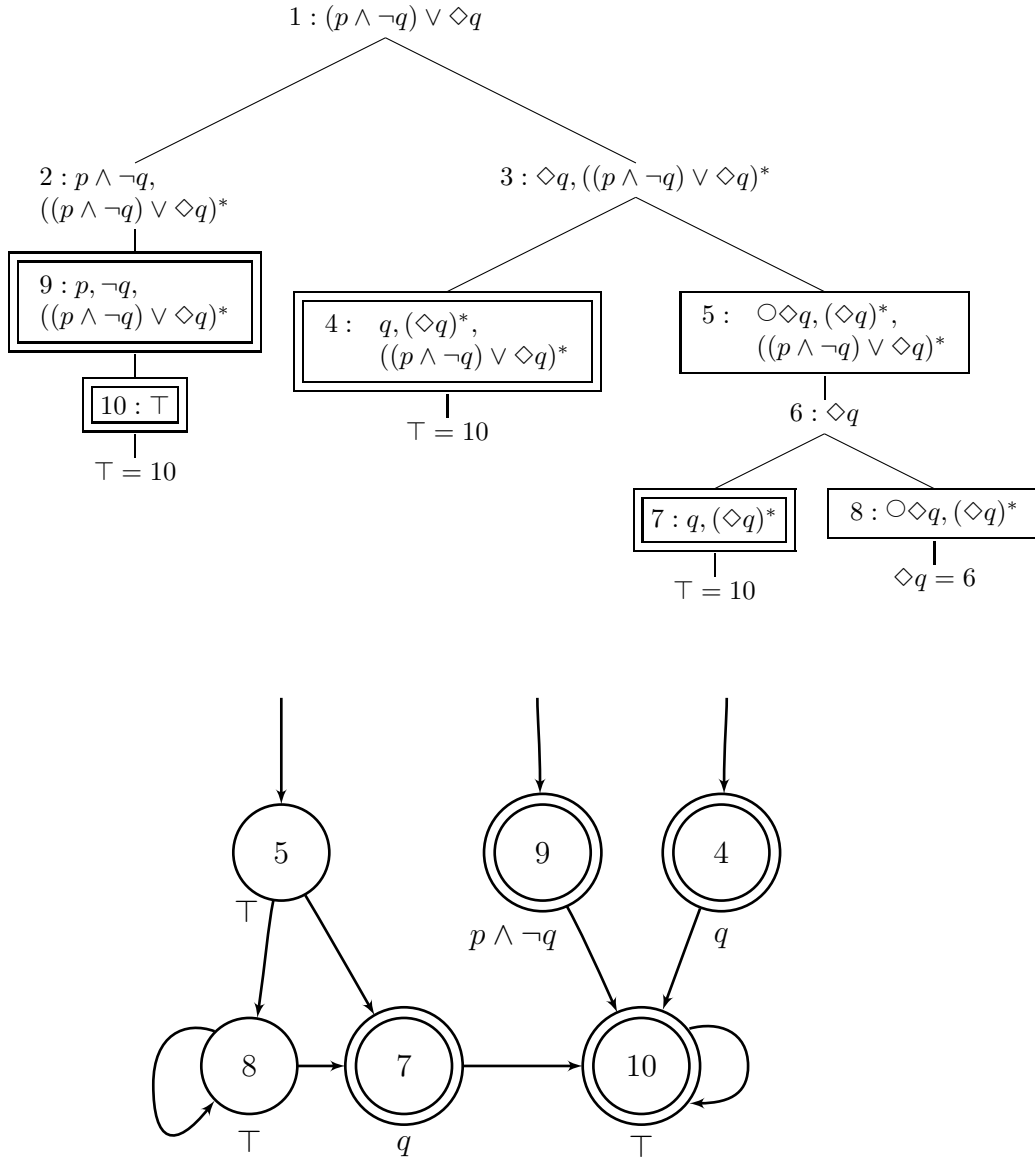
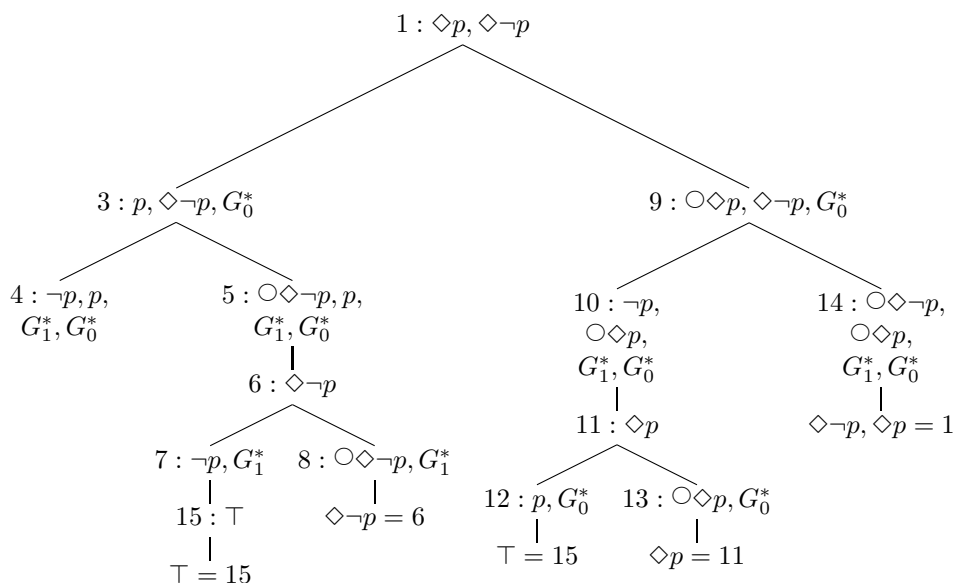


Figura 2.7: Un tableau completo per $(p \wedge \neg q) \vee \diamond q$ e l'automata che rappresenta tale formula

Figura 2.8: Un tableau completo per $\diamond p \wedge \diamond \neg p$

indichiamo con

$$\begin{aligned} G_0 & \text{ la formula } \diamond p \\ G_1 & \text{ la formula } \diamond \neg p \end{aligned}$$

Dopo aver cancellato il nodo 4, che è contraddittorio, si ottiene il tableau della figura 2.9, dove gli stati sono evidenziati in un rettangolo, gli stati di accettazione di $\diamond p$ in un doppio rettangolo e quelli di $\diamond \neg p$ in un rettangolo a bordo spesso. Gli stati di accettazione di entrambe le eventualities hanno un rettangolo a bordo spesso ed un doppio rettangolo esterno. L'automa che si ottiene è rappresentato nella figura 2.10. Qui gli stati di accettazione di $\diamond p$ sono evidenziati con un doppio rettangolo, quelli di $\diamond \neg p$ con un doppio cerchio, e quelli di entrambi con un doppio ottagono. L'insieme degli insiemi di stati di accettazione di tale automa è $F = \{f_1, f_2\}$, dove $f_1 = \{5, 7, 8, 12, 15\}$ e $f_2 = \{7, 10, 12, 13, 15\}$.

2.3.3 Un algoritmo per la costruzione dell'automa corrispondente a una formula

Nei sistemi che implementano il model checking per LTL, la costruzione dell'automa \mathcal{A}_F corrispondente alla formula F non passa normalmente per la costruzione del tableau per poi eliminarne alcuni nodi, ma si costruisce direttamente l'automa. Sarebbe infatti inutile memorizzare tutti i nodi del tableau, per poi cancellare quelli che non interessano.

Inoltre, il loop checking viene effettuato soltanto su *stati*, dato che i nodi intermedi non sono conservati in memoria. Ciò può a volte portare a proseguire la costruzione quando si potrebbe terminare prima confrontando le etichette di nodi intermedi. Ad esempio, nel tableau della figura 2.6, il figlio del nodo 5 non viene costruito, ma viene aggiunto un link al nodo 2. Nell'algoritmo che

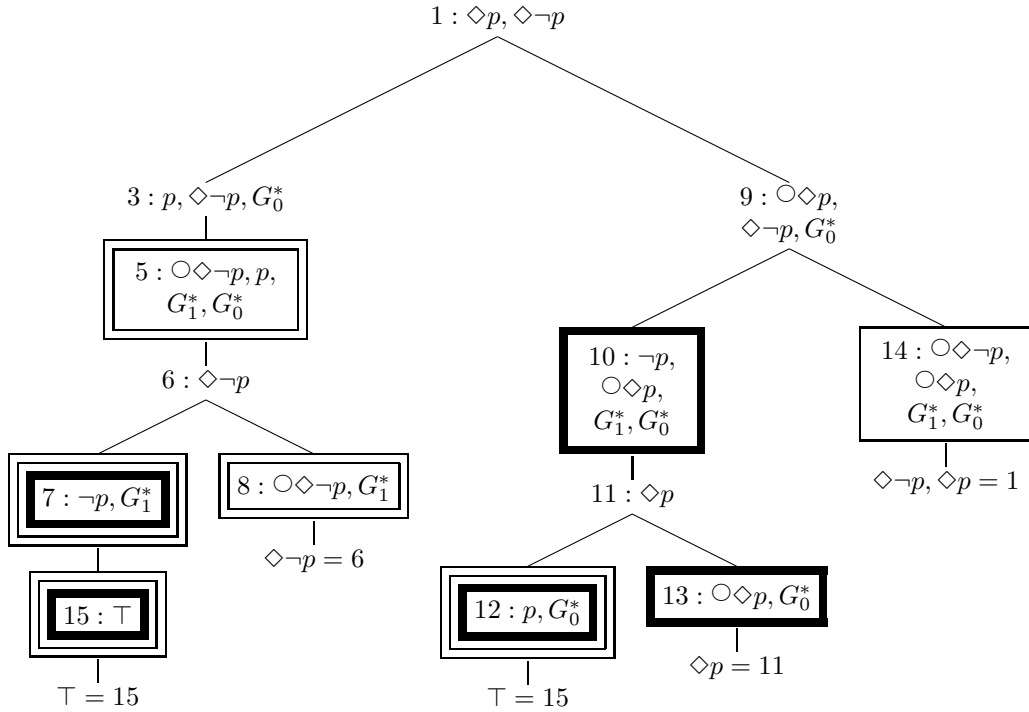


Figura 2.9: Un tableau completo per $\diamond p \wedge \diamond \neg p$, dopo la cancellazione dei nodi contraddittori, e l'automa ad esso corrispondente

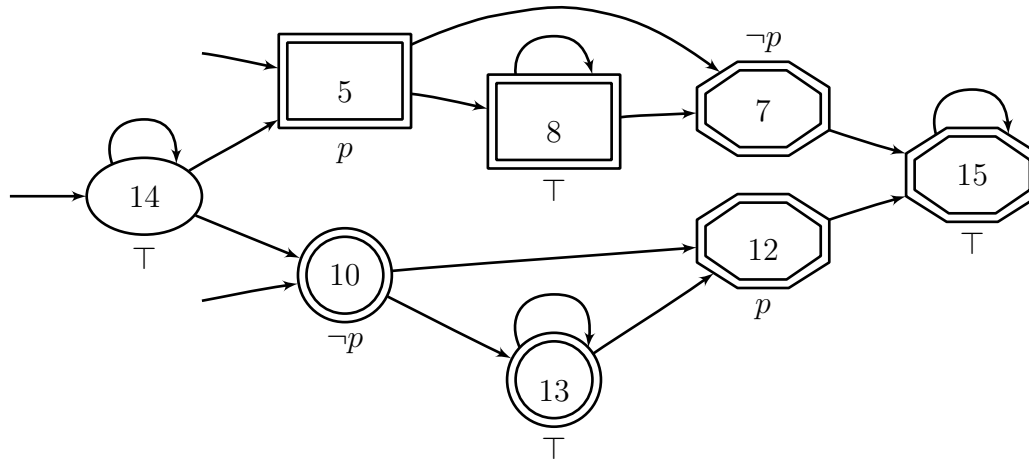


Figura 2.10: L'automa corrispondente al tableau della figura 2.9

descriviamo, il nodo 2 non è memorizzato, quindi il figlio del nodo 5 dovrà ancora essere espanso fino ad arrivare ad avere stati.

Le implementazioni di algoritmi per la costruzione di automi da formule LTL rappresentano i nodi mediante strutture modificabili. Quando un nodo viene espanso mediante una regola statica che non ramifica (che genererebbe cioè un unico figlio), il nodo stesso viene modificato. Solo quando il nodo viene espanso mediante una regola che ramifica, allora viene creata una copia del nodo (insieme a tutti gli archi entranti); la copia originaria viene modificata per diventare il figlio sinistro, la nuova copia diventa il figlio destro. Quando a un nodo N non si possono più applicare regole statiche (il nodo è ora uno stato), viene eseguito il controllo di cicli: se già esiste un nodo N' con la stessa etichetta di N , gli archi entranti in N sono aggiunti a quelli di N' e N viene cancellato. Altrimenti, si crea un nuovo nodo figlio di N che rappresenta la sua espansione tramite la regola \circ .

Più in dettaglio, possiamo descrivere come segue un algoritmo di generazione di un automa da una formula. Un *nodo* è una struttura dati che contiene i seguenti campi:

Name: un identificatore unico per il nodo;

Incoming: una lista degli identificatori dei nodi che hanno archi entranti nel nodo. I nodi iniziali hanno in questo campo l'identificatore *init*.

New, Literals, Old, Next: insiemi di sottoformule della formula iniziale, o sottoformule precedute da \circ .

New contiene le formule ancora da espandere, o da prendere comunque in considerazione. Esse verranno via via estratte da questo insieme e spostate negli altri: i letterali vengono spostati in *Literals*, le formule espanse in *Old* (quindi *Old* contiene le formule marcate); quando poi viene estratta una formula della forma $\circ A$, la formula A viene inserita nell'insieme *Next*.

Se N è un nodo, $Name(N)$, $Incoming(N)$, $New(N)$, $Literals(N)$, $Old(N)$ e $Next(N)$ sono i valori dei rispettivi campi di N .

L'algoritmo inizializza opportunamente il grafo e prosegue poi con il ciclo di espansione dei nodi. Ogni volta che viene costruito un nuovo nodo, gli viene assegnato un nome nuovo (nel campo *Name*).

Inizializzazione: si crea il nodo iniziale N_0 con:

$New(N_0) =$ insieme iniziale di formule

$Incoming(N_0) = \{init\}$

$Literals(N_0) = Old(N_0) = Next(N_0) = \emptyset$

Il ciclo di espansione dei nodi, descritto di seguito, prosegue finché è possibile espandere qualche nodo.

Per ogni nodo N :

- finché il campo $New(N)$ non è vuoto, si sceglie una formula A di $New(N)$, si cancella da $New(N)$ e:
 - Se A è un letterale allora, se il campo $Literals(N)$ contiene il letterale complementare, allora il nodo N viene cancellato, altrimenti A si inserisce in $Literals(N)$.

- Se A non è un letterale, si inserisce A in $Old(N)$, e:
 - * Se A si espande con una regola che non ramifica, le sue espansioni vengono aggiunte a $New(N)$.
 - * Se A si espande con una regola che ramifica, si crea una copia N' del nodo N con un nome nuovo (copiandone tutti gli altri campi, anche $Incoming(N)$), al campo $New(N)$ si aggiunge un'espansione di A , al campo $New(N')$ si aggiunge l'altra espansione di A .
 - * Se $A = \circ A'$, allora A' viene aggiunta al campo $Next(N)$.
- Quando il campo $New(N)$ è vuoto:
 - se il grafo già contiene un nodo N' con $Literals(N) = Literals(N')$, $Old(N') = Old(N)$ e $Next(N') = Next(N)$, si aggiungono gli elementi del campo $Incoming(N)$ al campo $Incoming(N')$ e N viene cancellato.
 - Altrimenti si crea un nuovo nodo N' con:

$$\begin{aligned} New(N') &= Next(N) \\ Incoming(N') &= \{Name(N)\} \\ Literals(N') &= Old(N') = Next(N') = \emptyset \end{aligned}$$

L'automa generato da tale algoritmo di traduzione è il seguente:

Alfabeto Σ : congiunzioni di letterali (atomi e negazioni di atomi in P)

Stati S : insieme dei nodi del grafo ottenuto mediante la costruzione sopra descritta.

Relazione di transizione $\Delta = \{(N, N') \mid Name(N) \in Incoming(N')\}$

Stati iniziali $I = \{N \mid init \in Incoming(N)\}$

Etichette: per ogni nodo N , se $Literals(N) = \{\ell_1, \dots, \ell_m\}$ per $m > 0$, allora $L(N) = \ell_1 \wedge \dots \wedge \ell_m$; altrimenti, se $Literals(N) = \emptyset$, $L(N) = \top$.

Stati di accettazione: siano E_1, \dots, E_k tutte le eventualities che sono sottoformule dell'insieme iniziale, dove per ogni $i = 1, \dots, k$: $E_i = \diamond A_i$ o $E_i = B_i \mathcal{U} A_i$. Allora $F = \{f_1, \dots, f_k\}$, dove per ogni $i = 1, \dots, k$:

$$f_i = \{N \mid E_i \notin Old(N) \text{ oppure } A_i \in Old(N) \cup Literals(N)\}$$

2.3.4 Verifica in teoria degli automi

La verifica formale di sistemi basata sulla teoria degli automi si basa sui passi seguenti:

1. Il sistema viene modellato mediante un BA \mathcal{A} etichettato da formule: se $\mathcal{A}' = (S, \Delta, I, L', 2^P, F)$ è l'automa etichettato da insiemi di atomi che rappresenta il sistema, allora $\mathcal{A} = (S, \Delta, I, L, 2^{2^P}, F)$ dove

$$L(s) = \bigwedge_{p \in L'(s)} p \wedge \bigwedge_{p \in P - L'(s)} \neg p$$

Qui, se S è un insieme di formule, $\bigwedge_{A \in S} A$ denota la congiunzione di tutte le formule in S .

Ad esempio, se $P = \{p, q, r\}$ e $L'(s) = \{p, r\}$, allora $L(s) = p \wedge r \wedge \neg q$.

2. Sia F la specifica che si vuole verificare per \mathcal{A} . F può essere rappresentata da un automa \mathcal{B}_F etichettato da formule, e se ne può poi costruire il complemento. Ma la costruzione del complemento di un automa è difficile e costosa.

Si costruisce allora direttamente l'automata $\mathcal{B}_{\neg F} = \overline{\mathcal{B}_F}$, che rappresenta la negazione della specifica $\neg F$.

3. Si costruisce il GBA intersezione

$$\mathcal{G} = \mathcal{A} \cap \mathcal{B}_{\neg F}$$

4. Si controlla se il linguaggio di \mathcal{G} è vuoto oppure no: il sistema soddisfa la specifica sse $\mathcal{L}(\mathcal{G}) = \emptyset$

Il controllo $\mathcal{L}(\mathcal{G}) = \emptyset$ consente di fornire un controesempio in caso di risposta negativa. Sia $\mathcal{G} = \mathcal{A} \cap \mathcal{B}_{\neg F} = (\Sigma, S, \Delta, I, L, F)$ l'automata intersezione di \mathcal{A} e $\mathcal{B}_{\neg F}$. Assumiamo che tutte le etichette in \mathcal{G} siano diverse da \perp : se qualche stato di \mathcal{G} è equivalente a \perp , viene cancellato.

Si noti che controllare se $L(s) \leftrightarrow \perp$ è un problema NP-completo. Tuttavia l'automata \mathcal{A} è etichettato da congiunzioni di letterali e tutte le etichette dell'automata $\mathcal{B}_{\neg F}$, che rappresenta $\neg F$, sono congiunzioni di letterali; poiché la costruzione dell'automata intersezione conserva questa proprietà, anche le etichette di \mathcal{G} sono congiunzioni di letterali. Ora, se A è una congiunzione di letterali, controllare se $A \leftrightarrow \top$ richiede tempo lineare.

Per controllare se $\mathcal{L}(\mathcal{G}) = \emptyset$ si applica la seguente osservazione: se \mathcal{G} è un GBA, con insiemi di accettazione $F = \{f_1, \dots, f_m\}$, allora $\mathcal{L}(\mathcal{G}) \neq \emptyset$ se e solo se esiste un ciclo di stati $s_1, s_2, \dots, s_k, s_1$ (una SSC $\{s_1, s_2, \dots, s_k\}$), raggiungibile da uno stato iniziale, che contiene uno stato di ciascun f_i :

$$\{s_1, s_2, \dots, s_k\} \cap f_i \neq \emptyset \text{ per ogni } i = 1, \dots, m$$

Per trovare le componenti fortemente connesse si può utilizzare una visita in profondità. Se $\mathcal{L}(\mathcal{G}) \neq \emptyset$ la visita produce un'esecuzione ρ in $\mathcal{L}(\mathcal{G})$ che si può rappresentare in modo finito: ρ è costituita da un prefisso σ_1 finito, seguito da una sequenza periodica di stati:

$$\rho = \sigma_1 (\sigma_2)^\omega \text{ con } \sigma_1, \sigma_2 \text{ finite}$$

(ρ è una sequenza *ultimately periodic*).

Come conseguenza, abbiamo il seguente risultato: se una formula F è soddisfacibile, allora F ha un modello *ultimately periodic*, costituito cioè da una sequenza di interpretazioni proposizionali che, dopo un segmento iniziale finito, ripete ciclicamente un altro segmento finito di interpretazioni proposizionali. Un tale modello si può rappresentare in modo finito: $\mathcal{M}(0), \mathcal{M}(1), \dots, \mathcal{M}(n), \dots, \mathcal{M}(n+k) = \mathcal{M}(n)$.

2.4 Esercizi

1. Sia $\mathcal{A} = \langle S, \Delta, I, L, \Sigma, \{f_1, f_2\} \rangle$ l'automa di Büchi generalizzato dove: $S = \{s_1, s_2, s_3\}$, $\Delta = \{(s_1, s_1), (s_1, s_2), (s_2, s_3), (s_3, s_1), (s_3, s_2)\}$, $I = \{s_1, s_3\}$, $L(s_1) = \{p\}$, $L(s_2) = \{p, q\}$, $L(s_3) = \{r\}$, $\Sigma = 2^{\{p, q, r\}}$, $f_1 = \{s_2\}$, $f_2 = \{s_3\}$.

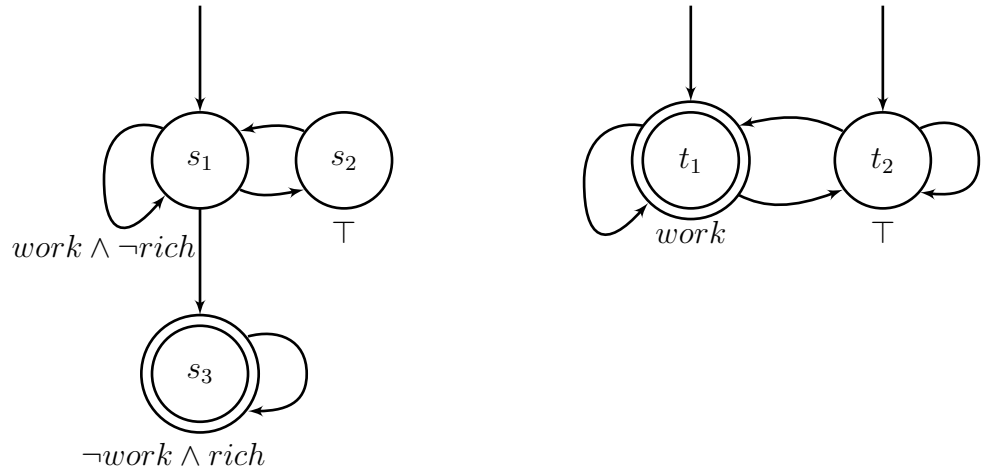
Dare una rappresentazione grafica di \mathcal{A} e costruire un automa semplice \mathcal{A}^* equivalente a \mathcal{A} . Definire le diverse componenti di \mathcal{A}^* (insieme degli stati, delle transizioni, ecc.) e darne una rappresentazione grafica.

Che cosa significa che l'automa \mathcal{A}^* è *equivalente* ad \mathcal{A} ?

2. Si consideri lo stesso automa del punto precedente. Rappresentare graficamente un automa equivalente etichettato da formule.
3. Sia $P = \{p, q\}$ e Σ l'insieme di tutti gli insiemi di assegnazioni proposizionali per P ($\Sigma = 2^{2^P}$) – si ricordi che gli insiemi di assegnazioni possono essere rappresentati da formule proposizionali. Siano inoltre $\mathcal{A}_0 = (S_0, \Delta_0, I_0, L_0, \Sigma, F_0)$ e $\mathcal{A}_1 = (S_1, \Delta_1, I_1, L_1, \Sigma, F_1)$ due automi sullo stesso alfabeto Σ , dove:

$$\begin{array}{ll} S_0 = \{a, b\} & S_1 = \{1, 2, 3\} \\ \Delta_0 = \{(a, b), (b, b)\} & \Delta_1 = \{(1, 1), (1, 2), (2, 1), (2, 3), (3, 3)\} \\ I_0 = \{a\} & I_1 = \{1, 2\} \\ F_0 = \{b\} & F_1 = \{2, 3\} \\ L_0(a) = p \vee q, L_0(b) = \neg p & L_1(1) = \neg p, L_1(2) = q, L_1(3) = p \end{array}$$

- (a) Rappresentare graficamente gli automi \mathcal{A}_0 e \mathcal{A}_1 ; costruire l'automa intersezione $\mathcal{B} = (S, \Delta, I, L, \Sigma, F)$ di \mathcal{A}_0 e \mathcal{A}_1 e darne una rappresentazione grafica. Inoltre, se il linguaggio di \mathcal{B} non è vuoto, fornire un esempio di parola accettata da \mathcal{B} .
 - (b) Trasformare l'automa di Büchi generalizzato \mathcal{B} in un automa semplice equivalente.
 - (c) Ricordando che automi etichettati da formule (nel linguaggio P) si possono considerare rappresentazioni compatte di automi sull'alfabeto $\Sigma' = 2^P$ (insieme di tutti i sottoinsiemi di P), rappresentare graficamente l'automa sull'alfabeto Σ' corrispondente a \mathcal{A}_0 .
4. Si considerino i due automi \mathcal{A}_1 e \mathcal{A}_2 sotto rappresentati:



- (a) Costruire l'intersezione $\mathcal{A} = \mathcal{A}_1 \cap \mathcal{A}_2$ dei due automi;
 - (b) Quale formula temporale rappresenta l'automa \mathcal{A}_2 (di destra)?
 - (c) Il linguaggio di \mathcal{A} è vuoto? Quale specifica si può dunque dire che sia soddisfatta o non soddisfatta dal sistema rappresentato dall'automa \mathcal{A}_1 ?
 - (d) Trasformare l'automa generalizzato \mathcal{A} in un automa semplice equivalente \mathcal{A}' .
 - (e) Il linguaggio di \mathcal{A}' è vuoto?
 - (f) Considerare l'automa \mathcal{A}'_1 che è come \mathcal{A}_1 per tutte le componenti, tranne che non vi sono vincoli di fairness ($F = \{s1, s2, s3\}$). Costruire $\mathcal{A} = \mathcal{A}'_1 \cap \mathcal{A}_2$ e verificare se il suo linguaggio è vuoto o no.
5. Si consideri il sistema costituito da un trapano che deve operare su un pezzo di legno. Prima e dopo la lavorazione, il trapano deve restare in uno stato di inattività per un certo tempo. La lavorazione e le fasi di inattività possono durare per un tempo indefinito. Inizialmente il trapano non è attivo e, dopo aver terminato la lavorazione, resta inattivo per sempre.
- (a) Definire un automa \mathcal{A} , etichettato da interpretazioni proposizionali, che rappresenti il sistema, sul linguaggio proposizionale $P = \{working, finished\}$ (*working* rappresenta il fatto che il pezzo è in lavorazione, *finished* che la lavorazione è terminata).
 - (b) Scrivere una formula della logica temporale lineare che rappresenti la proprietà seguente: in ogni stato in cui la lavorazione non è terminata, esiste uno stato futuro in cui il pezzo è sotto lavorazione.
 - (c) Verificare se il sistema soddisfa la proprietà del punto precedente. Nel caso in cui non la soddisfi, determinare un'esecuzione del sistema che non la soddisfa.
 - (d) Aggiungere al sistema un vincolo di fairness che garantisca che il pezzo prima o poi venga lavorato, e verificare se il nuovo sistema soddisfa ora la specifica.
6. Si consideri la formula $F = \Box(p \rightarrow \Diamond q)$.

- (a) Sviluppare un tableau completo per F e caratterizzare l'insieme dei cammini aperti del tableau.
- (b) Determinare un automa di Büchi che accetti tutti e soli i modelli di F . Rappresentare graficamente l'automa, evidenziando l'insieme degli stati di accettazione.
- (c) Scegliere un'esecuzione accettante ρ dell'automa e definire una parola accettata da ρ . Verificare che tale parola è un modello di F .