Traduzione di LTL in GBA (I)

Ogni formula G di LTL può essere "tradotta" in un automa $\mathcal{A}_G$ che accetta esattamente i modelli di G:

$$\mathcal{L}(\mathcal{A}_G) = \{\mathcal{M} \mid \mathcal{M} \models G\}$$

Step 1: costruzione di un tableau per G

Si costruisce un tableau $T = (N, \Delta_T, \{n_0\}, L_T, \Sigma_T, F_T)$ per la formula G da tradurre (in cui sono eventualmente stati cancellati i nodi chiusi), dove:

- N è l'insieme dei nodi,
- $n_0 \in N$ è la radice,
- Δ_T è la relazione padre-figlio,
- Σ_T è l'alfabeto:

$$\Sigma_T = 2^{subf(G) \cup \{\bigcirc B \mid B \in subf(G)\}}$$

- $L_T : N \to \Sigma$ è la funzione di etichettatura dei nodi;
- $F_T = \{f_{E_1}, ..., f_{E_m}\}$, dove $E_1, ..., E_m$ sono tutte le eventualities in $subf(G)$ e per ogni $i = 1, ..., m$, se $E_i = \Diamond A_i$

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

Cancellazione dei nodi “contraddittori”

Dal tableau per la formula si cancellano tutti i nodi contraddittori.

Iterativamente:

- Se l’etichetta di un nodo *N* contiene un atomo e la sua negazione, *N* viene cancellato.
- Se sono stati cancellati **tutti i figli** di un nodo *N*, il nodo *N* viene cancellato.

Traduzione di LTL in GBA (II)

Step 2: dal tableau si estrae l'automa che rappresenta G

Da $T = (N, \Delta_T, \{n_0\}, L_T, \Sigma_T, F_T)$ si estrae il **grafo degli stati** (nodi a cui è applicata la regola NEXT), ciascuno dei quali è etichettato dalla congiunzione dei letterali del nodo corrispondente.

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

Stati:

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

Stati iniziali:

$$I = \{n \in S \mid \text{esiste un cammino in } T \text{ da } n_0 \text{ a } n \text{ in cui } n \text{ è l'unico elemento di } S \text{ (nel cammino non è mai applicata la regola NEXT)}\}$$

Relazione di transizione:

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

Traduzione di LTL in GBA (III)

Eestrazione dell'automa dal tableau (segue)

Etichette:

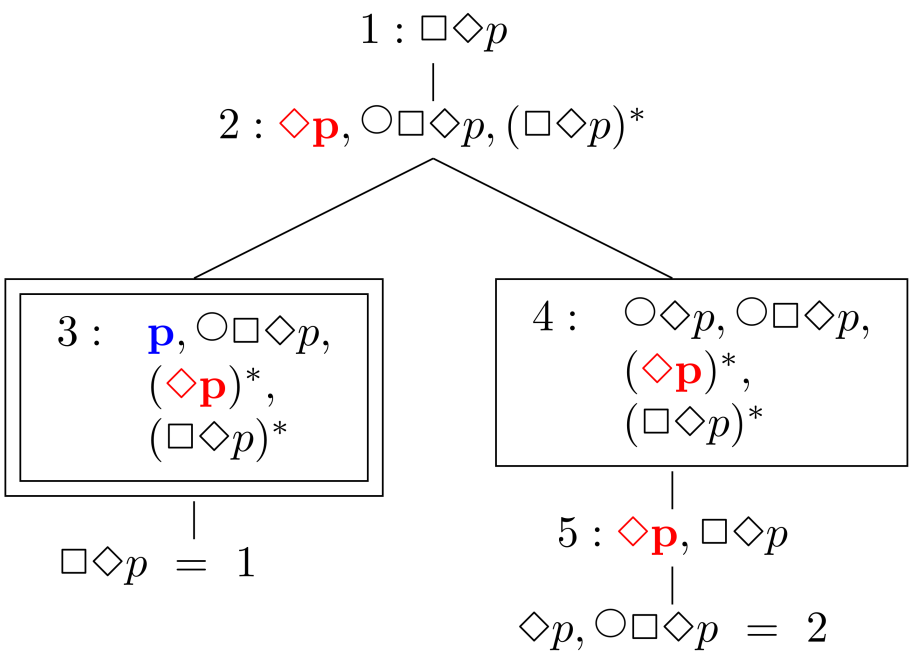
$L(n) = \ell_1 \wedge ... \wedge \ell_k$ dove $\ell_1, ..., \ell_k$ sono tutti i letterali che occorrono in $L_T(n)$
(se $k = 0$ allora $L(n) = \top$)

Stati di accettazione: $F = \{f_1, ..., f_m\}$, dove per ogni $i = 1, ..., m$

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

Si ottiene un GBA etichettato da congiunzioni di letterali

Esempio 1



$P = \{p\}$.

Non ci sono nodi chiusi da cancellare.

3 e 4 sono stati.

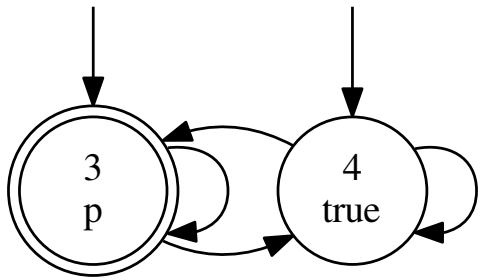
Nodi di accettazione per $\Diamond p$: $\{1, 3\}$

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

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

Esempio 1: semplificazione dell'automa

L'automa $\mathcal{A}_{\Box\Diamond p} = (S, \Delta, I, L, 2^{2^P}, \{f_1\})$ si può semplificare nel BA in cui c'è un unico insieme di stati di accettazione: $F = \{3\}$.

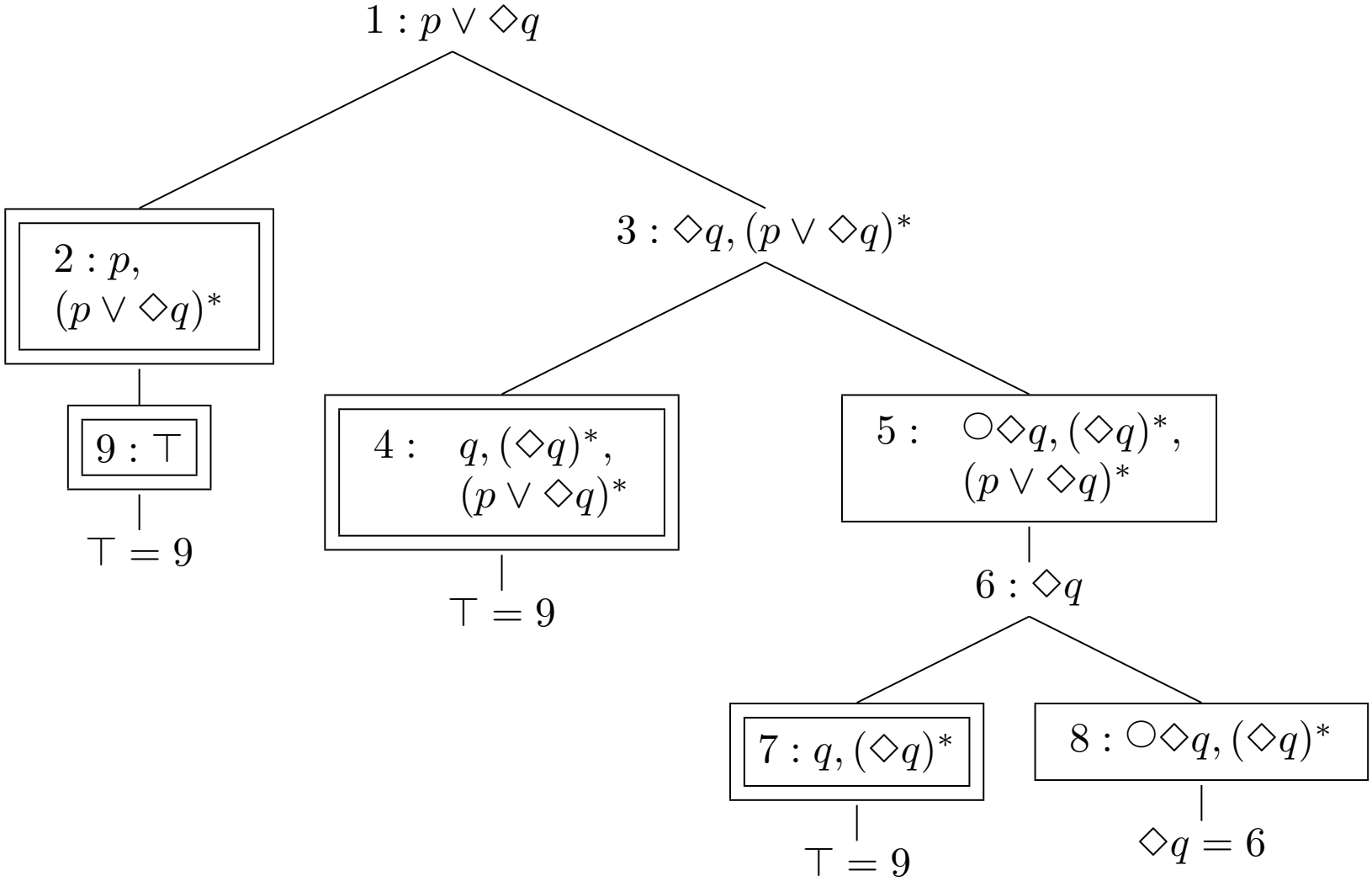


Semplificazione: se tra le sottoformule della formula iniziale c'è una sola eventuality E, allora si costruisce direttamente un BA con $F = f_E$.

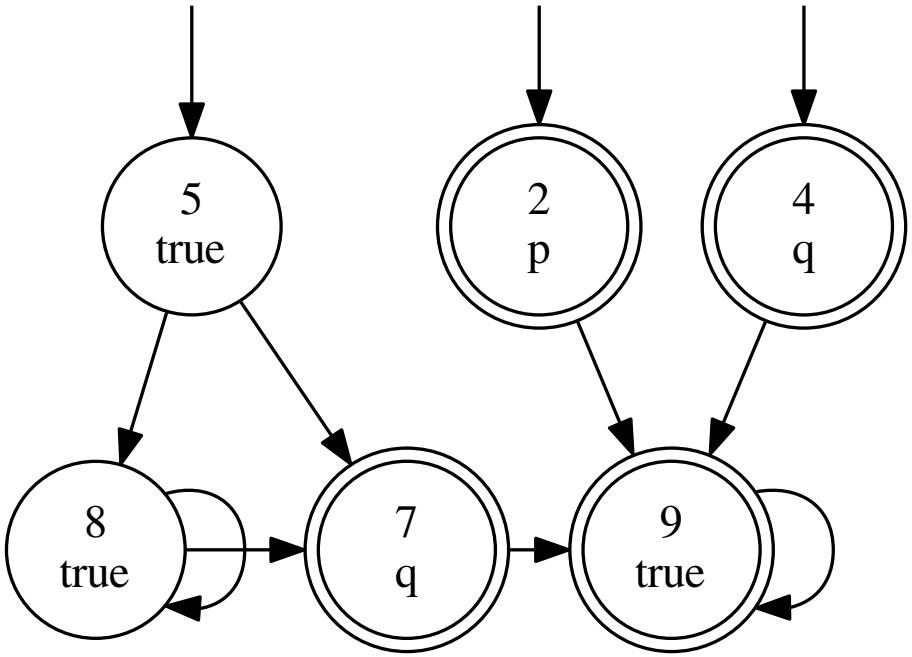
Nota: se la formula iniziale non contiene eventualities, allora tutti gli stati sono stati di accettazione dell'automa.

Esempio 2

Costruendo un tableau manualmente, si può effettuare il controllo di cicli anche su nodi generati dall'applicazione della regola NEXT (che non contengono formule marcate)

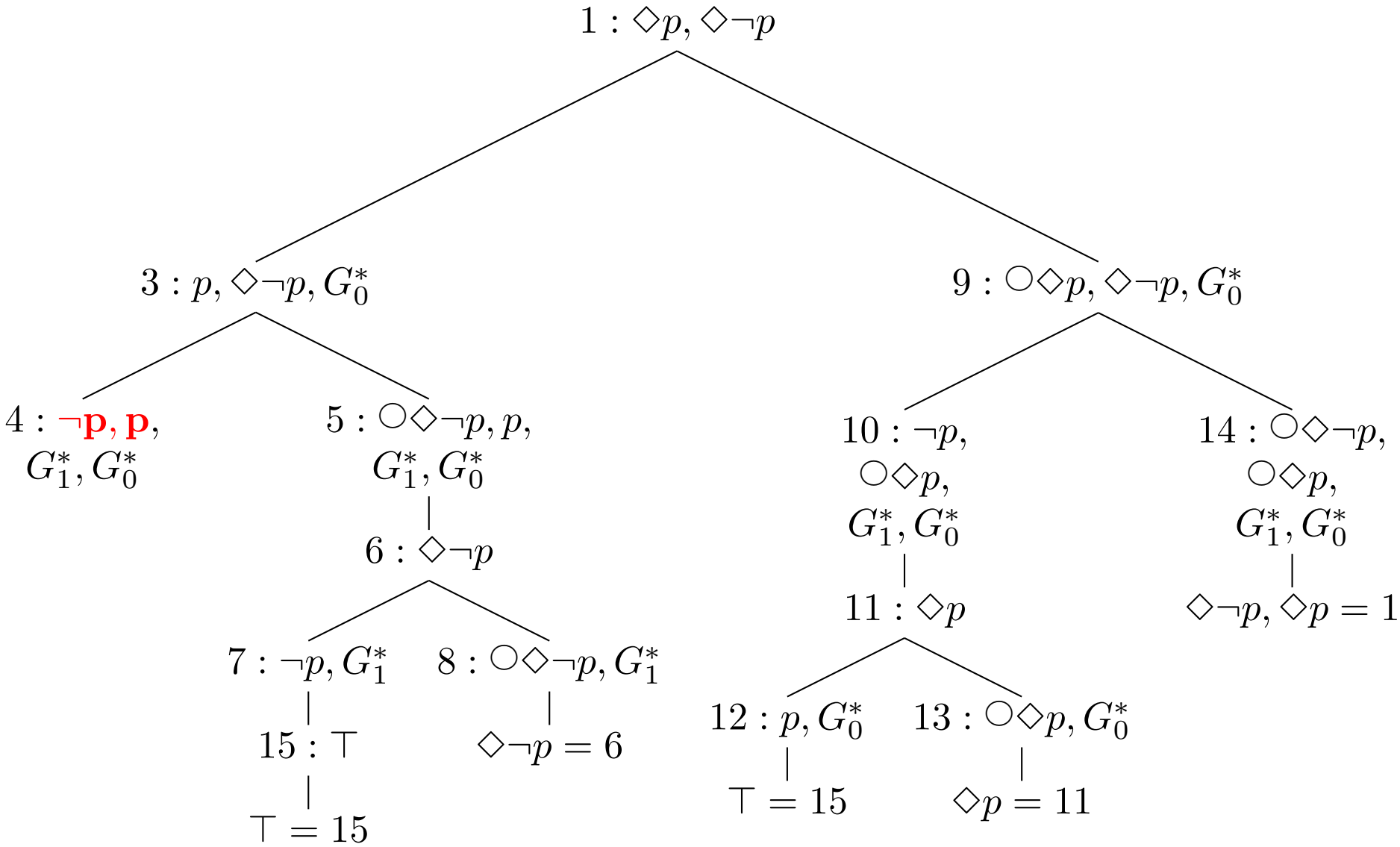


Esempio 2: l'automa

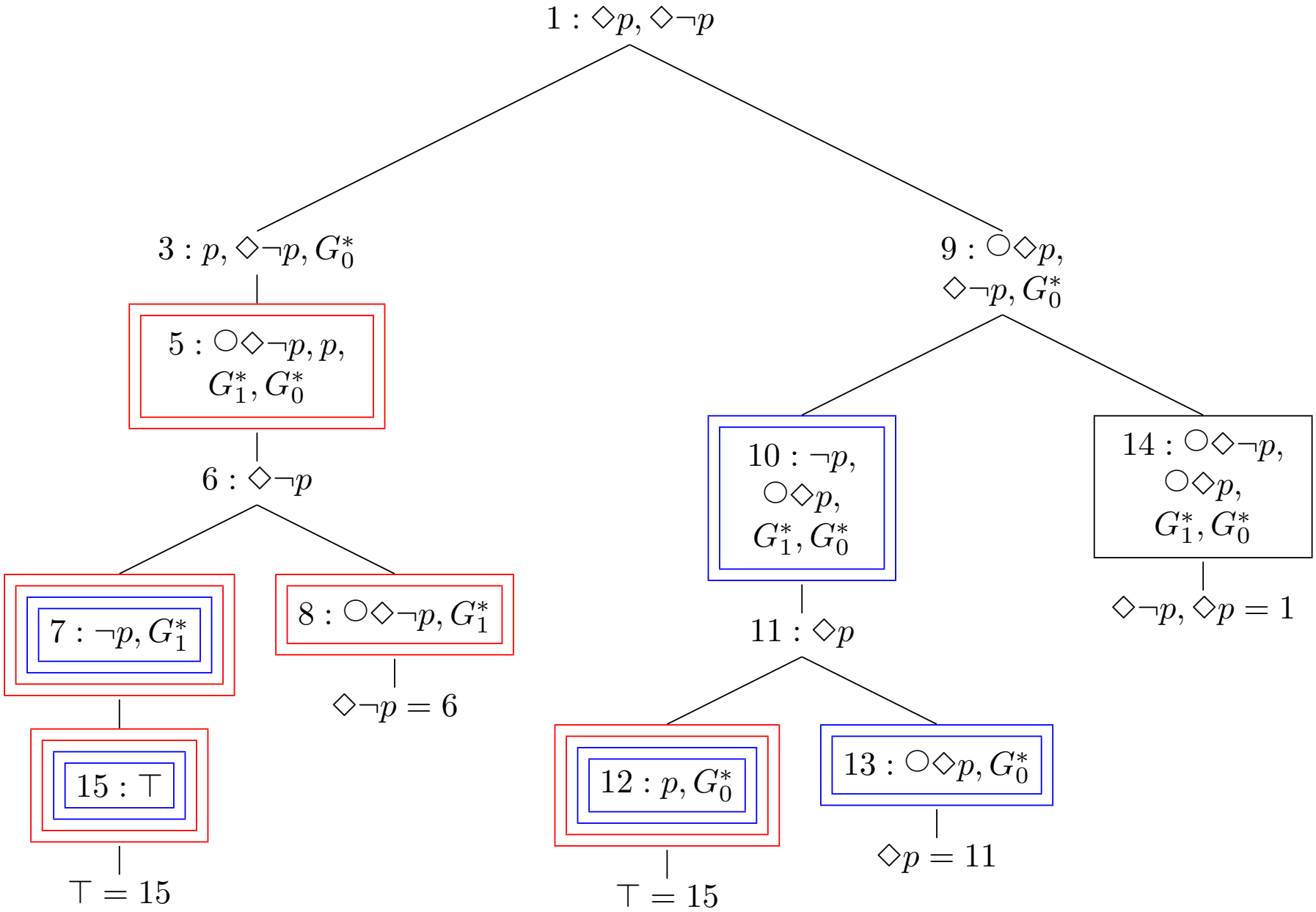


Esempio 3

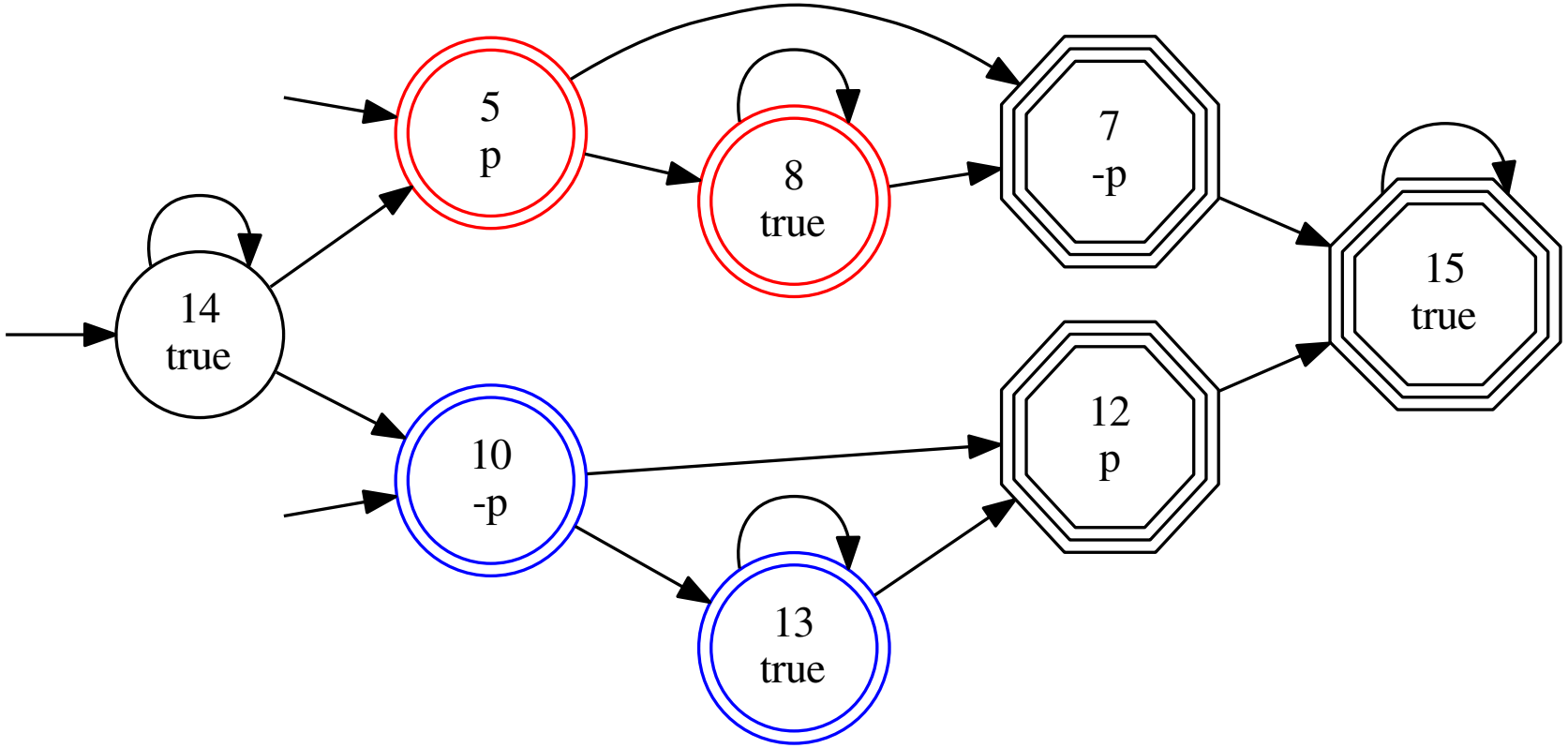
$G_0 = \Diamond p$
$G_1 = \Diamond \neg p$



Dopo aver cancellato i nodi chiusi



Esempio 3: l'automa



$F = \{f_1, f_2\}$, con $f_1 = \{5, 7, 8, 12, 15\}$ e $f_2 = \{7, 10, 12, 13, 15\}$

Algoritmo (vedi dettagli sulle dispense)

La costruzione automatica dell'automa $\mathcal{A}_G$ a partire dalla formula G avviene in genere senza passare per la costruzione intermedia del tableau:

- non vengono memorizzati tutti i nodi del tableau, per poi cancellare quelli che non interessano;
- ogni nodo è una struttura dati con diverse componenti:
  - un identificatore unico;
  - l'insieme degli archi entranti nel nodo;
  - l'insieme delle formule che devono essere ancora espanse;
  - l'insieme dei letterali nel nodo;
  - l'insieme delle formule marcate;
  - l'insieme delle formule della forma $\bigcirc A$
- I nodi vengono modificati via via che si procede nella loro espansione (ad esempio, quando si espande $A \wedge B$, si aggiungono al campo "formule da espandere" le formule A e B, e $A \wedge B$ viene spostata dal campo "formule da espandere" al campo "formule marcate").
- Un nodo viene "duplicato" solo quando gli si applica una regola che fa ramificare.
- Il controllo di cicli viene effettuato soltanto quando termina l'espansione statica di un nodo

Relazione tra la formula iniziale e l'automa generato dalla traduzione

L'automa $\mathcal{A}_G$ generato dalla traduzione della formula G è tale che:

$\mathcal{L}(\mathcal{A}_G)$ è l'insieme dei modelli di G

(ogni accepting run rappresenta un modello di G, e viceversa).

Quindi **G è soddisfacibile sse $\mathcal{L}(\mathcal{A}_G) \neq \emptyset$.**

Complessità della traduzione

Il numero di nodi costruiti ed il tempo per costruirli è esponenziale nella dimensione della formula iniziale.

L'esperienza mostra comunque che generalmente l'automa costruito è relativamente piccolo.

Esecuzioni e interpretazioni temporali

Sia G una formula e $\mathcal{A}_G$ l'automa che rappresenta G.

Ogni esecuzione $\rho = s_0, s_1, s_2, ...$ di $\mathcal{A}_G$ rappresenta un insieme di interpretazioni temporali: tutte le interpretazioni $\mathcal{M}$ tali che

- per ogni $i \in \mathbb{N}$: $\mathcal{M}_i \models L(s_i)$.

In altri termini, se $Literals(s_i)$ è l'insieme di letterali che etichettano in nodo del tableau corrispondente a s_i, allora l'esecuzione ρ rappresenta le interpretazioni $\mathcal{M}$ tali che per ogni $i \in \mathbb{N}$:

- se $p \in Literals(N_i)$ allora $p \in \mathcal{M}(i)$,
- se $\neg p \in Literals(N_i)$ allora $p \notin \mathcal{M}(i)$.

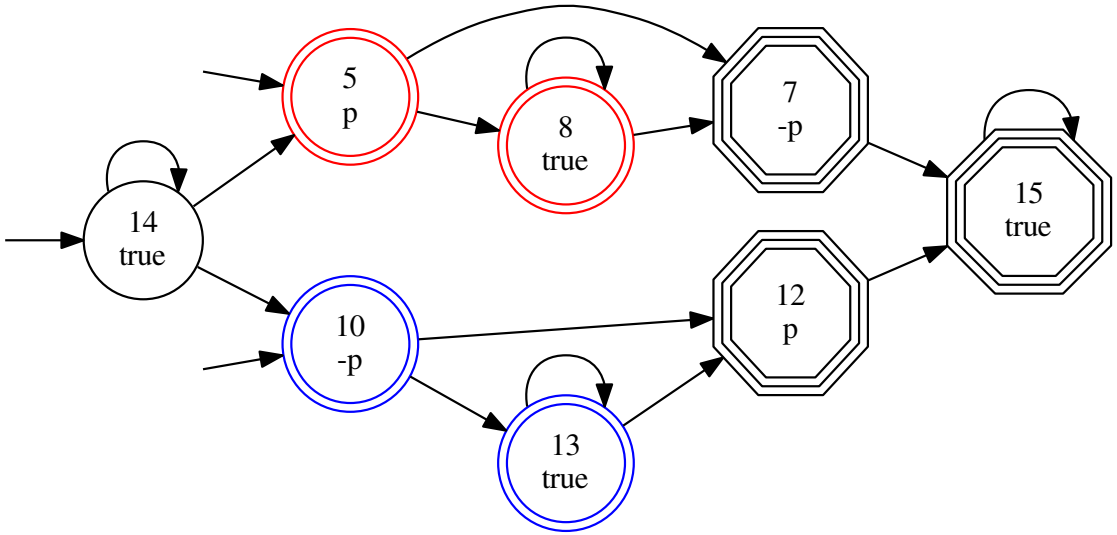
In particolare, un'interpretazione rappresentata da ρ è l'interpretazione $\mathcal{M}$, con:

$$\mathcal{M}(i) = Literals(N_i) \cap P$$

(per ogni $p \in P$: $p \in \mathcal{M}(i)$ sse $p \in Literals(N_i)$).

Se ρ è un'esecuzione di accettazione di $\mathcal{A}_G$, allora ogni interpretazione rappresentata da ρ è un modello di G.

Esempio: $\mathcal{A}_{\Diamond p \wedge \Diamond \neg p}$

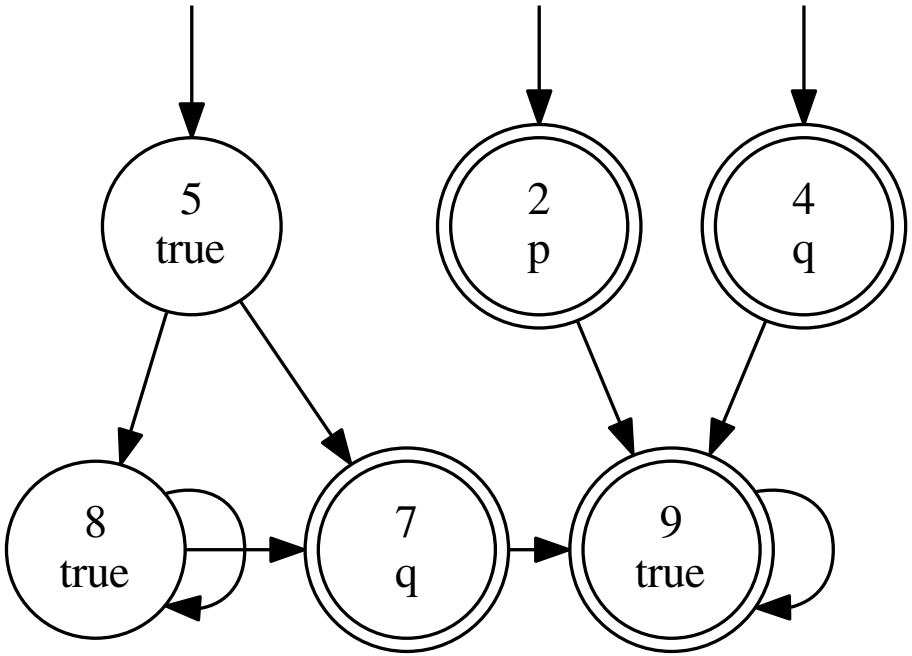


$\rho = \mathbf{14, 14, 5, 8, 8, 7, 15^{\omega}}$ è un'esecuzione di accettazione. Rappresenta tutte le interpretazioni $\mathcal{M}$ tali che $\mathcal{M}_2 \models p$ e $\mathcal{M}_5 \models \neg p$.

Ad esempio: $\mathcal{M}(0) = \{p\},\ \mathcal{M}(1) = \emptyset,\ \mathcal{M}(2) = \{p\},$
$\mathcal{M}(3) = \mathcal{M}(4) = \mathcal{M}(5) = \emptyset,\ \mathcal{M}(6) = \{p\}$ e $\mathcal{M}(k) = \emptyset$ per $k > 6$.
O anche: $\mathcal{M}(2) = \{p\}$ e $\mathcal{M}(i) = \emptyset$ se $i \neq 2$.
O anche: $\mathcal{M}(5) = \emptyset$ e $\mathcal{M}(i) = \{p\}$ se $i \neq 5$.

$\rho' = \mathbf{5, 8^{\omega}}$ non è un'esecuzione di accettazione, e non tutte le interpretazioni che rappresenta sono modelli di $\Diamond p \wedge \Diamond \neg p$: ad esempio, rappresenta anche $\mathcal{M}$ tale che $\mathcal{M}(i) = \{p\}$ per ogni i.

Esercizio: $\mathcal{A}_{p \vee \Diamond q}$



Determinare un'esecuzione d'accettazione ρ di $\mathcal{A}_{p \vee \Diamond q}$, un'interpretazione $\mathcal{M}$ rappresentata da ρ e dimostrare che $\mathcal{M} \models p \vee \Diamond q$

Determinare un'esecuzione ρ' che non sia d'accettazione e un'interpretazione $\mathcal{M}'$ rappresentata da ρ' tale che $\mathcal{M}' \not\models p \vee \Diamond q$

Il sistema SPIN

http://spinroot.com/spin/whatispin.html

Spin is a popular open-source software tool, used by thousands of people worldwide, that can be used for the formal verification of distributed software systems. The tool was developed at Bell Labs in the original Unix group of the Computing Sciences Research Center, starting in 1980. The software has been available freely since 1991, and continues to evolve to keep pace with new developments in the field. In April 2002 the tool was awarded the prestigious System Software Award for 2001 by the ACM.

Some applications

- verification of the control algorithms for the new flood control barrier built in the late nineties near Rotterdam in the Netherlands.
- Logic verification of the call processing software for a commercial data and phone switch, the PathStar switch that was designed and built at Lucent Technologies.
- Selected algorithms for a number of space missions were verified with the Spin model checker.

SPIN (= Simple Promela Interpreter)

- is a tool for analysing the logical conisistency of concurrent systems, specifically of data communication protocols.
- state-of-the-art model checker, used by $>$2000 users
- Concurrent systems are described in the modelling language called Promela.

Promela (= Protocol/Process Meta Language)

- specification language to model finite-state systems
- resembles the programming language C
- allows for the dynamic creation of concurrent processes.
- communication via message channels can be defined to be
  - synchronous (i.e. rendezvous), or
  - asynchronous (i.e. buffered).

A Promela model corresponds with a (usually very large, but) finite transition system

Properties to be verified are specified by means of LTL formulae