# Automi alternanti e model checking

Angelo Montanari (e Stefano Tognazzi)
Università di Udine

# Indice

# Automi di Buechi su parole infinite

Un *automa (non deterministico) di Buechi* su parole infinite $\mathcal{A}$ è una quintupla $(\Sigma, S, s^0, \rho, F)$, dove:

- $\Sigma$ è un alfabeto finito di simboli non vuoto;
- $S$ è un insieme finito di stati non vuoto;
- $s^0 \in S$ è lo stato iniziale;
- $\rho : S \times \Sigma \to 2^S$ è una funzione di transizione: $\rho(s, a)$ è l'insieme di stati che $\mathcal{A}$ può raggiungere a partire dallo stato $s$ in corrispondenza del carattere $a$; alternativamente, la funzione di transizione può essere vista come una relazione $\rho \subseteq S \times \Sigma \times S$.
- $F \subseteq S$ è l'insieme degli stati finali.

Il linguaggio riconosciuto da $\mathcal{A}$ viene indicato con $L(\mathcal{A})$.

# Risultati di base

## Proposizione

*Si assuma che $\mathcal{A}_1$ e $\mathcal{A}_2$ siano due automi (non deterministici) di Buechi, rispettivamente con $n_1$ e $n_2$ stati. È possibile costruire un automa (non deterministico) di Buechi $\mathcal{A}$, con $O(n_1 \cdot n_2)$ stati, tale che $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.*

## Proposizione

1. *Il problema del vuoto per gli automi (non deterministici) di Buechi è decidibile in tempo lineare (rispetto al numero di stati dell'automa).*
2. *Il problema di stabilire se esistono delle parole riconosciute da un automa (non deterministico) di Buechi con n stati è decidibile in spazio $O(\log^2 n)$.*

# Formule Booleane positive

Dato un insieme $X$ di variabili proposizionali, sia $\mathcal{B}^+(X)$ l'insieme delle *formule Booleane positive* costruite a partire da $X$, ossia costruite a partire dagli elementi di $X$ utilizzando $\wedge$ (congiunzione) e $\vee$ (disgiunzione). Per comodità, assumiamo che $\mathcal{B}^+(X)$ contenga anche le costanti logiche *true* e *false*.

Sia $Y \subseteq X$. Diremo che $Y$ soddisfa $\varphi \in \mathcal{B}^+(X)$ se l'assegnamento del valore di verità vero alle variabili proposizionali in $Y$ e falso alle variabili proposizionali in $X \setminus Y$ rende vera la formula $\varphi$.

*Esempio*. Si consideri la formula $\varphi = (s_1 \vee s_2) \wedge (s_3 \vee s_4)$. Gli insiemi $\{s_1, s_3\}$ e $\{s_1, s_4\}$ rendono entrambi vera $\varphi$.

# Formule Booleane positive e funzione di transizione

Si consideri un automa (non deterministico) di Buechi $\mathcal{A} = (\Sigma, S, s^0, \rho, F)$ e sia $\rho(s, a) = \{s_1, s_2, s_3\}$. Ogni stato $s'$ in $\rho(s, a)$ può essere visto come una delle possibili scelte non deterministiche dell'automa per il prossimo stato. Tale situazione può essere rappresentata mediante la seguente formula di $\mathcal{B}^+(S)$:

$$s_1 \vee s_2 \vee s_3$$

Negli automi di Buechi alternanti $\rho(s, a)$ può essere una qualunque formula di $\mathcal{B}^+(S)$. Ad esempio, la transizione:

$$\rho(s, a) = (s_1 \wedge s_2) \vee (s_3 \wedge s_4)$$

specifica che $\mathcal{A}$ accetta la parola $aw$ (con $a \in \Sigma$ e $w \in \Sigma^*$), quando si trova nello stato $s$, se esso accetta la parola $w$ a partire sia da $s_1$ che da $s_2$ oppure a partire sia da $s_3$ che da $s_4$.

# Automi alternanti di Buechi su parole infinite

Formalmente, un automa alternante di Buechi su parole infinite è una quintupla $\mathcal{A} = (\Sigma, S, s^0, \rho, F)$, dove tutte le componenti, ad eccezione della funzione di transizione $\rho$, sono definite come nel caso degli automi (non deterministici) di Buechi e la funzione di transizione $\rho$ è della forma: $\rho : S \times \Sigma \to \mathcal{B}^+(S)$.

Data la presenza di scelte universali ($\wedge$) accanto a scelte esistenziali ($\vee$) nella funzione di transizione degli automi alternanti, un run di un automa alternante non è una parola, ma un *albero*.

Più precisamente, un run $r$ di un automa alternante di Buechi su una parola infinita $\alpha = a_0 a_1 \ldots$ è un albero infinito etichettato con simboli dell'insieme degli stati $S$ tale che:

- $r(\epsilon) = s^0$;
- se $|x| = i$, $r(x) = s$ e $\rho(s, a_i) = \theta$, allora $x$ ha $k$ figli $x_1, \ldots, x_k$, per qualche $k \leq |S|$, e $\{r(x_1), \ldots, r(x_k)\}$ soddisfa $\theta$.

# Un esempio

Sia $\rho(s_0, a_0) = (s_1 \vee s_2) \wedge (s_3 \vee s_4)$. I nodi dell'albero di computazione a livello 1 conterranno l'etichetta $s_1$ o l'etichetta $s_2$ e anche l'etichetta $s_3$ o l'etichetta $s_4$.

Un run può contenere anche rami finiti: se $|x| = i$, $r(x) = s$ e $\rho(s, a_i) = \mathit{true}$, allora $x$ può essere privo di figli. Al contrario, non è possibile che $\rho(s, a_i) = \mathit{false}$ dal momento che *false* non è soddisfacibile. Chiediamo, inoltre, la completezza della funzione di transizione, ossia escludiamo la possibilità che $\rho(s, a_i)$ sia indefinita per qualche $s \in S$ e $a_i \in \Sigma$.

La condizione di accettazione per gli automi di Buechi alternanti è pertanto definita nel seguente modo: ogni cammino infinito in $r$ include un numero infinito di occorrenze di almeno uno stato finale.

# Problema del (non) vuoto per automi alternanti di Buechi

**Proposizione**

*Sia $\mathcal{A}$ un automa alternante di Buechi con n stati. È possibile costruire un automa (non deterministico) di Buechi $\mathcal{A}_{nd}$ con $2^{O(n)}$ stati tale che $L(\mathcal{A}_{nd}) = L(\mathcal{A})$.*

**Proposizione**

1. *Il problema del (non) vuoto per gli automi alternanti di Buechi è decidibile in tempo esponenziale.*
2. *Il problema del (non) vuoto per gli automi alternanti di Buechi è decidibile in spazio quadratico.*

# Automi di Buechi su alberi infiniti

Un automa (non deterministico) di Buechi su alberi infiniti di arità $\mathcal{D}$ è una sestupla $\mathcal{A} = (\Sigma, \mathcal{D}, S, s^0, \rho, F)$, dove:

- $\Sigma$ è un alfabeto finito di simboli;
- $\mathcal{D} \subset \mathcal{N}$ è un insieme finito di arietà;
- $S$ è un insieme finito di stati;
- $s^0 \in S$ è lo stato iniziale;
- $\rho$ è una funzione di transizione della forma $\rho : S \times \Sigma \times \mathcal{D} \to 2^{S^*}$, dove $\rho(s, a, k) \in S^k$ for each $s \in S, a \in \Sigma$, and $k \in \mathcal{D}$;
- $F \subseteq S$ è l'insieme degli stati finali.

Denotiamo con $T(\mathcal{A})$ l'insieme degli alberi riconosciuti da $\mathcal{A}$.

# Problema del (non) vuoto per automi di Buechi su alberi infiniti

**Proposizione**

*Il problema del (non) vuoto per gli automi (non deterministici) di Buechi su alberi infiniti è decidibile in tempo quadratico.*

# Automi alternanti di Buechi su alberi infiniti

- Una automa alternante di Buechi su alberi infiniti $\mathcal{A}$ è una sestupla $(\Sigma, \mathcal{D}, S, s^0, \rho, F)$, dove tutte le componenti, a parte $\rho$, sono definite come nel caso degli automi di Buechi (non deterministici) su alberi infiniti.
- $\rho$ è una funzione parziale $\rho : S \times \Sigma \times \mathcal{D} \to \mathcal{B}^+(\mathbb{N} \times S)$, dove $\rho(s, a, k) \in \mathcal{B}^+(\{1, \ldots, k\} \times S)$ per ogni $s \in S$, $a \in \Sigma$, $k \in \mathcal{D}$ per i quali $\rho(s, a, k)$ è definita.

*Esempio*. Sia $\rho(s, a, 2) = ((1, s_1) \vee (2, s_2)) \wedge ((1, s_3) \vee (2, s_1))$. $\mathcal{A}$ può scegliere fra 4 alternative: nella prima una 'copia' di $\mathcal{A}$ procede in direzione 1 nello stato $s_1$ e un'altra in direzione 1 nello stato $s_3$; nella seconda una procede in direzione 1 nello stato $s_1$ e un'altra in direzione 2 nello stato $s_1$; nella terza una procede in direzione 2 nello stato $s_2$ e un'altra in direzione 1 nello stato $s_3$; nella quarta una procede in direzione 2 nello stato $s_2$ e un'altra in direzione 2 nello stato $s_1$. Si noti come più copie possano procedere nella stessa direzione.

# Run degli automi alternanti di Buechi su alberi infiniti

Un run $r$ di un automa alternante di Buechi su alberi infiniti con etichette in $\Sigma$ e arietà in $\mathcal{D}$ è un albero infinito con etichette in $\mathcal{N}^* \times S$. Sia $t = \langle \tau, \mathcal{T} \rangle$, dove $\tau$ è il dominio e $\mathcal{T}$ è la funzione di etichettatura, l'albero di input e sia $r = \langle \tau_r, \mathcal{T}_r \rangle$. Ogni nodo di $\tau_r$ corrisponde ad un nodo di $\tau$, mentre un nodo di $\tau$ può essere associato a più nodi di $\tau_r$ (un nodo di $\tau_r$, etichettato con $(x, s)$, descrive una copia di $\mathcal{A}$ che legge il nodo $x$ di $\tau$ nello stato $s$).

Formalmente, $\langle \tau_r, \mathcal{T}_r \rangle$ deve soddisfare le seguenti condizioni:

1. $\mathcal{T}_r(\epsilon) = (\epsilon, s^0)$;
2. sia $y \in \tau_r$, $\mathcal{T}_r(y) = (x, s)$, $arity(x) = k$, $\rho(s, \mathcal{T}(x), k) = \theta$; allora, deve esistere un insieme $Q = \{(c_1, s_1), \ldots, (c_n, s_n)\} \subseteq \{1, \ldots, k\} \times S$ tale che:
   - $Q$ soddisfa $\theta$;
   - per ogni $1 \leq i \leq n$ vi è un nodo $y \cdot i \in \tau_r$ e $\mathcal{T}_r(y \cdot i) = (x \cdot c_i, s_i)$.

# Un esempio

Sia $t = \langle \tau, \mathcal{T} \rangle$ un albero con

- $arity(\epsilon) = 2$
- $\mathcal{T}(\epsilon) = a$
- $\rho(s^0, a, 2) = ((1, s_1) \vee (1, s_2)) \wedge ((1, s_3) \vee (1, s_1))$

I nodi di $r = \langle \tau_r, \mathcal{T}_r \rangle$ a livello 1 includeranno

- l'etichetta $(1, s_1)$ o l'etichetta $(1, s_2)$ e
- l'etichetta $(1, s_3)$ o l'etichetta $(1, s_1)$

# Il problema del (non) vuoto per gli automi alternanti di Buechi su alberi infiniti

**Proposizione**

*Sia $\mathcal{A}$ un automa alternante di Buechi su alberi infiniti con $n$ stati. È possibile costruire un automa (non deterministico) di Buechi su alberi infiniti $\mathcal{A}_{nd}$ con $2^{O(n \log n)}$ stati tale che $T(\mathcal{A}) = T(\mathcal{A}_{nd})$.*

**Proposizione**

*Il problema del (non) vuoto per gli automi alternanti di Buechi su alberi infiniti è decidibile in tempo esponenziale.*

# Automi di Buechi su alberi infiniti su un alfabeto con una sola lettera

Il problema del (non) vuoto per gli automi (non deterministici) di Buechi su alberi infiniti è riducibile al problema del (non) vuoto per gli automi (non deterministici) di Buechi su alberi infiniti etichettati con simboli di un alfabeto con una sola lettera.

Idea: anziché controllare che il linguaggio riconosciuto da un automa $\mathcal{A} = (\Sigma, \mathcal{D}, S, s^0, \rho, F)$ non sia vuoto, si controlla che non sia vuoto il linguaggio riconosciuto dall'automa $\mathcal{A}' = (\{a\}, \mathcal{D}, S, s^0, \rho', F)$, dove, per ogni $s \in S$ e $k \in \mathcal{D}$, $\rho'(s, a, k) = \bigcup_{b \in \Sigma} \rho(s, b, k)$.

È facile vedere che $\mathcal{A}$ accetta qualche albero se e solo se $\mathcal{A}'$ accetta qualche albero $a$-etichettato: a partire da una computazione di successo di $\mathcal{A}'$, è possibile costruire un albero di input sul quale la computazione di successo $\mathcal{A}'$ è anche una computazione di successo per $\mathcal{A}$; il viceversa segue immediatamente dalla definizione di $\rho'$.

# Tale riduzione non può essere applicata agli automi alternanti (in generale)

Si supponga di definire $\mathcal{A}'$ come nel caso degli automi di Buechi su alberi infiniti, con $\rho'(s, a, k) = \bigcup_{b \in \Sigma} \rho(s, b, k)$.

Se $\mathcal{A}'$ accetta qualche albero $a$-etichettato, non è detto che $\mathcal{A}$ accetti qualche albero.

Una condizione necessaria per la validità della riduzione è, infatti, che copie diverse di $\mathcal{A}'$ che operano sullo stesso sottoalbero di input $t$ assumano la stessa etichettatura per $t$.

Niente impedisce, però, che una copia di $\mathcal{A}'$ assuma una certa etichettatura di $t$ e un'altra copia di $\mathcal{A}'$ assuma un'etichettatura diversa di $t$.

# Automi alternanti di Buechi su alberi infiniti su un alfabeto con una sola lettera

Una possibile via d'uscita: il problema di cui al lucido precedente non si pone nel caso in cui $\mathcal{A}$ sia definito su un alfabeto con una singola lettera.

In alcuni casi concreti di interesse, gli automi alternanti di Buechi si trovano ad operare su alberi infiniti su un alfabeto con una sola lettera.

## Proposizione

*Il problema del (non) vuoto per gli automi alternanti di Buechi su alberi infiniti su un alfabeto di un solo carattere è decidibile in tempo quadratico.*

# Automi alternanti deboli

Gli automi alternanti deboli sono una sottoclasse degli automi alternanti di Buechi su alberi infiniti tale che:

- esiste una partizione dell'insieme degli stati $S$ negli insiemi $S_1, \ldots, S_n$ (il numero di insiemi della partizione è detto profondità dell'automa) tale che, per $1 \leq i \leq n$, $S_i \subseteq F$ (insieme accettante) oppure $S_i \cap F = \emptyset$ (insieme rigettante) ed
- è possibile definire un ordinamento parziale $\leq$ degli insiemi $S_1, \ldots, S_n$ tale che, per ogni $s \in S_i$ e $s' \in S_j$, se $s' \in \rho(s, a, k)$, per qualche $a \in \Sigma$ e $k \in \mathcal{D}$, allora $S_j \leq S_i$.

Ne segue che ogni cammino infinito in un run di un automa alternante debole finisce intrappolato in un qualche insieme $S_i$. Ciò consente di concludere che un cammino è di successo se e solo se l'insieme $S_i$ in cui rimane intrappolato è un insieme accettante.

# Il problema del (non) vuoto per automi alternanti deboli su un alfabeto con una sola lettera

**Proposizione**

*Il problema del (non) vuoto per automi alternanti deboli su alberi infiniti su un alfabeto con una sola lettera è decidibile in tempo lineare.*

# Automi alternanti deboli con alternanza limitata

Esiste una sottoclasse degli automi alternanti deboli di particolare interesse che contiene gli automi alternanti deboli con alternanza limitata. In tali automi, ogni insieme $S_i$ della partizione può essere classificato come *transiente*, *esistenziale* o *universale*.

Per ogni insieme $S_i$, $s \in S_i$, $a \in \Sigma$ e $k \in \mathcal{D}$, si ha che:

- se $S_i$ è *transiente*, allora $\rho(s, a, k)$ non contiene elementi di $S_i$;
- se $S_i$ è *esistenziale*, allora gli elementi di $S_i$ in $\rho(s, a, k)$ sono legati in maniera disgiuntiva (equivalentemente, se la transizione viene riscritta in forma normale disgiuntiva, in ogni disgiunto compare al più un elemento di $S_i$);
- se $S_i$ è *universale*, allora gli elementi di $S_i$ in $\rho(s, a, k)$ sono legati in maniera congiuntiva (equivalentemente, se la transizione viene riscritta in forma normale congiuntiva, in ogni congiunto compare al più un elemento di $S_i$).

# Il problema del (non) vuoto per automi alternanti deboli con alternanza limitata su un alfabeto con una sola lettera

Dalla definizione di automi alternanti deboli con alternanza limitata segue che un'alternanza può presentarsi solo nel passaggio da un insieme $S_i$ all'insieme successivo (ossia da uno stato che è legato in maniera congiuntiva agli stati dell'insieme cui appartiene ad uno stato che è legato in maniera disgiuntiva agli stati dell'insieme cui appartiene, e viceversa).

In altri termini, quando una copia di un automa visita uno stato in un insieme esistenziale (rispettivamente, universale) $S_i$, essa procede in modo esistenziale (rispettivamente, universale) fino a quando rimane in $S_i$.

## Proposizione

*Il problema del (non) vuoto per automi alternanti deboli con alternanza limitata di dimensione n e profondità m su un alfabeto con una sola lettera può essere risolto in spazio* $O(m \cdot \log^2 n)$.

# LTL e automi alternanti: traduzione - 1

I modelli di una formula della Logica Temporal Lineare (LTL) possono essere visti come computazioni (una computazione è una funzione $\pi : \omega \to 2^{\mathcal{AP}}$, dove $\mathcal{AP}$ è l'insieme delle variabili proposizionali).

È possibile dimostrare che le computazioni che soddisfano una data formula sono esattamente quelle accettate da un automa che opera su parole infinite.

## Proposizione

*Data una formula $\varphi$ di LTL, è possibile costruire un automa alternante di Buechi su parole infinite $\mathcal{A}_\phi = (\Sigma, S, s^0, \rho, F)$, con $\Sigma = 2^{\mathcal{AP}}$ e $|S| \in O(|\varphi|)$, tale che $L(\mathcal{A}_\varphi)$ è esattamente l'insieme delle computazioni che soddisfano la formula $\varphi$.*

# LTL e automi alternanti: traduzione - 2

- L'insieme degli stati $S$ consiste di tutte le sottoformule di $\varphi$ e delle loro negazioni (identifichiamo $\neg\neg\psi$ con $\psi$).
- Lo stato iniziale $s^0$ è esattamente $\varphi$.
- L'insieme degli stati finali $F$ consiste di tutti gli stati/formule di $S$ della forma $\neg(\xi U \psi)$.
- Per definire la funzione $\rho$, introduciamo il *duale* $\bar{\theta}$ di una formula $\theta$, ottenuto a partire da $\theta$ sostituendo $\vee$ con $\wedge$ (e viceversa), *true* con *false* (e viceversa) e negando le sottoformule in $S$. In modo più formale,
  - $\bar{\xi} = \neg\xi$ per $\xi \in S$
  - $\overline{true} = false$
  - $\overline{false} = true$
  - $\overline{(\alpha \wedge \beta)} = \bar{\alpha} \vee \bar{\beta}$ (se $\alpha \wedge \beta$ non appartiene a $S$)
  - $\overline{(\alpha \vee \beta)} = \bar{\alpha} \wedge \bar{\beta}$ (se $\alpha \vee \beta$ non appartiene a $S$)

# LTL e automi alternanti: traduzione - 3

La funzione di transizione $\rho$ è definita nel seguente modo:

- $\rho(p, a) = \mathit{true}$ se $p \in a$
- $\rho(p, a) = \mathit{false}$ se $p \notin a$
- $\rho(\xi \wedge \psi, a) = \rho(\xi, a) \wedge \rho(\psi, a)$
- $\rho(\neg\psi, a) = \overline{\rho(\psi, a)}$
- $\rho(X\psi, a) = \psi$
- $\rho(\xi U \psi, a) = \rho(\psi, a) \vee (\rho(\xi, a) \wedge \xi U \psi)$

Le regole per *true* e *false* sono facilmente derivabili.

Si noti che, nella definizione di $\rho(\xi U \psi, a)$, il secondo congiunto del secondo disgiunto $\xi U \psi$ è uno stato (formula atomica di $\mathcal{B}^+(S)$).

# Un esempio

Consideriamo la formula $\varphi = \neg(pUq)$, con $p, q \in \mathcal{AP}$, e la computazione $\pi = \{p\}^\omega$ in cui $p$ è sempre vero e $q$ è sempre falso.

È immediato vedere che la computazione $\pi$ soddisfa $\varphi$.

La funzione di transizione $\rho$ per $\mathcal{A}_\phi$, conterrà la seguente transizione: $\rho(\neg(pUq), \{p\}) = \overline{\rho(q, \{p\}) \vee (\rho(p, \{p\}) \wedge pUq)} = \overline{false \vee (true \wedge pUq)} = \overline{pUq} =$ ($pUq$ è (una sottoformula di) $\varphi$) $\neg(pUq)$

Esiste una computazione di $\mathcal{A}_\phi$ su $\pi$ in cui lo stato finale/ accettante $\neg(pUq)$ si ripete infinite volte (sequenza di stati $\{\neg(pUq)\}^\omega$).

# LTL e automi non deterministici di Buechi

## Proposizione

*Data una formula $\varphi$ di LTL, è possibile costruire un automa (non deterministico) di Buechi su parole infinite $\mathcal{A}_{nd\varphi} = (\Sigma, S, s^0, \rho, F)$, dove $\Sigma = 2^{\mathcal{AP}}$ e $|S|$ è in $2^{O(|\phi|)}$, tale che $L(\mathcal{A}_{nd\varphi})$ è esattamente l'insieme delle computazioni che soddisfano la formula $\varphi$.*

# Semantica di CTL

La semantica di CTL può essere definita rispetto a programmi (che operano su un insieme di variabili proposizionali $\mathcal{AP}$) della forma $P = (W, w^0, R, V)$, dove:

- $W$ è un insieme di stati (se $W$ è finito, $P$ è detto programma a stati finiti);
- $w^0 \in W$ è uno stato iniziale;
- $R \subseteq W^2$ è una relazione di accessibilità totale (assumiamo la totalità per ragioni di natura tecnica);
- $V : W \to 2^{\mathcal{AP}}$ è una funzione che, dato uno stato, specifica quali variabili proposizionali sono vere in quello stato.

Un cammino in $P$ è una sequenza di stati $u_0, u_1, \ldots$ tale che, per ogni $i \geq 0$, vale $R(u_i, u_{i+1})$. La relazione $\models$ è definita nel modo usuale.

# CTL e automi alternanti deboli con alternanza limitata: traduzione - 1

Un programma $P$ è detto programma ad albero se (l'unfolding del grafo) $(W, R)$ è un albero con radice $w^0$. Se (l'unfolding di) $(W, R)$ è un albero $\mathcal{D}$-ario, allora $P$ è un albero $\mathcal{D}$-ario infinito ($R$ è totale) etichettato con elementi in $2^{\mathcal{AP}}$.

È possibile dimostrare che i programmi ad albero che soddisfano una data formula sono esattamente quelli accettati da un automa alternante debole con alternanza limitata.

## Proposizione

*Dati una formula $\varphi$ di CTL e un insieme finito $\mathcal{D} \subset \mathbb{N}$, è possibile costruire un automa alternante debole con alternanza limitata $\mathcal{A}_\varphi = (\Sigma, \mathcal{D}, S, s^0, \rho, F)$, dove $\Sigma = 2^{\mathcal{AP}}$ e $|S|$ è in $O(|\phi|)$, tale che $T(\mathcal{A}_\phi)$ è esattamente l'insieme dei programmi ad albero $\mathcal{D}$-ari che soddisfano $\varphi$.*

# CTL e automi alternanti deboli con alternanza limitata: traduzione - 2

- L'insieme degli stati $S$ consiste di tutte le sottoformule di $\varphi$ e delle loro negazioni (identifichiamo $\neg\neg\psi$ con $\psi$).
- Lo stato iniziale $s^0$ è $\varphi$.
- L'insieme $F$ è composto da tutti gli stati/formule in $S$ della forma $\neg E(\xi U \psi)$ e $\neg A(\xi U \psi)$.
- Per definire la funzione $\rho$, introduciamo il *duale* $\bar{\theta}$ di una formula $\theta$, definito come nel caso di *LTL*:
  - $\rho(p, a, k) = true$ se $p \in a$
  - $\rho(p, a, k) = false$ se $p \notin a$
  - $\rho(\neg\psi, a, k) = \overline{\rho(\psi, a, k)}$
  - $\rho(\xi \wedge \psi, a, k) = \rho(\xi, a, k) \wedge \rho(\psi, a, k)$
  - $\rho(EX\phi, a, k) = \bigvee_{c=1}^{k}(c, \phi)$
  - $\rho(AX\phi, a, k) = \bigwedge_{c=1}^{k}(c, \phi)$
  - $\rho(E(\xi U \psi), a, k) = \rho(\psi, a, k) \vee (\rho(\xi, a, k) \wedge \bigvee_{c=1}^{k}(c, E(\xi U \psi)))$
  - $\rho(A(\xi U \psi), a, k) = \rho(\psi, a, k) \vee (\rho(\xi, a, k) \wedge \bigwedge_{c=1}^{k}(c, A(\xi U \psi)))$

# CTL e automi alternanti deboli con alternanza limitata: traduzione - 3

Infine, la partizione degli stati di $\mathcal{A}_\varphi$ e l'ordinamento (parziale) delle classi della partizione sono definiti nel seguente modo:

- lo stato/formula $\psi$ in $S$ costituisce un insieme singoletto $\{\psi\}$ della partizione;
- l'ordinamento parziale è definito nel seguente modo: $\{\xi\} \leq \{\psi\}$ se e solo se $\xi$ è una sottoformula o la negazione di una sottoformula di $\psi$;
- tutti gli insiemi della partizione sono transienti, ad eccezione degli insiemi della forma $\{E(\xi U \psi)\}$ o della forma $\{\neg A(\xi U \psi)\}$, che sono esistenziali, e degli insiemi della forma $\{A(\xi U \psi)\}$ o della forma $\{\neg E(\xi U \psi)\}$, che sono universali.

# Model checking per LTL

Dati un programma a stati finiti e una formula $\varphi$ di *LTL* che specifica le computazioni ammissibili (del programma), il problema del *model checking* consiste nello stabilire se tutte le computazioni del programma sono ammissibili.

- Sia $\mathbf{u} = u_0, u_1, \ldots$ un cammino di un programma a stati finiti $P = (W, w^0, R, V)$ tale che $u_0 = w^0$.
- Definiamo *computazione* di $P$ la sequenza $V(u_0), V(u_1), \ldots$.
- Diremo che $P$ soddisfa una formula $\varphi$ di *LTL* se *tutte* le computazioni di $P$ soddisfano $\varphi$.

Il problema del model checking per LTL è il problema di stabilire se $P$ soddisfa o meno $\varphi$.

# Model checking per LTL: passi intermedi - 1

L'automa $\mathcal{A}_P$ per il programma $P$.

- Un programma $P = (W, w^0, R, V)$ può essere visto come un automa (non deterministico) di Buechi $\mathcal{A}_P = (\Sigma, W, w^0, \rho, W)$, dove $\Sigma = 2^{\mathcal{AP}}$ e $v \in \rho(u, a)$ se e solo se $uRv$ e $a = V(u)$.
- Dato che l'insieme degli stati finali coincide con l'insieme di tutti gli stati, ogni run infinito di $\mathcal{A}_P$ è di successo (accettante) e, quindi, $L(\mathcal{A}_P)$ è l'insieme di tutte le computazioni di $P$.

L'automa $\mathcal{A}_\varphi$ per la formula $\varphi$ di LTL.

- Per ogni formula $\varphi$ di LTL, è possibile costruire un automa $\mathcal{A}_\varphi$ tale che $L(\mathcal{A}_\varphi)$ è esattamente l'insieme delle computazioni che soddisfano la formula $\phi$.

# Model checking per LTL: passi intermedi - 2

- Date le proprietà di chiusura rispetto all'intersezione e alla complementazione degli automi (non deterministici) di Buechi, il problema di verificare se $L(\mathcal{A}_P) \subseteq L(\mathcal{A}_\varphi)$ (problema del model checking) è equivalente al problema di verificare se l'intersezione delle computazioni accettate da $\mathcal{A}_P$ e di quelle che rendono falsa $\varphi$ è vuota, ossia al problema del vuoto per $L(\mathcal{A}_P) \cap L(\overline{\mathcal{A}_\varphi})$.
- Per costruire $L(\overline{\mathcal{A}_\phi})$ si considerino le seguenti uguaglianze: $L(\overline{\mathcal{A}_\phi}) = \overline{L(\mathcal{A}_\phi)} = \Sigma^\omega - L(\mathcal{A}_\phi)$.
- È possibile costruire un automa $\mathcal{A}_{\neg\phi}$ che accetta il linguaggio $L(\overline{\mathcal{A}_\phi})$ con $2^{O(|\phi|)}$ stati.
- Infine, è possibile costruire un automa che riconosce il linguaggio intersezione $L(\mathcal{A}_P) \cap L(\mathcal{A}_{\neg\phi})$ con $O(|W| \cdot 2^{O(|\phi|)})$ stati.

# Un algoritmo di model checking basato su automi per LTL

## Proposizione

*Stabilire se un programma a stati finiti $P$ soddisfa una formula $\varphi$ di LTL può essere fatto in tempo $O(|P| \cdot 2^{O(|\varphi|)})$ e in spazio $O((|\varphi| + \log |P|)^2)$.*

Si noti che la complessità temporale è polinomiale nella dimensione del programma ed esponenziale nella dimensione della formula (specifica).

Dato che la dimensione della formula è di norma sufficientemente ridotta, tale soluzione si dimostra effettivamente praticabile – si veda il sistema Aalta (Another Algorithm for LTL To Buechi Automata).

# Model checking per CTL

Nel caso delle logiche temporali ramificate, come CTL, ogni programma corrisponde ad un singolo albero di computazione.

Il model checking si riduce alla verifica dell'accettazione di tale albero di computazione da parte dell'automa che descrive la formula (specifica).

# Model checking per CTL: passi intermedi - 1

- Un programma $P = (W, w^0, R, V)$ può essere visto come un albero $\langle \tau_P, \mathcal{T}_\mathcal{P} \rangle$, con etichette in $W$, ottenuto attraverso l'unfolding di $P$ a partire dallo stato iniziale $w^0$.
- Per ogni $w$ in $W$, definiamo $arity(w)$ come il numero dei suoi successori rispetto alla relazione $R$.
- Sia $succ_R(w) = (w_1, \ldots, w_{arity(w)})$ la lista ordinata dei successori di $w$ (assumiamo che i nodi di $W$ siano ordinati).
- $\tau_P$ e $\mathcal{T}_\mathcal{P}$ possono essere definiti induttivamente nel seguente modo:
  - $\epsilon \in \tau_P$ e $\mathcal{T}_\mathcal{P}(\epsilon) = w^0$;
  - per $y \in \tau_P$, con $succ_R(\mathcal{T}_P(y)) = (w_1, \ldots, w_k)$, e per ogni $1 \leq i \leq k$, $y \cdot i \in \tau_P$ e $\mathcal{T}_P(y \cdot i) = w_i$.

# Model checking per CTL: passi intermedi - 2

- L'albero del programma può essere successivamente trasformato in un albero $\mathcal{D}$-ario etichettato con elementi in $2^{\mathcal{AP}}$
- Tale albero è formalmente definito dalla coppia $(\tau_P, V \cdot \mathcal{T}_P)$, dove $V \cdot \mathcal{T}_P = V(\mathcal{T}_P(y))$, con $y \in \tau_P$.

Consideriamo ora una formula $\varphi$ di *CTL*. Sia $\mathcal{A}_{\mathcal{D},\varphi}$ l'automa alternante debole con alternanza limitata che accetta esattamente gli alberi $\mathcal{D}$-ari che soddisfano la formula $\varphi$.

È facile vedere che $(\tau_P, V \cdot \mathcal{T}_P)$ è accettato da $\mathcal{A}_{\mathcal{D},\varphi}$ se e solo se $P \models \varphi$.

Mostriamo, ora, che l'automa prodotto di $P$ e $\mathcal{A}_{\mathcal{D},\varphi}$ è un automa alternante debole con alternanza limitata su un alfabeto di una lettera che è non vuoto se e solo se $(\tau_P, V \cdot \mathcal{T}_P)$ è accettato da $\mathcal{A}_{\mathcal{D},\varphi}$.

# Model checking per CTL: passi intermedi - 3

- Sia $\mathcal{A}_{\mathcal{D},\varphi} = (2^{\mathcal{AP}}, \mathcal{D}, S, \varphi, \rho, F)$ e sia $S_1, \ldots, S_n$ la partizione di $S$.
- L'automa prodotto di $P$ e $\mathcal{A}_{\mathcal{D},\varphi}$ è l'automa $\mathcal{A}_{P,\varphi} = (\{a\}, \mathcal{D}, W \times S, (w^0, \varphi), \delta, G)$, dove:
  - dati $s \in S$, $w \in W$, $succ_R(w) = (w_1, \ldots, w_k)$ e $\rho(s, V(w), k) = \theta$, definiamo $\delta((w, s), a, k) = \theta'$, dove $\theta'$ è ottenuta da $\theta$ sostituendo ogni atomo $(c, s')$ presente in $\theta$ con l'atomo $(c, (w_c, s'))$;
  - $G = W \times F$;
  - $W \times S$ è partizionato in $W \times S_1, W \times S_2, \ldots, W \times S_n$;
  - per $1 \leq i \leq n$, $W \times S_i$ è transiente (rispettivamente, esistenziale, universale) se $S_i$ è transiente (rispettivamente, esistenziale, universale).

  Si noti che se $P$ ha $m_1$ stati e $\mathcal{A}_{\mathcal{D},\varphi}$ ha $m_2$ stati, allora $\mathcal{A}_{P,\phi}$ ha $O(m_1 \cdot m_2)$ stati.

# Un algoritmo di model checking per CTL basato su automi

**Proposizione**

*$T(\mathcal{A}_{P,\phi})$ non è vuoto se e solo se $P \models \phi$*

**Proposizione**

*Stabilire se un programma a stati finiti $P$ soddisfa una formula $\varphi$ di CTL può essere fatto in tempo $O(|P| \cdot |\phi|)$ e spazio $O(|\phi| \cdot \log^2 |P|)$.*