

Università degli Studi di Napoli Federico II

**Facoltà di Scienze MM.FF.NN.
Corso di Laurea in Informatica**

Anno Accademico 2009/2010



Appunti di Calcolabilità e Complessità
Lezione 9: “Introduzione alle logiche temporali lineari”

Acanfora Carmine N97/18
Camuso Dario N97/21

INDICE

1.1 MODELLAZIONE DEI SISTEMI E STRUTTURE DI KRIKKE	3
1.2 TRASFORMAZIONE DALLE MACCHINE DI MOORE IN STRUTTURE DI KRIKKE.....	5
1.3 TRASFORMAZIONE DELLE STRUTTURE DI KRIKKE IN AUTOMI DI BÜCHI	7
2.1 LOGICA TEMPORALE LINEARE (LTL)	8
2.2 ESEMPI	10

1.1 MODELLAZIONE DEI SISTEMI E STRUTTURE DI KRIPKE

Quando abbiamo un sistema, dal punto di vista formale, lo vediamo come una macchina che legge un input, lo elabora modificando il proprio stato interno e restituisce un output. Questa è l'idea più astratta che si può avere di una macchina. Le macchine più semplici sono quelle in cui abbiamo un insieme di stati finito, ossia le diverse possibili configurazioni della macchina sono finite.

Il primo concetto su cui ragionare è quello di avere una macchina a stati finiti (un automa) che dia la possibilità di avere un output. Esistono due possibili modelli per queste macchine (la differenza sta nel come calcolare l'output):

- **Macchine di Moore:** l'output è funzione esclusiva dello stato, a seconda dello stato in cui la macchina si trova l'ambiente riceverà un determinato output. Possono essere viste come un grafo in cui i nodi sono labellati con un segnale di output.
- **Macchine di Mealy:** l'output dipende sia dallo stato che dalla lettera correntemente letta. Possono essere viste come un grafo in cui gli archi sono labellati con un segnale di output.

Si dimostra che le due tipologie di macchine sono equivalenti, ossia che per ogni macchina di Moore esiste una macchina di Mealy che riconosce lo stesso identico linguaggio. La differenza sta nel fatto che le macchine di Mealy sono esponenzialmente più succinte. Possiamo tradurre una macchina di Moore in una di Mealy senza problemi, mentre il viceversa è possibile ma ci costa.

Da questo punto in poi verranno trattate solo macchine di Moore.

$M = \langle \Sigma_I, \Sigma_O, Q, \delta, \lambda, q_0 \rangle$

Σ_I : alfabeto di input

Σ_O : alfabeto di output

$\delta: Q \times \Sigma_I \rightarrow Q$ funzione di transizione: dagli stati ed i segnali di input restituisce un nuovo stato

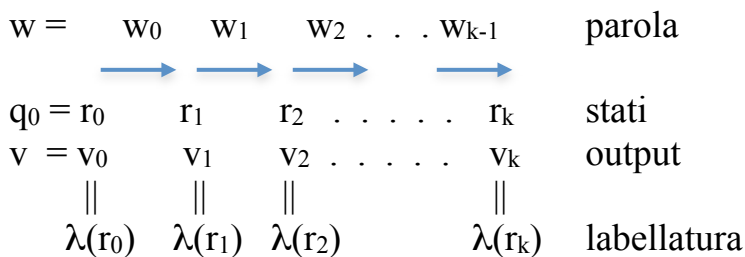
$\lambda: Q \rightarrow \Sigma_O$ funzione di labellatura: dato uno stato restituisce un segnale di output

Lo stato iniziale q_0 è unico perché la macchina è deterministica.

Non ci sono stati d'accettazione perché questa macchina non è un riconoscitore di linguaggi, ma un trasduttore che trasforma un insieme d'informazioni (input) in un altro (output). Un esempio sono gli algoritmi di ordinamento che non devono accettare ma trasformare l'informazione trattata. Considereremo trasduttori deterministici in cui la δ dato un input e lo stato restituirà uno ed un solo stato.

Se abbiamo una parola $w \in \Sigma_I^*$ la sua esecuzione sulla macchina M sarà un *run* classico degli automi deterministici, a cui sarà associato anche l'output. Dunque:

$$r \in \text{run}(w) \quad \text{e} \quad v \in \text{output}(w) \subseteq \Sigma_O^*$$



Data una macchina e la sua specifica, ossia cosa la macchina dovrebbe fare per essere considerata corretta, vogliamo studiare la correttezza della macchina stessa. Serve un linguaggio per descrivere la specifica della macchina che sia altrettanto formale quanto questi modelli.

La prima cosa da fare è definire dei modelli per la descrizione in generale delle specifiche e questi modelli derivano dallo studio delle logiche modali e prendono il nome di Kripke Structure.

La logica modale estende la logica proposizionale con degli operatori di “necessità” in modo da poter esprimere condizioni del tipo “questa proprietà è necessario che sia vera” oppure “questa proprietà può essere vera”. Questi concetti non possono essere espressi attraverso la logica proposizionale. Per aumentare l’espressività non basta più avere una funzione di valutazione che assegna il valore di verità alle varie proposizioni, ma occorre un modello (in generale sarà un grafo labellato) che permetterà di esprimere anche la verità o meno degli operatori modali. La logica modale è anche conosciuta come la logica in cui la semantica è a più mondi in quanto le varie proposizioni hanno un significato diverso tra mondo e mondo, come se si avessero degli universi paralleli. Ogni universo ha i suoi concetti di verità ed è possibile passare da un universo all’altro. Kripke Structure:

$$K = \langle AP, W, R, L \rangle$$

AP: Atomic Proposition, proposizioni atomiche prive di variabili

W: World, insieme di stati

R: relazione di transizione tra mondi e mondi

L: funzione di labellatura che associa ad ogni mondo un insieme di AP che considereremo vere in quel mondo

$$R \subseteq W \times W$$

$$L: W \rightarrow 2^{AP}$$

Data una Kripke Structure K possiamo ottenere un insieme di percorsi:

$$\text{Path}(K) \subseteq W^\omega$$

$$\text{Path}(K) = \{ \pi \in W^\omega \mid \forall i \in \mathbb{N}, (\pi_i, \pi_{i+1}) \in R \}$$

Dunque i *paths* sono una sequenza di mondi coerenti con la R.

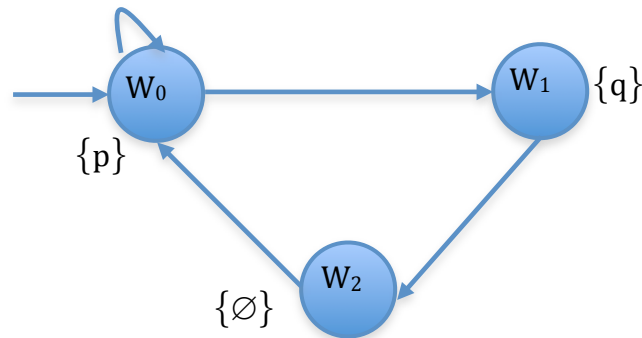
Il linguaggio della Kripke Structure è il seguente:

$$L(K) = \{ L(\pi) \mid \pi \in \text{Path}(K) \} \subseteq (2^{AP})^\omega$$

$$\text{dove } L(\pi) = L(\pi_0) \cdot L(\pi_1) \cdot L(\pi_2) \cdot \dots = \prod_{i=0}^{\infty} L(\pi_i)$$

$L(K)$ è l'insieme di tutte le parole formate su proposizioni vere formate al passare del tempo.

Consideriamo la seguente struttura composta da tre mondi (W_0, W_1, W_2):



Le AP sono p e q ed il linguaggio rappresentato dal modello (ovvero quello accettato dalla struttura) è il seguente:

$$(\{p\}^* \{q\} \emptyset)^\omega + \{p\}^\omega$$

Le Kripke Structure di default sono definite su parole infinite, poiché per ottenere il caso finito è sufficiente inserire un loop su se stesso per lo stato che si vuole rendere finale.

1.2 TRASFORMAZIONE DALLE MACCHINE DI MOORE IN STRUTTURE DI KRIPKE

Vogliamo trasformare una macchina di Moore in una Kripke Structure in modo da poter dire che una macchina di Moore è un modello molto più pratico, ma dal punto di vista dell'astrazione è preferibile un modello basato su Kripke Structure. Consideriamo una macchina di Moore:

$$M = \langle \Sigma_I, \Sigma_O, Q, \delta, \lambda, q_0 \rangle$$

La vogliamo trasformare in una Kripke Structure. La trasformazione language-equivalent relativamente all'alfabeto di output, però nella labellatura della Kripke Structure includeremo anche alfabeto di input.

Sia la macchina di Moore che la Kripke Structure sono un grafo quindi intuitivamente sono abbastanza simili.

Ad ogni segnale viene associata un'Atomic Proposition, dunque l'insieme delle AP è dato dall'unione dei segnali di input e di output.

$$AP = \Sigma_I \cup \Sigma_O$$

Il mondo iniziale W_0 di questa struttura sarà q_0 poiché come mondi considereremo lo stato iniziale unito gli stati per l'alfabeto di input.

$$W_0 = q_0$$

$$W = \{q_0\} \cup Q \times \Sigma_I$$

Questo perché la struttura di Kripke non ha il concetto di input in se (c'è solo un alfabeto che descrive quali sono le proprietà vere o false a seconda dello stato), dunque lo stato oltre a rappresentare lo stato della macchina rappresenta anche quale sarebbe l'input in quel momento e dato lo stato si avranno anche tutti i possibili input. Il mondo iniziale serve perché all'inizio non si conosce l'input (si possono avere tutti i possibili input), poi una volta che si ha l'input si passa allo stato successivo con associato quell'input in particolare. All'inizio non avendo l'input si ha soltanto lo stato che leggerà un qualsiasi input ed andrà nello stato successivo con tutti i possibili input, poi, poiché la struttura è non deterministica, si può andare in uno di questi stati e scegliendolo vorrà dire è stato letto quel particolare carattere di input. Per quanto riguarda la funzione di transizione abbiamo una copia per ogni segnale di input diverso:

$$(q_0, (\delta(q_0, \sigma), \sigma)) \in R \quad \forall \sigma \in \Sigma_I$$

Dallo stato iniziale si passa allo stato seguente rispetto a σ quando ho letto σ . Dopo aver effettuato questo passaggio si avranno sempre delle coppie stato/lettera dell'input. Nel primo passo non si ha conoscenza dell'input associato, mentre nel secondo si conosce l'input.

$$((q, \sigma), (\delta(q, \sigma'), \sigma')) \in R \quad \forall (q, \sigma) \in W \quad \forall \sigma' \in \Sigma_I$$

Se ci si trova in uno stato q leggendo σ si andrà nel successore di q rispetto a σ' ricordando il segnale di input σ' .

La funzione di labellatura contiene sia l'output (come la macchina di Moore) che l'input, perché l'input non è qualcosa di intrinseco nella Kripke Structure ma deve essere inserito dall'esterno. La labellatura del mondo iniziale coinciderà con la labellatura dello stato iniziale di output della macchina di Moore.

$$\begin{aligned} L(q_0) &= \{\lambda(q_0)\} \\ L((q, \sigma)) &= \{\lambda(q), \sigma\} \end{aligned}$$

Per qualsiasi altra coppia stato/segnale di input la labellatura contiene sia l'output dello stato che il segnale di input ed in questo modo la Kripke Structure fornisce contemporaneamente informazioni sia sull'input che sull'output. Servono due alfabeti (uno per l'input Σ_I ed uno per l'output Σ_O) per fare distinzione fra le due informazioni visto che con un unico alfabeto sarebbe impossibile.

Il processo contrario, da Kripke Structure a macchina di Moore, non è possibile poiché la R è una relazione per cui ogni stato può avere più successori. La trasformazione inversa non è possibile perché M è una macchina di Moore deterministica. Questo tipo di trasformazione viene detta "trasformazione con ritardo" perché c'è un ritardo di uno step fra il segnale di input e quello di output. La macchina all'inizio non ha input, il primo segnale di input sarà dato dai successori dello stato q_0 . In alternativa si può considerare σ' non come il carattere letto in precedenza ma come quello che verrà letto successivamente, ossia come una previsione futura. Questo nuovo modello è detto "senza ritardo" poiché tutti gli stati hanno sia l'input che l'output (previsione).

1.3 TRASFORMAZIONE DELLE STRUTTURE DI KRIPKE IN AUTOMI DI BÜCHI

Gli automi servono a descrivere formalmente tutte le tecniche di verifica e la prima cosa da fare è trasformare un Kripke Structure (astrazione del modello del sistema) in un automa che riconosce le sue computazioni.

$$K = \langle AP, W, R, L \rangle \rightarrow N = \langle 2^{AP}, W, W_0, \delta, W \rangle$$

N è un automa di Büchi che accetta lo stesso linguaggio di K. L'insieme di possibili input è dato dalle possibili AP che possono essere vere (2^{AP}), gli stati sono i mondi della Kripke Structure, gli stati iniziali sono i mondi iniziali di K e come accettazione si mette tutto. La funzione di transizione è riportata di seguito:

$$\delta(q, \sigma) = \begin{cases} \{q' \in W \mid (q, q') \in R\} & \text{se } \sigma = L(q) \\ \emptyset & \text{altrimenti} \end{cases}$$

Dato uno stato q la funzione di transizione deve restituire tutti i mondi q' successivi di q secondo la R quando quello che sto leggendo (σ) è effettivamente la labellatura che nella Kripke Structure era stata assegnata allo stato q. In caso contrario è restituito l'insieme vuoto. In tutte le computazioni le parole sono accettate se sono delle sequenze di informazioni che sotto hanno degli stati che formano un *run* coerente con l'automa e con la relazione di transizione.

Possiamo costruire l'automa relativo al complemento della specifica di una logica. Se l'intersezione fra questo automa e quello descritto nella costruzione precedente (N) è vuota allora le esecuzioni del modello della macchina non sono sbagliate perché non soddisfano ciò che non vogliamo, mentre se l'intersezione è non vuota allora esiste un'esecuzione della macchina che soddisfa la specifica sbagliata. Quindi basandoci sul vuoto capiremo se il modello è corretto o meno.

2.1 LOGICA TEMPORALE LINEARE (LTL)

La logica che verrà descritta è detta lineare perché vede il flusso temporale come un'unica linea, non sarà trattato il concetto di non determinismo della logica. La computazione verrà vista come un unico flusso temporale lineare.

SINTASSI:

Una formula LTL Ψ è espressa come di seguito (P sta per proposizioni):

$$\Psi := P \mid \neg\Psi \mid \Psi \wedge \Psi \mid \Psi \vee \Psi \mid X\Psi \mid \Psi U \Psi \mid \Psi R \Psi$$

Qui stiamo descrivendo la sintassi quindi Ψ e Ψ sono classi di formule che rispettano la stessa grammatica.

X: *next*, nel momento successivo a quello in cui sto Ψ sarà vera

U: *until*, la parte destra dovrà essere vera prima o poi nel tempo e fintantoché ciò non si verifica la parte sinistra deve essere vera

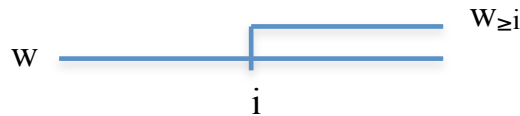
R: *release*, duale dell'*until*

Per avere una definizione formale della semantica è necessario definire il concetto di classe di possibili strutture o modelli per la logica LTL.

I modelli di questa logica saranno parole infinite sull'alfabeto 2^{AP} .

$$w \in \text{LTL-STRUCTURE}(2^{AP})^\omega$$

Indicheremo con $w_{\geq i}$ il suffisso dalla posizione i in poi per la parola w :



SEMANTICA:

La semantica si definisce ricorsivamente su tutti gli elementi. Il simbolo di modello sarà \models e w una parola infinita su $(2^{AP})^\omega$ e $p \in AP$. Inoltre poiché la semantica è ricorsiva la semantica di qualsiasi formula può essere espressa in base ai componenti della formula stessa.

$$w \models p \quad \Leftrightarrow \quad p \in L(W_0)$$

$$w \models \neg\Psi \quad \Leftrightarrow \quad w \not\models \Psi$$

$$w \models \Psi_1 \text{ op } \Psi_2 \quad \Leftrightarrow \quad w \models \Psi_1 \text{ op } w \models \Psi_2 \quad \text{op} \in \{\wedge, \vee\}$$

$$w \models X\Psi \quad \Leftrightarrow \quad w_{\geq 1} \models \Psi$$

$$w \models \Psi_1 U \Psi_2 \quad \Leftrightarrow \quad \exists i \in \mathbb{N}: w_{\geq i} \models \Psi_2 \wedge \forall j \in \mathbb{N} \ 0 \leq j < i \ w_{\geq j} \models \Psi_1$$

$$w \models \Psi_1 R \Psi_2 \quad \Leftrightarrow \quad \forall i \in \mathbb{N}: w_{\geq i} \models \Psi_2 \vee \exists j \in \mathbb{N} \ 0 \leq j < i \ w_{\geq j} \models \Psi_1$$

$$K \models \Psi \quad \Leftrightarrow \quad \forall w \in L(K): w \models \Psi$$

Per le Kripke Structure $L(K)$ è l'insieme delle parole infinite su 2^{AP} che erano le labellature dei *paths* nella struttura.

Si possono utilizzare due ulteriori operatori (uno duale dell'altro):

$F\Psi = \text{true} \cup \Psi$ *future*: nel futuro dovrà essere vero Ψ

$G\Psi = \text{false} \cap \Psi$ *globally*: d'ora in poi Ψ sarà sempre vero

Questi operatori non sono definiti nella sintassi e nella semantica perché sono due casi particolari dell'*until* e del *release*.

Per una formula LTL Ψ valgono le seguenti definizioni:

Ψ è *soddisfacibile* se: $\exists w \in (2^{AP})^\omega: w \models \Psi \Leftrightarrow \exists \text{Kripke Structure}: K \models \Psi$

Se esiste $w \models \Psi$ allora la stessa parola può essere considerata come una successione di stati, una Kripke Structure infinita in cui c'è un unico percorso ed è quello che soddisfa. D'altra parte se $K \models \Psi$ significa che tutte le parole della Kripke Structure soddisfano, quindi ne deve esistere almeno una ed è soddisfatta la prima parte.

Ψ è *valida* se: $\forall w \in (2^{AP})^\omega: w \models \Psi \Leftrightarrow \neg\Psi$ non è soddisfacibile

Supponiamo che la prima parte sia vera e la seconda falsa. In questo caso se esiste un modello per $\neg\Psi$ vuol dire che esiste una parola che soddisfa $\neg\Psi$. Per definizione quella parola non può soddisfare Ψ , ma la prima parte impone che Ψ soddisfi tutte le possibili parole dunque si cade in contraddizione.

Per quanto riguarda il model checking, a seconda di quello che ci serve, potremo dire che i modelli di Ψ sono o le parole che soddisfano Ψ o i modelli che hanno come parole tutte quelle che soddisfano Ψ .

$$\text{model}(\Psi) = \{K \mid K \models \Psi\} \mid \{w \in (2^{AP})^\omega: w \models \Psi\}$$

Vogliamo definire l'equivalenza fra formule LTL. Diremo che due formule LTL sono equivalenti se la struttura è contemporaneamente modello di entrambe. Tutto ciò che rende vera una deve rendere vera l'altra e viceversa.

$$\Psi_1 \equiv \Psi_2 \Leftrightarrow (\forall K: K \models \Psi_1 \Leftrightarrow K \models \Psi_2)$$

Utilizzando le leggi di De Morgan, espanse sulla logica LTL, si ottengono le seguenti formule duali:

$$\neg(\Psi_1 \wedge \Psi_2) \equiv (\neg\Psi_1) \vee (\neg\Psi_2)$$

$$\neg X\Psi \equiv X\neg\Psi$$

$$\neg(\Psi_1 \cup \Psi_2) \equiv ((\neg\Psi_1) \cap (\neg\Psi_2))$$

La dualità permette di trasformare qualsiasi formula LTL in una equivalente in *Positive Normal Form* (utilizzando le formule di De Morgan viene portata la negazione davanti alle proposizioni atomiche).

TEOREMA: ogni formula LTL Ψ ha un'equivalente formula Ψ' in PNF

$$\forall \Psi \in \text{LTL}, \quad \exists \Psi' \in \text{LTL(PNF)} \quad \Psi \equiv \Psi'$$

La dimostrazione è per induzione e si basa sulle proprietà di dualità.

Le seguenti proprietà vengono dette “one step unfolding” o “fix point properties” e sono basate sull’*until* ed il *release* perché il *next* è un qualcosa di fisso mentre gli altri due sono caratterizzati dall’arbitrarietà.

$$\begin{aligned}\Psi_1 U \Psi_2 &\equiv \Psi_2 \vee \Psi_1 \wedge X\Psi_1 U \Psi_2 \\ \Psi_1 R \Psi_2 &\equiv \Psi_2 \wedge (\Psi_1 \vee X\Psi_1 R \Psi_2)\end{aligned}$$

Queste proprietà riducono il controllo dell’*until* al controllo ora di un qualcosa e poi al passo successivo dell’*until* di nuovo, stessa cosa per il *release*. Per una generica formula Ψ possiamo generalizzare in:

$$\Psi \equiv \Psi_2 \vee \Psi_1 \wedge X\Psi$$

2.2 ESEMPI

1. Ogni richiesta di risorsa effettuata sempre deve essere prima o poi garantita, ossia soddisfatta.

$$G[G \text{ req} \rightarrow F \text{ grant}]$$

2. Ogni richiesta di risorsa effettuata infinitamente spesso deve essere prima o poi garantita, ossia soddisfatta.

$$G[GF \text{ req} \rightarrow F \text{ grant}]$$

Lo si esprime dicendo che d’ora in poi troverò sempre un punto in cui ci sarà req, non è detto che sia vero in ogni istante ma che per ore e fino all’infinito esisterà sempre un punto dopo quello attuale in cui troverò req.

3. Le proprietà di safety sono utilizzate per garantire qualcosa che deve andare sempre a buon fine, solitamente sono espresse mediante l’utilizzo dell’operatore *globally*.

$$G[\neg \text{error}]$$

$$G[\neg \text{proc}_1 \vee \neg \text{proc}_2]$$

L’ultima è una proprietà di safety per lo scheduling dei processi che accedono e sezioni critiche comuni.