UNIVERSITÀ DEGLI STUDI DI UDINE

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea in Informatica

Tesi di Laurea

GESTIONE DI INSIEMI DI GRANULARITÀ TEMPORALI MEDIANTE AUTOMI IN SISTEMI DI BASI DI DATI

Relatore:
Prof. ANGELO MONTANARI

Laureando:
DAVIDE BRESOLIN

Correlatore:
Dott. GABRIELE PUPPIS

ANNO ACCADEMICO 2002-2003

Indice

Introduzione

Le attività umane sono da sempre strettamente correlate con il tempo e con le unità temporali utilizzate per dividerlo (per esempio, anni, mesi, giorni, minuti, etc...). Il problema di rappresentare e gestire le relazioni temporali ha quindi una notevole rilevanza anche in molti campi dell'informatica, a partire dai fondamenti della logica (logiche temporali, problemi di soddisfacibilità di vincoli temporali, etc...), fino alle applicazioni nell'ambito delle basi di dati e dell'intelligenza artificiale (sistemi esperti, basi di dati temporali, informatica medica, etc...). Per una rassegna generale degli studi nel campo della rappresentazione e della gestione delle informazioni temporali, non limitata a specifiche applicazioni, suggeriamo la lettura di [CM00].

Nel campo delle applicazioni delle strutture temporali ai sistemi informatici si ha spesso a che fare con il concetto di *granularità*, ossia di unità temporale che suddivide la linea del tempo in intervalli, e di *calendario*, che qui assume il significato di sistema che raggruppa diverse granularità in modo ordinato, mantenendo traccia delle relazioni che intercorrono tra di esse. Il modo in cui le informazioni temporali vengono espresse nel linguaggio naturale, utilizzando, secondo le necessità, diverse unità temporali (per esempio, "l'appuntamento è fissato per *domani*", "la *scorsa settimana* Paolo è venuto a trovarmi", etc...), si riflette spesso nei requisiti che i sistemi informatici per la manipolazione di tali informazioni devono rispettare. La richiesta che un formalismo per la rappresentazione e la manipolazione del tempo consenta di operare con informazioni espresse mediante granularità diverse è dunque assai frequente.

Le applicazioni tipiche in cui questi concetti vengono utilizzati sono:

- le **basi di dati temporali**, per le quali è utile disporre di un linguaggio di interrogazione che consenta di definire le granularità mediante le quali le informazioni sono espresse e di operare conversioni tra granularità diverse;

- i **sistemi di basi di dati distribuiti**, che, raccogliendo informazioni provenienti da sorgenti diverse, spesso espresse mediante granularità diverse, richiedono la presenza di meccanismi automatici di conversione per poter rappresentare i dati in maniera uniforme;

- la **specifica e validazione dei sistemi reattivi**, per le quali sorge spesso la necessità di verificare la consistenza dei vincoli temporali imposti al sistema (per esempio, un sistema real-time), mediante la formulazione di espressioni formali (ad esempio, formule di una logica temporale) associate a granularità diverse.

Secondo un approccio comunemente accettato (cfr. [BDE+98]), ogni *granularità temporale* può essere vista come una partizione di un opportuno *dominio temporale* in gruppi di elementi, considerati come unità indivisibili (*granuli*). Molte delle granularità di interesse pratico possono essere modellate come sequenze infinite di granuli che presentano pattern ripetuti ed, eventualmente, intervalli vuoti tra i granuli oppure all'interno dei granuli. In alcuni casi la nozione di granularità viene limitata alle *suddivisioni regolari* del dominio temporale (ad esempio, secondi, minuti, ore, giorni, etc...), in cui i granuli hanno dimensione uniforme, oppure alle *suddivisioni contigue* del dominio temporale (ad esempio, mesi, anni, etc...), in cui non esistono intervalli vuoti tra i granuli. In [BJW00] viene proposta una nozione generale di granularità temporale in grado di catturare tutte queste possibilità. Tale definizione viene inoltre estesa consentendo di associare un'etichetta a ciascun granulo.

In letteratura sono stati proposti molti formalismi, di diversa natura, per rappresentare la nozione di granularità temporale. Ognuno di essi deve (dovrebbe) tuttavia soddisfare i seguenti requisiti:

- **Espressività.** L'insieme di tutte le possibili granularità non è numerabile; di conseguenza, ogni formalismo rappresenta solamente un sottoinsieme proprio di tutte le granularità. Tale sottoinsieme deve essere sufficientemente esteso per risultare di interesse pratico.

- **Manipolazione algoritmica.** Il formalismo deve consentire di rappresentare in modo finito granularità infinite e deve permettere la manipolazione delle stesse mediante opportuni algoritmi.

- **Compattezza.** Il formalismo deve consentire di ottenere rappresentazioni delle granularità che siano il più possibile compatte ed efficienti.

In particolare, è possibile identificare tre diversi approcci alla rappresentazione delle granularità: l'approccio insiemistico-algebrico (ad esempio, Collection Expression [LMF86, BDS99], Slice Expression [NS92], Algebra degli Intervalli [ZU96], Calendar Algebra [BJW00, NJW02], etc...), l'approccio logico-dichiarativo (si vedano [MPP99, FM02, Fra01, CFP03] per alcuni risultati riguardanti l'espressività delle logiche temporali interpretate sulle granularità e [BN96] per un approccio basato sulla programmazione logica) e l'approccio basato su stringhe o automi (si veda [Wij00] per un approccio basato su stringhe e [DM01, DMP03a, DMP03b] per alcuni approcci basati su automi).

Il formalismo più significativo basato sull'approccio insiemistico-algebrico è quello della Calendar Algebra (descritto in [BJW00, NJW02]). Esso, analogamente ai formalismi descritti in [LMF86, BDS99] (Collection Expression) e in [NS92] (Slice Expression), è basato sulla costruzione di termini a partire da un insieme finito di granularità e di operatori, e consente di rappresentare molte granularità di interesse pratico, incluse le granularità infinite e periodiche. Il suo limite principale è quello di non essere particolarmente adatto per risolvere i problemi di equivalenza e manipolazione delle strutture temporali, come le operazioni di *conversione tra granularità*, che consentono di determinare quali granuli di una granularità sono in relazione con un opportuno insieme di granuli di una seconda granularità. Nonostante ciò, in [NJW02] vengono approfonditi alcuni aspetti relativi alla manipolazione delle strutture temporali e vengono forniti alcuni algoritmi per la conversione tra granularità.

Un nuovo promettente approccio alla rappresentazione delle granularità è stato proposto da Wijsen in [Wij00] mediante la definizione di un sistema basato su codifiche di parole fondamentalmente periodiche, dette *Granspec*. Questo formalismo è espressivo almeno quanto la Calendar Algebra e permette inoltre di risolvere in maniera elegante i problemi di equivalenza e minimizzazione delle rappresentazioni. Successivamente, Dal Lago e Montanari in [DM01] e Dal Lago, Montanari e Puppis in [DMP03a] hanno proposto alcuni formalismi basati su automi per rappresentare le granularità, che sono espressivi quanto le Granspec e consentono di utilizzare gli algoritmi della teoria degli automi per manipolare tali granularità. In particolare, in [DMP03a] viene illustrato come sia possibile, nei formalismi basati su automi, effettuare mediante appositi algoritmi sia le conversioni tra granularità che la codifica dei termini della Calendar Algebra.

Questa tesi riprende i risultati citati, estendendo i formalismi basati su automi per consentire la rappresentazione di *insiemi di granularità*, come le cosiddette *granularità dinamiche*, ossia granularità non ancorate al dominio temporale sottostante. Il limite principale dei formalismi presentati in [DM01, DMP03a] è infatti quello di non poter rappresentare più di una parola fondamentalmente periodica (ossia, più di una granularità) con un singolo automa. A questo scopo, studieremo i linguaggi ω-regolari riconosciuti dagli automi di Büchi (cfr. [Tho90]), fornendo una caratterizzazione dei linguaggi composti di sole parole fondamentalmente periodiche rappresentabili mediante questa classe di automi. Inoltre, presenteremo una opportuna classe di automi (detti *automi di Büchi fondamentalmente periodici*) che riconosce esattamente tali insiemi e ne studieremo le proprietà di chiusura rispetto alle principali operazioni sui linguaggi. Infine, descriveremo alcuni problemi paradigmatici della teoria degli automi che risultano utili nel contesto delle granularità temporali e mostreremo come sia possibile utilizzare alcuni risultati di decidibilità per gli automi di Büchi per fornire una soluzione ad alcuni di essi. Per il problema dell'equivalenza tra le rappresentazioni forniremo, invece, un algoritmo che consente (sfruttando le soluzioni al problema del partizionamento stabile e della minimizzazione degli automi regolari) di costruire una *forma canonica* per gli automi fondamentalmente periodici.

Il resto della tesi è organizzato come segue.

Nel Capitolo 1 è descritta la notazione utilizzata e vengono definiti formalmente i concetti di granularità e di sistema di granularità. Inoltre, si descrivono alcune relazioni che possono intercorrere tra le granularità e si presenta brevemente il formalismo della Calendar Algebra.

Il Capitolo 2 è dedicato allo studio del formalismo delle Granspec proposto da Wijsen. Si studiano i problemi dell'equivalenza e della minimizzazione delle rappresentazioni in tale formalismo, e la sua espressività viene confrontata con quella della Calendar Algebra.

Nel Capitolo 3 vengono descritti i formalismi basati su automi proposti da Dal Lago, Montanari e Puppis. Si introduce la nozione di *automa single-string* e se ne studiano l'espressività ed il problema dell'equivalenza. Tale classe di automi viene quindi estesa (mediante la definizione degli *automi single-string estesi* e degli *automi etichettati ristretti*), in modo da poter generare rappresentazioni compatte delle granularità temporali. Anche per queste nuove classi di automi vengono studiati l'espressività ed il problema dell'equivalenza.

Nel Capitolo 4 si studiano i linguaggi ω-regolari e si determina una caratterizzazione degli insiemi di parole fondamentalmente periodiche riconoscibili dagli automi di Büchi. Tale studio porta in modo naturale alla definizione di una opportuna classe di automi (*automi fondamentalmente periodici*) che riconosce solamente parole fondamentalmente periodiche ed allo studio delle relative proprietà di espressività e di chiusura.

Il Capitolo 5 è dedicato allo studio di alcuni problemi paradigmatici per gli automi fondamentalmente periodici. Dopo aver definito formalmente tali problemi, mostreremo come alcuni risultati di decidibilità provenienti dalla teoria degli automi di Büchi permettano di risolvere alcuni di essi. Per il problema dell'equivalenza (per il quale non è possibile utilizzare lo stesso approccio degli automi di Büchi) viene presentata una soluzione che si basa sulla creazione di una *forma canonica*, ossia sulla determinazione di un automa unico rispetto al linguaggio riconosciuto. In tal modo il problema dell'equivalenza viene ridotto al confronto tra le forme canoniche degli automi coinvolti.

Nelle conclusioni, infine, sono riassunti i risultati principali della tesi e sono presentati alcuni possibili approfondimenti dei temi trattati. In particolare, si discute sulle possibili estensioni delle operazioni di conversione tra granularità e degli operatori della Calendar Algebra agli insiemi di granularità, si propongono alcune possibili estensioni alla classe degli automi fondamentalmente periodici per renderla più espressiva e si descrivono brevemente due ulteriori approcci alla rappresentazione di insiemi di granularità presenti in letteratura.

Capitolo 1

Granularità temporali

1.1 Notazione utilizzata

In questa sezione descriviamo brevemente la notazione e le convenzioni utilizzate in questo capitolo e nei successivi.

L'insieme dei numeri naturali viene denotato con $\mathbb{N}$ e le principali operazioni aritmetiche e relazioni tra numeri naturali vengono interpretate e rappresentate nel modo usuale. Gli intervalli di numeri interi vengono rappresentati con $[n, m]$, $[n, m[$ e $[n, \infty[$, che indicano rispettivamente gli insiemi $\{k \in \mathbb{N} | n \leq k \leq m\}$, $\{k \in \mathbb{N} | n \leq k < m\}$ e $\{k \in \mathbb{N} | n \leq k\}$. Inoltre se $n > m$ l'intervallo $[n, m]$ rappresenta l'insieme vuoto $\emptyset$.

Se A è un generico insieme finito detto *alfabeto*, le *parole finite* sull'alfabeto A sono tutte le sequenze finite di elementi di A, mentre le *parole infinite*, o ω*-parole*, sull'alfabeto A sono tutte le sequenze infinite a destra di elementi di A. Se w è una generica parola (finita o infinita) sull'alfabeto A indichiamo con $w(1), w(2), \ldots, w(n), \ldots$ l'elemento di A (detto *carattere*) che appare rispettivamente nella prima, seconda, n-esima posizione di w. Gli intervalli di numeri naturali possono essere utilizzati per rappresentare sottosequenze della parola w: per esempio $w[1, 6]$ è la sequenza che contiene i primi sei caratteri di w e $w[5, \infty[$ è la sequenza infinita $w(5)w(6)w(7)\ldots$.

La lunghezza di una parola viene indicata con $|w|$, mentre la parola vuota (cioè di lunghezza zero) viene indicata con ε. L'insieme di tutte le parole finite sull'alfabeto A viene denotato con A^*, mente l'insieme di tutte le parole infinite si indica con A^ω. Con A^+ viene indicato l'insieme $A^* \setminus \{\varepsilon\}$ delle parole finite non vuote su A.

Introduciamo inoltre un nuovo insieme di operatori parametrici $\#_a : A^* \cup A^\omega \to \mathbb{N} \cup \{\infty\}$ che contano il numero di occorrenze del simbolo $a \in A$ di una parola (finita o infinita). Per estensione, definiamo anche gli operatori $\#_{a,b,\ldots} = \#_a + \#_b + \ldots$ e $\#_B = \sum_{b \in B} \#_b$, con $B \subseteq A$. Vale chiaramente l'uguaglianza $\#_A(w) = |w|$ per ogni parola finita o infinita w. Se w è una parola infinita, denotiamo con $\mathrm{Inf}(w)$ l'insieme di tutti i simboli di A che occorrono infinite volte in w, ossia l'insieme $\mathrm{Inf}(w) = \{a \in A | \#_a(w) = \infty\}$.

Il simbolo $\cdot$ indica l'operazione di concatenazione di due parole: se $u \in A^*$ e $v \in A^*$ oppure $v \in A^\omega$ la parola $u \cdot v$ viene ottenuta per giustapposizione di u e v. Se $v \in A^*$ e $V \subseteq A^*$ valgono le seguenti uguaglianze:

- $v^0 = \varepsilon$ e $v^k = v^{k-1} \cdot v$ per ogni $k \in \mathbb{N}$;
- $v^\omega = v \cdot v \cdot v \cdot \ldots$;
- $V^0 = \{\varepsilon\}$;
- $V^n = \{v_1 \cdot v_2 \cdot \ldots \cdot v_n | \forall i \in [1, n].v_i \in V\}$;
- $V^* = \bigcup_{n \in \mathbb{N}} V^n$;
- $V^\omega = \{v_1 \cdot v_2 \cdot \ldots | \forall i \in [1, \infty[.v_i \in V\}$.

Se w è una generica parola (finita o infinita) ed esiste una opportuna parola finita v tale che $w = v^k$, con $k > 1$ o $k = \omega$, allora w è una parola *periodica* e v è un *periodo* di w. In alcuni testi si dice "la parola w ha periodo $|v|$" descrivendo con la parola "periodo" ciò che in questa tesi si denota con "*w ha un periodo di lunghezza $|v|$*". Si definisce *fondamentalmente periodica* ogni parola infinita ottenuta concatenando una ω-parola periodica con un *prefisso* finito. Se w è fondamentalmente periodica allora esistono due parole finite u, v tali che $w = u \cdot v^\omega$; u è detto *prefisso* di w mentre v è detto *periodo* della parola fondamentalmente periodica.

1.2 Definizione di granularità

Per introdurre la definizione matematica di granularità occorre prima definire l'insieme degli istanti temporali utilizzati per definire e rappresentare le relazioni temporali. Tale insieme è detto *dominio temporale* e può essere visto intuitivamente come un insieme ordinato di istanti temporali considerati primitivi.

Definizione 1.1. Un *dominio temporale* è una coppia $\langle T, \leq \rangle$, dove T è un insieme di *istanti temporali* e $\leq$ è una relazione d'ordine totale su T.

Per ogni relazione $\leq$ indichiamo con $<$ la relazione tale che, per ogni coppia di elementi x e y, $x < y$ se e solo se $x \leq y$ e $x \neq y$. Un dominio temporale $\langle T, \leq \rangle$ può essere:

- **limitato superiormente (inferiormente)** se esiste un elemento massimo (minimo) di T rispetto alla relazione $\leq$;
- **discreto** se per ogni elemento $t \in T$ (escluso il massimo) esiste un successore immediato $t' \in T$ (cioè se $\exists t' \in T.t < t' \wedge \nexists t'' \in T.t < t'' < t'$) e se per ogni elemento (escluso il minimo) esiste un predecessore immediato;
- **denso** se per ogni coppia di elementi $t < t'$ appartenenti a T esiste un terzo elemento $t'' \in T$ compreso tra i due (cioè se $\exists t'' \in T.t < t'' < t'$);
- **continuo** se T è isomorfo all'insieme $\mathbb{R}$ dei numeri reali.

Esempio 1.2. Le coppie $\langle \mathbb{N}, \leq \rangle$ e $\langle \mathbb{R}, \leq \rangle$, dove $\leq$ è la relazione d'ordine usuale sui numeri naturali e reali, sono domini temporali. Più precisamente $\langle \mathbb{N}, \leq \rangle$ è un dominio discreto e limitato inferiormente, mentre $\langle R, \leq \rangle$ è un dominio denso e continuo.

Diamo ora la definizione di granularità (cfr. [BJW00, BDE+98, NJW02]).

Definizione 1.3. Una *granularità* (*semplice*) su un dominio temporale $\langle T, \leq \rangle$ è una funzione totale $G : \mathbb{Z} \to \mathcal{P}(T)$ tale che, per ogni $i, j \in \mathbb{Z}$:

1. se $i < j$ e $G(i), G(j) \neq \emptyset$ allora $\forall t \in G(i). \forall t' \in G(j). t < t'$;
2. se $i < j$ e $G(i), G(j) \neq \emptyset$ allora $\forall k \in [i, j]. G(k) \neq \emptyset$.

L'insieme $\mathbb{Z}$, dominio della funzione G, è detto *insieme degli indici* di G. I *granuli* di G sono gli insiemi non vuoti $G(i) \subseteq T$, al variare di $i \in \mathbb{Z}$. Indichiamo come granulo *iniziale* (*finale*) il primo (ultimo) granulo della granularità G: una granularità si dice *limitata superiormente* (*inferiormente*) se possiede un granulo finale (iniziale).

Diciamo che un granulo $G(i)$ *copre* un istante di tempo $t \in T$ se e solo se $t \in G(i)$. Per estensione una granularità G copre l'intero dominio temporale se per ogni istante $t \in T$ esiste un indice $i \in \mathbb{Z}$ tale che il granulo $G(i)$ copre t. Due granuli $G(i), G(j)$ si dicono *contigui* se non esiste $t \in T$ tale che $G(i) < t < G(j)$ (dove con $S < t$ si intende che per ogni $s \in S$, $s < t$ e con $t < S$ si intende che per ogni $s \in S$, $t < s$).

Si può notare come la prima condizione della Definizione 1.3 stabilisca che i granuli di G non si possano sovrapporre e che l'ordine dei loro indici rispetti il loro ordine temporale. La seconda condizione stabilisce che l'insieme degli indici a cui corrispondono insiemi non vuoti di istanti temporali è un intervallo convesso di $\mathbb{Z}$.

La Definizione 1.3 permette facilmente di codificare sia granularità come *Giorno*, *Settimana* o *Mese*, che coprono l'intero dominio temporale, sia granularità limitate come *Anni-dal-2003*, sia granularità con granuli non contigui come *SettimanaLavorativa* o *GiornoLavorativo* e anche granularità con granuli non convessi (ossia granuli per i quali esiste un istante t tale che $\mathrm{Inf}(G(i)) < t < \mathrm{Sup}(G(i))$ e $t \notin G(i)$) come *AnnoLavorativo* o *MeseLavorativo*.

In questa tesi utilizzeremo una rappresentazione grafica per le (porzioni di) granularità in cui i segmenti continui denotano gli istanti temporali coperti da qualche granulo e i segmenti tratteggiati rappresentano gli istanti non coperti dai granuli di G. Le suddivisioni sulla linea del tempo separano i granuli di G, mentre i numeri riportati sono gli indici dei granuli.

Esempio 1.4. In Figura 1.1 riportiamo le rappresentazioni tipiche per le granularità *Giorno*, *GiornoLavorativo*, *Settimana*, *SettimanaLavorativa*, *Mese* e *MeseLavorativo*.

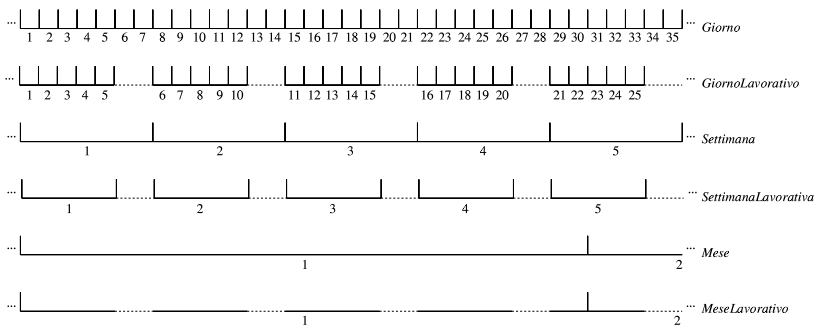

Figura 1.1: Esempi di granularità temporali

Osservazione 1.5. È importante notare che, mentre un dominio temporale può essere discreto, denso o continuo, una granularità definisce sempre un *insieme numerabile di granuli*, ognuno identificato da un numero intero. La definizione di granularità è quindi il primo passo per ottenere una *codifica effettiva* delle strutture temporali e delle loro relazioni. Tuttavia per poter operare in modo effettivo, cioè mediante algoritmi, sulle granularità è necessario restringersi ad un sottoinsieme proprio delle granularità catturate dalla Definizione 1.3; tale insieme contiene solamente le granularità che ammettono rappresentazioni finite.

1.3 Relazioni tra granularità

In questa sezione definiremo le principali relazioni tra granularità temporali che possiedono lo stesso dominio temporale (cfr. [BJW00]). Queste relazioni saranno utili, nelle sezioni e nei capitoli successivi, per costruire rappresentazioni finite di granularità che descrivono le relazioni che intercorrono tra le granularità rappresentate ed una granularità di base fissata.

Definizione 1.6. Una granularità G *si raggruppa in una granularità* H (o, viceversa H *raggruppa* G) se e solo se, per ogni indice $z \in \mathbb{Z}$, esiste un insieme $S_z \subseteq \mathbb{Z}$ (eventualmente infinito) tale che $H(z) = \bigcup_{i \in S_z} G(i)$. Con $G \trianglelefteq H$ si denota che G si raggruppa in H.

La definizione stabilisce che ogni granulo della granularità H debba essere un'unione di granuli della granularità G, tuttavia gli insiemi S_z che contengono gli indici dei granuli di G che si raggruppano nel granulo $H(z)$ possono essere anche intervalli non convessi, e possono esistere granuli di G che non si raggruppano in alcun granulo di H.

Si noti che la relazione $\trianglelefteq$ è un *preordine*, cioè soddisfa le proprietà riflessiva e transitiva.

Esempio 1.7. In Figura 1.2 sono rappresentate due granularità G e H tali che $G \trianglelefteq H$. Si può notare come i granuli $G(4)$ e $G(6)$ non siano inclusi in alcun granulo di H. Tuttavia non esistono granuli di H che contengono istanti temporali non appartenenti a granuli di G.

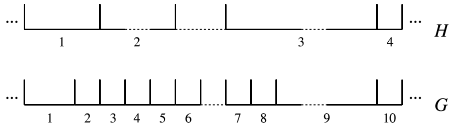

Figura 1.2: Esempio di istanza della relazione $G \trianglelefteq H$

Altri esempi della relazione $\trianglelefteq$ sono le coppie $\langle$*Giorno, SettimanaLavorativa*$\rangle$ e $\langle$*Giorno, MeseLavorativo*$\rangle$.

Definizione 1.8. Una granularità G *raffina una granularità* H se e solo se per ogni indice $i \in \mathbb{Z}$ esiste un indice $z \in \mathbb{Z}$ tale che $G(i) \subseteq H(z)$. Denotiamo questa relazione con $G \preceq H$.

La relazione di raffinamento $\preceq$ è duale rispetto a quella di raggruppamento: ciascun granulo di G è incluso in qualche granulo di H, mentre i granuli di H possono contenere intervalli temporali non coperti da granuli di G. Anche la relazione $\preceq$ gode delle proprietà riflessiva e transitiva.

Esempio 1.9. La Figura 1.3 rappresenta due granularità G e H tali che $G \preceq H$: si noti come i granuli $H(2)$ e $H(4)$ coprano istanti temporali non coperti da G, mentre tutti i granuli di G sono inclusi in qualche granulo di H.

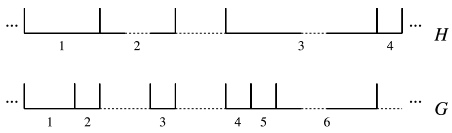

Figura 1.3: Esempio di istanza della relazione $G \preceq H$

Altri esempi della relazione di raffinamento sono le coppie di granularità $\langle$*GiornoLavorativo, Giorno*$\rangle$ e $\langle$*GiornoLavorativo, Mese*$\rangle$.

Definizione 1.10. Una granularità G *partiziona una granularità* H se e solo se G si raggruppa in H e G raffina H ($G \trianglelefteq H$ e $G \preceq H$).

Si può notare che se la granularità G partiziona la granularità H, allora le immagini delle due granularità coincidono, cioè $\bigcup_{i\in\mathbb{Z}} G(i) = \bigcup_{z\in\mathbb{Z}} H(z)$. Nella relazione di partizione si stabilisce infatti una corrispondenza tra granuli di G e granuli di H e viceversa: per ogni granulo di G esiste un granulo di H che lo contiene, mentre ogni granulo di H può essere visto come unione di granuli di G.

Esempio 1.11. Rappresentiamo in Figura 1.4 due granularità G e H tali che G partiziona H. Si può notare come la sola differenza con la Figura 1.2 che rappresenta la relazione $\trianglelefteq$ sia l'assenza dei granuli che violano le condizioni imposte dalla relazione di raffinamento $G \preceq H$.

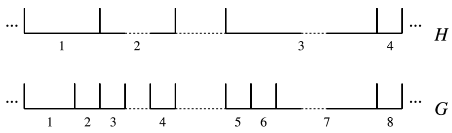

Figura 1.4: Esempio di istanza della relazione G partiziona H

Esempi di istanze della relazione di partizione sono le coppie $\langle$*Giorno*, *Settimana*$\rangle$ e $\langle$*Mese, Anno*$\rangle$.

Definizione 1.12. Una granularità G *è sottogranularità di* H se e solo se per ogni indice $i \in \mathbb{Z}$ esiste un indice $z \in \mathbb{Z}$ tale che $G(i) = H(z)$. Questa relazione si denota con $G \sqsubseteq H$.

Si noti che vale l'implicazione $G \sqsubseteq H \Rightarrow G \preceq H$ e quindi la relazione $\sqsubseteq$ gode delle proprietà transitiva e riflessiva.

Esempio 1.13. In Figura 1.5 è rappresentata un'istanza della relazione $G \sqsubseteq H$.

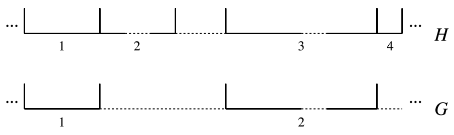

Figura 1.5: Esempio di istanza della relazione $G \sqsubseteq H$

Esempi di istanze della relazione di sottogranularità sono le coppie $\langle$*Giorno Lavorativo, Giorno*$\rangle$ e $\langle$*Maggio, Mese*$\rangle$.

Definizione 1.14. Una granularità G *è coperta da una granularità* H (o viceversa, H *copre* G) se e solo se l'immagine di G è contenuta nell'immagine di H, cioè se vale l'inclusione $\bigcup_{i\in\mathbb{Z}} G(i) \subseteq \bigcup_{z\in\mathbb{Z}} H(z)$. Questa relazione viene indicata con il simbolo $\widehat{\subseteq}$.

Anche la relazione di copertura gode delle proprietà transitiva e riflessiva ed è quindi un preordine.

Esempio 1.15. Le coppie $\langle$*Settimana, Mese*$\rangle$ e $\langle$*GiornoLavorativo, Anno*$\rangle$ sono esempi di istanze della relazione $\widehat{\subseteq}$.

Definizione 1.16. Una granularità G *è trasposizione di una granularità* H se e solo se esiste un intero $k \in \mathbb{Z}$ tale che, per ogni indice $i \in \mathbb{Z}$, $G(i) = H(i + k)$. Indicheremo con $G \rightleftharpoons H$ questa relazione.

È possibile che granularità differenti rappresentino la stessa suddivisione temporale assegnando allo stesso granulo indici differenti. Può essere utile considerare tali granularità come distinte, ma in generale si preferisce considerarle "equivalenti", a meno di una traslazione degli indici. È facile verificare che la relazione di trasposizione è una relazione di equivalenza poiché gode delle proprietà riflessiva, simmetrica e transitiva, ed associa le granularità che rappresentano la stessa suddivisione temporale a meno di traslazione degli indici. Inoltre vale $G \rightleftharpoons H$ se e solo se $G \sqsubseteq H$ e $H \sqsubseteq G$.

Osservazione 1.17. Se si indica con $[G]_{\rightleftharpoons}$ la classe di equivalenza di G rispetto alla relazione di trasposizione (cioè l'insieme $\{H | H \rightleftharpoons G\}$) e si estendono le relazioni $\trianglelefteq$, $\preceq$ e $\sqsubseteq$ all'insieme quoziente (cioè all'insieme $\{[G]_{\rightleftharpoons} | G$ è una granularità$\}$), si ottengono tre *relazioni d'ordine parziale* che godono delle proprietà riflessiva, transitiva e antisimmetrica.

Definiamo ora due particolari istanze della relazione di raggruppamento $\trianglelefteq$ in cui la granularità H raggruppa G in modo periodico (o fondamentalmente periodico). Queste definizioni, anche se più complicate della definizione generale della relazione $\trianglelefteq$, permetteranno di catturare le granularità rappresentabili come composizioni periodiche (o fondamentalmente periodiche) dei granuli di G.

Definizione 1.18. Una granularità G *si raggruppa periodicamente in* H (o, viceversa, H *raggruppa periodicamente* G) se e solo se valgono le seguenti condizioni:

- $G \trianglelefteq H$;

- esistono due interi positivi q ed r tali che r è inferiore al numero di granuli di H e, per ogni coppia di interi z e n, se vale $H(z) = \bigcup_{k=1}^{n} G(i_k)$ e $H(z+r) \neq \emptyset$, allora $H(z+r) = \bigcup_{k=1}^{n} G(i_k + q)$.

La relazione così definita si indica con il simbolo $\trianglelefteq^p$.

La relazione di raggruppamento periodico è un caso particolare della relazione di raggruppamento caratterizzata dalla ripetizione periodica del "pattern di raggruppamento" dei granuli di G nei granuli di H. Infatti, poiché G si raggruppa in H, ogni granulo $H(z)$ è l'unione di alcuni granuli di G; la periodicità è stabilita dalla seconda condizione della Definizione 1.18: essa afferma che, se esiste il successore r-esimo del granulo $H(z)$ (cioè se $H(z+r) \neq \emptyset$), allora $H(z+r)$ raggruppa i granuli di G nello stesso modo di $H(z)$. Questo definisce un pattern di raggruppamento dei granuli di G in H che viene ripetuto dopo r granuli di H, a cui corrispondono esattamente q granuli di G. Il valore di q è detto *periodo* della relazione $\trianglelefteq^p$.

Esempio 1.19. In Figura 1.6 sono rappresentate due granularità G e H tali che $G \trianglelefteq^p H$ con parametri $q = 6$ e $r = 2$.

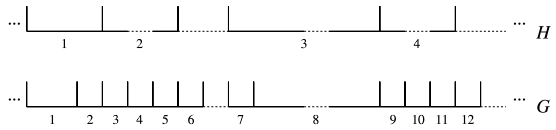

Figura 1.6: Esempio di istanza della relazione $G \trianglelefteq^p H$

Osservazione 1.20. La relazione $\trianglelefteq^p$ descritta dalla Definizione 1.18 garantisce l'esistenza di una rappresentazione finita di H in termini di granuli di G. In particolare, tale rappresentazione può essere data fornendo le seguenti informazioni:

- i valori di q e r;
- il minimo intervallo I che contiene gli indici dei granuli di H;
- gli insiemi finiti $S_0, S_1, \ldots, S_{r-1}$ di indici di G che descrivono la composizione di ognuno degli r granuli che si ripetono in H.

La descrizione della granularità H si ottiene quindi con la seguente formula:

$$H(z) = \bigcup_{i \in S_{(z \bmod r)}} G(i + \lfloor z/r \rfloor \cdot q), \text{ per ogni } z \in I.$$

Nel caso dell'Esempio 1.19 si ha che la granularità H è codificata dai parametri $q = 6$, $r = 2$, $S_0 = \{1, 2\}$, $S_1 = \{3, 5\}$ e da un opportuno intervallo I di interi.

Esistono inoltre granularità che sono intuitivamente periodiche rispetto ad altre granularità ma per le quali esistono alcune "eccezioni" (cioè segmenti temporali) dove la periodicità non viene rispettata. È quindi utile estendere la Definizione 1.18 anche ai casi in cui la periodicità ammette un numero finito di eccezioni.

Definizione 1.21. Una granularità G *si raggruppa periodicamente con finite eccezioni in* H (o, viceversa, H *raggruppa periodicamente con finite eccezioni* G) se e solo se valgono le seguenti condizioni:

- $G \trianglelefteq H$;
- esiste un numero finito di intervalli convessi $E_1, E_2, \ldots, E_s$ (detti *eccezioni*) contenenti un numero finito di indici appartenenti a $\mathbb{Z}$, ed esistono due interi positivi q e r tali che r è inferiore al minimo numero di granuli compresi tra due eccezioni e, per ogni coppia di interi z e n, se valgono le relazioni $H(z) = \bigcup_{k=1}^{n} G(i_k)$, $H(z+r) \neq \emptyset$ e $\forall x.E_x \cap [z, z+r] = \emptyset$, allora $H(z+r) = \bigcup_{k=1}^{n} G(i_k + q)$.

Indichiamo con il simbolo $\trianglelefteq^{fp}$ la relazione appena definita.

La relazione $\trianglelefteq^{fp}$ consente la presenza di un numero finito di eccezioni (che coprono un insieme finito di granuli), ma richiede che negli intervalli tra due eccezioni successive G si raggruppi in H in modo periodico rispettando un pattern di raggruppamento analogo a quello definito dalla relazione di raggruppamento periodico $\trianglelefteq^{p}$.

Esempio 1.22. Sono rappresentate in Figura 1.7 due granularità G e H tali che $G \trianglelefteq^{fp} H$ con parametri $s = 1$, $E_1 = \{4\}$, $q = 2$ e $r = 1$.

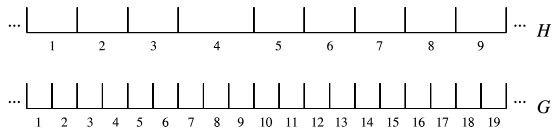

Figura 1.7: Esempio di istanza della relazione $G \trianglelefteq^{fp} H$

Osservazione 1.23. Anche la relazione $\trianglelefteq^{fp}$ garantisce l'esistenza di una rappresentazione finita per H in termini dei granuli di G, in maniera simile a quanto visto per la relazione di raggruppamento periodico senza eccezioni. In questo caso la rappresentazione può essere data fornendo:

- i valori di s, q e r;
- gli intervalli $E_1, \ldots, E_s$ che rappresentano le eccezioni;
- le rappresentazioni dei granuli di H (in termini dei granuli di G) che hanno indice compreso in qualche eccezione E_x;
- il minimo intervallo I che contiene tutti i granuli di H;
- gli insiemi finiti $S_0, S_1, \ldots, S_{r-1}$ di indici di G che descrivono la composizione dei granuli delle parti periodiche di H.

La definizione della struttura dei granuli di H rispetto a quelli di G è simile a quella data nell'Osservazione 1.20, ma più complicata per la presenza delle eccezioni, ed evitiamo quindi di riportarla nel dettaglio.

1.4 Sistemi di granularità

Il termine *sistema di granularità* indica genericamente un insieme strutturato di granularità sullo stesso dominio temporale.

In questa sezione definiremo alcune proprietà rilevanti dei sistemi di granularità, descriveremo le condizioni per cui tali proprietà sono soddisfatte e forniremo un esempio di sistema di granularità. Infine formalizzeremo la nozione di calendario e ne daremo un esempio.

Definizione 1.24. Fissato un insieme C di granularità sullo stesso dominio temporale e una relazione θ su di esso, una granularità $G \in C$ si dice *minima* (rispettivamente *massima*) *rispetto a* $\langle C, \theta \rangle$ se e solo se per ogni $H \in C$ vale la relazione $G\theta H$ (rispettivamente $H\theta G$).

Per esempio, dato l'insieme di tutte le granularità definite sul dominio temporale $\langle \mathbb{R}, \leq \rangle$ e la relazione tra granularità $\preceq$, la granularità vuota è minima, mentre la granularità costituita da un unico granulo che copre tutto il dominio temporale è massima. Una biezione da $\mathbb{Z}$ ad un dominio temporale discreto è invece minima rispetto alla relazione $\trianglelefteq$. Ovviamente, fissata una relazione tra granularità θ, esistono insiemi che non possiedono minimo o massimo: per esempio l'insieme $\{$*Settimana, Mese*$\}$ è privo di una granularità minima rispetto a $\trianglelefteq$ e rispetto a $\preceq$.

Osservazione 1.25. Nel caso in cui θ è una relazione d'ordine parziale, le granularità minime e massime, se esistono, sono uniche.

Definizione 1.26. Un insieme di granularità su uno stesso dominio temporale C forma un *reticolo* rispetto a una relazione d'ordine parziale tra granularità θ se e solo se per ogni coppia di granularità esistono l'estremo inferiore e l'estremo superiore rispetto a θ. Se per ogni coppia di granularità l'estremo superiore e l'estremo inferiore *appartengono a* C, C è detto *reticolo chiuso*.

Se C è un reticolo rispetto alla relazione θ e G_1 e G_2 sono due granularità di C, indichiamo l'estremo inferiore e l'estremo superiore della coppia G_1, G_2 rispettivamente con $\mathrm{Inf}(G_1, G_2)$ e $\mathrm{Sup}(G_1, G_2)$.

Si noti che la Definizione 1.26 è la definizione usuale di reticolo, e che esempi di ordini parziali su granularità sono le relazioni $\trianglelefteq$ e $\preceq$ estese all'insieme quoziente di $\rightleftharpoons$. L'insieme quoziente di tutte le granularità sul dominio temporale $\langle \mathbb{R}, \leq \rangle$ è quindi un reticolo chiuso rispetto alla relazione $\preceq$.

1.4.1 Vincoli sui sistemi di granularità

Quando si utilizza la nozione di sistema di granularità in una particolare applicazione o sistema formale, è necessario specializzare la nozione di granularità per poter garantire che il sistema goda delle proprietà definite in precedenza. In particolare si possono porre restrizioni

- sull'*insieme degli indici*;
- sul *dominio temporale*;
- sulla *struttura dei granuli*;
- sulle *relazioni tra le granularità*.

Consideriamo ora alcune possibilità per ognuna delle categorie. È possibile imporre, per esempio, che gli indici dei granuli siano interi positivi (imponendo che $G(i) = \emptyset$ per ogni $i \leq 0$) anziché tutti i numeri interi. Tale scelta sarà generalmente applicata nei capitoli successivi, quando mostreremo come sia possibile rappresentare tutte le istanze della relazione $\trianglelefteq^{fp}$ e alcune istanze della relazione $\trianglelefteq$ mediante parole fondamentalmente periodiche.

Le scelte possibili sul dominio temporale sono tipicamente tra domini discreti e domini densi. In generale, se si vuole rappresentare un insieme di granularità che contiene una granularità minima rispetto alla relazione $\trianglelefteq$ la scelta di un dominio temporale discreto è probabilmente la migliore; mentre se si vogliono rappresentare granularità arbitrariamente fini è necessario utilizzare un dominio denso.

La struttura dei granuli può invece essere ristretta in diversi modi:

1. escludendo le granularità con granuli non convessi
   (ad esempio, *MeseLavorativo*, *AnnoLavorativo*, etc. . .);
2. escludendo le granularità con granuli non contigui
   (ad esempio, *GiornoLavorativo*, *SettimanaLavorativa*, etc. . .);
3. escludendo le granularità che non coprono l'intero dominio temporale;
4. escludendo le granularità con granuli di dimensione non uniforme
   (ad esempio, *Mese*, *Anno*, etc. . .).

I sistemi di granularità possono presentare infine vincoli sulle relazioni tra le granularità, in particolare è possibile:

5. escludere le trasposizioni di granularità
   (imponendo che non esistano granularità distinte G e H tali che $G \rightleftharpoons H$);
6. imporre che le granularità siano confrontabili rispetto ad una relazione θ
   (ad esempio, *Settimana* e *Mese* non sono confrontabili rispetto alle relazioni $\trianglelefteq$ e $\preceq$ e non possono comparire insieme in un sistema in cui vale questo vincolo).

Nella sezione successiva si farà riferimento ai vincoli presentati con la loro numerazione.

1.4.2 Proprietà dei sistemi di granularità

Si possono provare molte proprietà formali interessanti per la classe dei sistemi di granularità per i quali si impone il vincolo 5 Garantendo che non esistano coppie di granularità che sono equivalenti per la relazione di trasposizione $\rightleftharpoons$ le tre relazioni $\trianglelefteq$, $\preceq$ e $\sqsubseteq$ sono ordini parziali per il sistema e possono essere usate per confrontare granularità.

In [BJW00] Bettini, Jajodia e Wang studiano in particolare la relazione di raffinamento $\preceq$: se il sistema di granularità soddisfa il vincolo 5 esiste sia un unico massimo $G_\top$ che un unico minimo $G_\bot$. $G_\top$ è composto da un unico granulo (con indice opportuno) che copre tutto il dominio temporale, mentre $G_\bot$ è la granularità composta da soli granuli vuoti ($G_\bot(i) = \emptyset$ per ogni indice $i \in \mathbb{Z}$). Inoltre il sistema di granularità è un reticolo, come enunciato nel teorema seguente.

Teorema 1.27. *Ogni sistema di granularità C tale che per ogni coppia di granularità $G_1, G_2 \in C$ vale l'implicazione $G_1 \rightleftharpoons G_2 \Rightarrow G_1 = G_2$ è un reticolo rispetto alla relazione di raffinamento $\preceq$.*

La dimostrazione del teorema costruisce, a partire da una coppia di granularità G_1, G_2 di C, l'estremo superiore $\mathrm{Sup}(G_1, G_2)$ e l'estremo inferiore $\mathrm{Inf}(G_1, G_2)$ della coppia. L'esistenza dei due estremi dimostra che il sistema di granularità è un reticolo.

Viene mostrato inoltre nel seguente Teorema come sia possibile, applicando ulteriori vincoli, ottenere un sistema di granularità che è un *reticolo chiuso* rispetto alla relazione $\preceq$. In tali sistemi gli estremi superiore e inferiore di ogni coppia di granularità sono a loro volta granularità appartenenti al sistema.

Teorema 1.28. *Un sistema di granularità C che rispetti il vincolo* 5 *ed una ulteriore combinazione dei vincoli* 1, 2, 3, 4, 6 *è un reticolo chiuso rispetto alla relazione $\preceq$ se*

- *il vincolo* 2 *è rispettato se e solo se è rispettato anche il vincolo* 3 *o* 6*;*
- *il vincolo* 4 *è rispettato se e solo se è rispettato anche il vincolo* 6*.*

Osservazione 1.29. Nel Teorema 1.28 si fa riferimento a particolari combinazioni dei vincoli 1, 2, 3, 4, 5 e 6 presentati nella sezione precedente per garantire la chiusura del sistema di granularità. Esistono tuttavia sistemi di granularità che applicano combinazioni diverse dei vincoli, oppure vincoli diversi da quelli presentati, ma che conservano lo stesso la proprietà di reticolo chiuso: in tal caso l'esistenza degli estremi Inf e Sup e la loro chiusura rispetto al sistema va verificata caso per caso, fornendo una loro costruzione opportuna e dimostrandone l'appartenenza al sistema.

Diamo ora un esempio di sistema di granularità che soddisfa la proprietà di reticolo chiuso, e che viene utilizzato in [BJW00] come sistema di base per tutte le applicazioni presentate.

Definizione 1.30. Chiamiamo *sistema generale delle granularità sui reali* (abbreviato in *GGR*) l'insieme contenente tutte le granularità sul dominio temporale $\langle \mathbb{R}, \leq \rangle$ tali che:

- $G(i) = \emptyset$ per ogni $i \leq 0$;
- $G(i) = \emptyset \Rightarrow G(i+1) = \emptyset$ per ogni $i > 0$.

La prima condizione della definizione di *GGR* impone che gli indici delle granularità siano solamente i numeri naturali maggiori di zero e, assieme alla seconda condizione, impone che l'indice del primo granulo di ogni granularità sia sempre 1.

In *GGR* non possono quindi esistere coppie di granularità che siano equivalenti per trasposizione (vale quindi il vicolo 5), e le relazioni $\preceq$, $\trianglelefteq$, $\sqsubseteq$ sono ordini parziali. Il sistema non impone ulteriori vincoli sulla struttura dei granuli e quindi, per i Teoremi 1.27 e 1.28, *GGR* è un reticolo chiuso rispetto alla relazione $\preceq$.

1.4.3 Definizione di Calendario

Un'altra relazione molto usata per confrontare le granularità di un sistema, oltre alla relazione $\preceq$ vista in precedenza, è la relazione di raggruppamento $\trianglelefteq$.

In particolare siamo interessati a studiare i sistemi (finiti) di granularità per i quali esiste una granularità minima $G_\perp$ per la relazione $\trianglelefteq$: in tal caso possiamo utilizzare un insieme numerabile (per esempio, gli interi o i naturali) per codificare tale granularità, e codificare le altre granularità del sistema rappresentando il modo in cui i loro granuli raggruppano i granuli di $G_\perp$. Queste osservazioni portano alla definizione di *calendario*.

Definizione 1.31. Un sistema finito di granularità che include una granularità minima $G_\perp$ rispetto alla relazione $\trianglelefteq$ è detto *calendario*.

Esempio 1.32. La Figura 1.8 rappresenta un calendario C la cui granularità minima è *Secondo*. Gli archi tra le granularità rappresentano le istanze della relazione $\trianglelefteq$: si noti come ogni granularità di C diversa da *Secondo* ha un predecessore immediato.

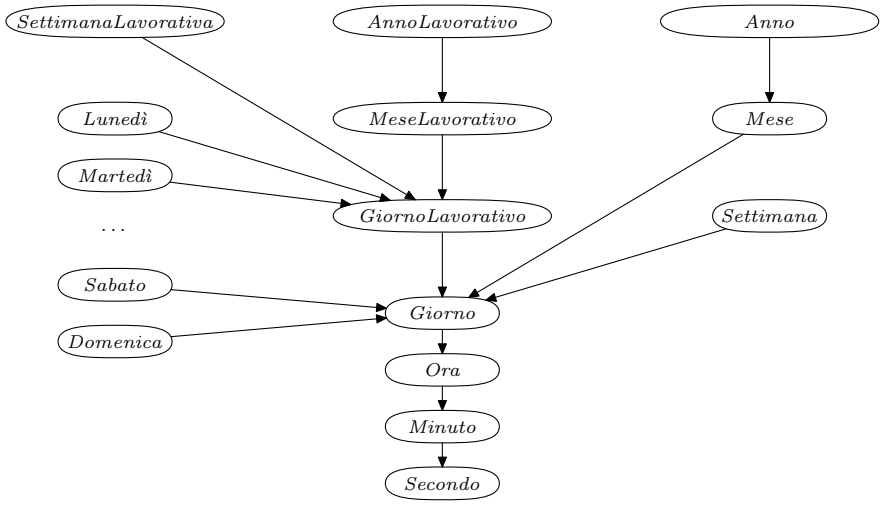

Figura 1.8: Rappresentazione della relazione $\trianglelefteq$ in un calendario

1.4.4 Conversioni tra granularità

Spesso, nella manipolazione delle informazioni associate a diverse granularità temporali, è necessario operare conversioni da una granularità all'altra. Ad esempio, è possibile voler conoscere il giorno della settimana in cui cade una ricorrenza, oppure conoscere la distanza in giorni che intercorre tra due date, o sapere la durata in giorni di un determinato mese dell'anno. Altre operazioni coinvolgono direttamente le informazioni associate alle granularità temporali: se per esempio in una tabella di una base di dati temporale vengono memorizzate i dati sulla quantità di pioggia caduta per ogni giorno dell'anno, una interrogazione che richiedere di calcolare la quantità di pioggia caduta nell'ultimo mese necessita di rielaborare le informazioni in termini di mesi. Tutte le operazioni di questo genere sono dette *conversioni tra granularità*. In [BJW00], Bettini et al. propongono alcune semplici operazioni di base che permettono di risolvere i problemi di conversione tra granularità che illustreremo in questa sezione.

Definiamo dunque le operazioni di conversione $\lceil i \rceil_G^H$ e $\lfloor z \rfloor_G^H$. La prima operazione restituisce l'indice del granulo di H che include interamente il granulo di G di indice i, mentre la seconda funzione determina l'insieme degli indici dei granuli di G la cui unione produce esattamente il granulo di H di indice z.

Definizione 1.33. Se H e G sono due granularità, indichiamo con $\lceil\ \rceil_G^H$ la funzione di conversione tale che, per ogni indice $i \in \mathbb{Z}$, $\lceil i \rceil_G^H = z$ se e solo se vale $G(i) \subseteq H(z)$, altrimenti $\lceil i \rceil_G^H$ è indefinita (useremo in tal caso la notazione $\lceil i \rceil_G^H \uparrow$).

Per esempio, l'operazione $\lceil i \rceil_{Settimana}^{Mese}$ restituisce, se esiste, l'indice del mese che include interamente l'i-esima settimana:

$$\begin{cases} \lceil 1 \rceil_{Settimana}^{Mese} = 1, \\ \lceil 2 \rceil_{Settimana}^{Mese} = 1, \\ \lceil 3 \rceil_{Settimana}^{Mese} = 1, \\ \lceil 4 \rceil_{Settimana}^{Mese} = 1, \\ \lceil 5 \rceil_{Settimana}^{Mese} \uparrow, \\ \lceil 6 \rceil_{Settimana}^{Mese} = 2, \\ \dots \end{cases}$$

Definizione 1.34. Se H e G sono due granularità, indichiamo con $\lfloor\ \rfloor_G^H$ la funzione di conversione tale che, per ogni indice $z \in \mathbb{Z}$, $\lfloor z \rfloor_G^H = (i, n)$ se e solo se $H(z) = \bigcup_{k \in [i, i+n[} G(k)$ e $G(i), G(i+n-1) \neq \emptyset$, altrimenti la funzione è indefinita.

La funzione $\lfloor z \rfloor_G^H$ restituisce quindi una coppia (i, n) che stabilisce l'indice i del primo granulo e il numero n dei granuli di G che compongono $H(z)$. Per esempio, l'operazione $\lfloor z \rfloor_{Giorno}^{Mese}$ restituisce il giorno di inizio ed il numero di giorni del mese identificato da z:

$$\begin{cases} \lfloor 1 \rfloor_{Giorno}^{Mese} = (1, 31) \\ \lfloor 2 \rfloor_{Giorno}^{Mese} = (32, 28) \\ \dots \end{cases}$$

1.5 Calendar Algebra

In questa sezione studieremo un formalismo simbolico, chiamato *Calendar Algebra*, che può essere utilizzato per rappresentare una vasta classe di granularità temporali in maniera naturale e compatta. L'idea sottostante questo formalismo è l'osservazione che le granularità incluse in un sistema non sono generalmente isolate, ma sono piuttosto strettamente correlate. Per esempio, nel calendario di Figura 1.8 esiste una granularità minima di base (*Secondo*) e le altre granularità possono essere costruite a partire da essa attraverso opportune operazioni. Il formalismo della Calendar Algebra utilizza alcune operazioni simboliche (dette *operatori*) per catturare le relazioni tra le coppie di granularità di un calendario, e per definire nuove granularità in funzione di quelle preesistenti.

Gli operatori della Calendar Algebra si dividono in due categorie: gli operatori "grouping-oriented", che combinano i granuli degli operandi per formare

nuove granularità, e gli operatori "granule-oriented", che generano nuove granularità selezionando i granuli degli operandi che soddisfano determinate proprietà. In questa sezione, per semplificare le definizioni degli operatori, estenderemo la definizione di granularità data in precedenza, quindi presenteremo i principali operatori della Calendar Algebra e presenteremo alcuni risultati di espressività.

Questo formalismo non è adatto a risolvere il problema dell'equivalenza tra le rappresentazioni, quindi questo aspetto sarà trattato solamente nei capitoli successivi, dopo aver introdotto i formalismi che rappresentano le granularità mediante parole fondamentalmente periodiche. Per una trattazione completa della Calendar Algebra si veda [BJW00].

1.5.1 Granularità etichettate

Estendiamo la definizione di granularità associando a ciascun granulo un'etichetta.

Definizione 1.35 (Granularità etichettata). Una *granularità etichettata* (o *generalizzata*) su un dominio temporale $\langle T, \leq \rangle$ è una coppia $\langle \mathcal{L}_G, G \rangle$, dove $\mathcal{L}_G$ è un sottoinsieme di numeri interi (detto *etichettatura di* G) e G è una funzione totale $G : \mathcal{L}_G \to \mathcal{P}(T)$ tale che, per ogni $i, j \in \mathcal{L}_G$:

- se $i < j$ e $G(i), G(j) \neq \emptyset$ allora $\forall t \in G(i). \forall t' \in G(j). t < t'$;
- se $i < j$ e $G(i), G(j) \neq \emptyset$ allora $\forall k \in [i, j]. G(k) \neq \emptyset$.

Si noti come la definizione di granularità etichettata catturi anche tutte le granularità che soddisfano la Definizione 1.3 (basta semplicemente porre $\mathcal{L}_G = \mathbb{Z}$): chiameremo *granularità semplici* tali granularità e *granularità etichettate* (oppure semplicemente *granularità*, qualora ciò non sia ambiguo) quelle date dalla Definizione 1.35.

Le definizioni delle relazioni $\trianglelefteq$, $\preceq$, $\sqsubseteq$, $\widehat{\subseteq}$, $\rightleftharpoons$, $\trianglelefteq^p$ e $\trianglelefteq^{fp}$ possono essere estese alle granularità etichettate semplicemente sostituendo a $\mathbb{Z}$ l'insieme delle etichette delle rispettive granularità. Anche le funzioni di conversione $\lceil \, \rceil_G^H$ e $\lfloor \, \rfloor_G^H$ possono essere estese al caso in cui G e H sono granularità etichettate, facendole operare sugli insiemi $\mathcal{L}_G$ e $\mathcal{L}_H$ anziché sull'insieme $\mathbb{Z}$. Nel caso della relazione $\subseteq$ è utile fare riferimento alla seguente variante.

Definizione 1.36. Una granularità etichettata $\langle \mathcal{L}_G, G \rangle$ *è sottogranularità allineata di* $\langle \mathcal{L}_H, H \rangle$ se e solo se $\mathcal{L}_G \subseteq \mathcal{L}_H$ e, per ogni indice $i \in \mathcal{L}_G$, vale l'implicazione $G(i) \neq \emptyset \Rightarrow G(i) = H(i)$.

Esempio 1.37. In Figura 1.9 sono rappresentate due granularità G e H tali che G è sottogranularità allineata di H.

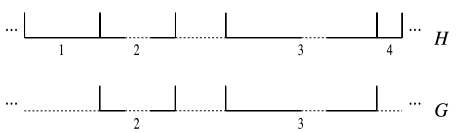

Figura 1.9: Esempio di istanza della relazione "sottogranularità allineata"

1.5.2 Operatori grouping-oriented

Definizione 1.38 (Operatore *Group*). Se G è una granularità semplice e m un intero positivo, l'espressione $Group_m(G)$ rappresenta la granularità H tale che per ogni indice x, $H(x) = \bigcup_{y \in [(x-1)m+1, xm]} G(y)$.

Per esempio, vale l'espressione *Settimana*= $Group_7(Giorno)$, come mostrato in Figura 1.10, se si assume che il giorno etichettato con 1 sia il primo della settimana.

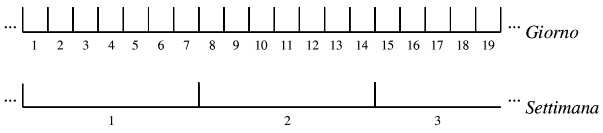

Figura 1.10: Esempio di applicazione dell'operatore *Group*

Definizione 1.39 (Operatore *AlterTick*). Se G e H sono granularità semplici e l, k, m interi tali che G partiziona H e $1 \leq l \leq m$, l'operazione $AlterTick^m_{l,k}(G, H)$ genera una granularità semplice H' tale che

$$H'(z) = \begin{cases} \emptyset & \text{se } H(z) = \emptyset, \\ \bigcup_{i \in [b'_z, t'_z[} G(i) & \text{altrimenti,} \end{cases}$$

dove:

- $H(z) = \bigcup_{i \in [b_z, t_z[} G(i)$,
- $b'_z = \begin{cases} b_z + (h-1) \cdot k & \text{se } z = (h-1) \cdot m + l, \\ b_z + h \cdot k & \text{altrimenti,} \end{cases}$
- $t'_z = t_z + h \cdot k$.

L'operazione $AlterTick^m_{l,k}(G, H)$ genera una nuova granularità espandendo (se $k > 0$) o contraendo (se $k < 0$) i granuli di H in modo periodico. Ad esempio la Figura 1.11 rappresenta la granularità H' generata dall'operazione $AlterTick^3_{2,-1}(G, H)$.

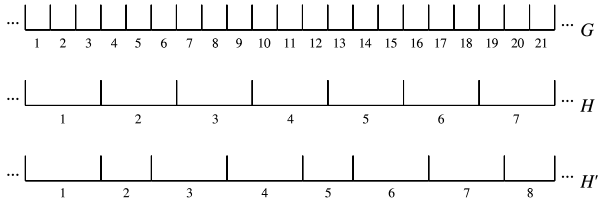

Figura 1.11: Esempio di applicazione dell'operatore *AlterTick*

Definizione 1.40 (Operatore *Shift*). Se G è una granularità semplice e m un intero, l'operazione $Shift_m(G)$ genera una granularità semplice H tale che $H(z) = G(z + m)$ per ogni $z \in \mathbb{Z}$.

L'operatore *Shift* genera dunque granularità che sono trasposizione di quelle fornite in input. Vale quindi la relazione $G \rightleftharpoons \mathit{Shift}_m(G)$ per ogni G e m.

Definizione 1.41 (Operatore *Combine*). Se $\langle \mathcal{L}_H, H\rangle$ e $\langle \mathcal{L}_G, G\rangle$ sono granularità etichettate, l'operazione $\mathit{Combine}(H, G)$ genera una granularità etichettata $\langle \mathcal{L}_{H'}, H'\rangle$ raggruppando tutti i granuli di G che sono inclusi in un granulo di H. Formalmente $\langle \mathcal{L}_{H'}, H'\rangle$ è tale che per ogni indice z, $H'(z) = \bigcup_{G(i)\subseteq H(z)} G(i)$.

Per esempio la granularità *MeseLavorativo* può essere rappresentata dal termine simbolico $\mathit{Combine}(\mathit{Mese}, \mathit{GiornoLavorativo})$, come mostrato in Figura 1.12.

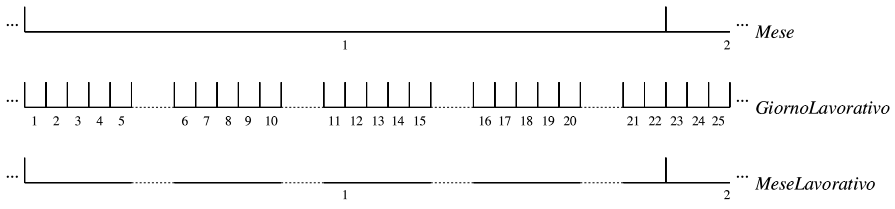

Figura 1.12: Esempio di applicazione dell'operatore *Combine*

Definizione 1.42 (Operatore *AnchoredGroup*). Se G è una granularità semplice (se cioè $\mathcal{L}_G = \mathbb{Z}$) e $\langle \mathcal{L}_H, H\rangle$ è una sottogranularità allineata di G, l'operazione $\mathit{AnchoredGroup}(G, H)$ genera una granularità etichettata H' tale che $\mathcal{L}_{H'} = \mathcal{L}_H$ e $H'(z) = \bigcup_{i\in[z,succ(z)[} G(i)$, dove $succ(z)$ è l'etichetta del granulo immediatamente successivo a $H(z)$.

Ad esempio, la granularità *SettimanaLavorativa* è rappresentata dal termine simbolico $\mathit{AnchoredGroup}(\mathit{GiornoLavorativo}, \mathit{Lunedì})$.

1.5.3 Operatori granule-oriented

Definizione 1.43 (Operatore *Subset*). Se $\langle \mathcal{L}_G, G\rangle$ è una granularità etichettata e m, n due interi tali che $m \leq n$, l'operazione $\mathit{Subset}^n_m(G)$ genera una granularità etichettata $\langle \mathcal{L}_H, H\rangle$ tale che $\mathcal{L}_H = \mathcal{L}_G \cap [m, n]$ e $H(z) = G(z)$ per ogni $z \in \mathcal{L}_H$.

L'operatore *Subset* seleziona quindi un intervallo di granuli della granularità in input: per esempio è possibile definire la sottogranularità Anno_{20} di *Anno*, che contiene solamente gli anni del ventesimo secolo, come $\mathit{Anno}_{20} = \mathit{Subset}^{2000}_{1901}(\mathit{Anno})$.

Introduciamo ora tre operatori binari di *selezione* (*SelectDown*, *SelectUp* e *SelectByIntersect*) che generano granularità selezionando i granuli del primo operando secondo le loro relazioni con i granuli del secondo operando. Per facilitare la definizione degli operatori introduciamo la seguente notazione: per ogni insieme finito S di interi ordinati $x_1 < x_2 < \ldots < x_n$, definiamo l'insieme $\Delta^l_k(S)$ come $\{x_i \in S.k \leq i < k + l\}$ se k e l sono interi positivi. Se k è negativo si definisce $\Delta^l_k = \{x_i \in S.|S| + k - l + 2 \leq i < |S| + k\}$.

Intuitivamente l'insieme $\Delta^l_k(S)$ contiene i primi l valori di S a partire dal k-esimo elemento se $k > 0$, e i primi l valori che si incontrano visitando all'indietro

S partendo dalla posizione $|S| + k + 1$. Per esempio, $\Delta^2_3(\{1,2,3,4,5,6,7\}) = \{3,4\}$ e $\Delta^2_{-3}(\{1,2,3,4,5,6,7\}) = \{4,5\}$.

Definizione 1.44 (Operatore *SelectDown*). Se $\langle \mathcal{L}_G, G\rangle$ e $\langle \mathcal{L}_H, H\rangle$ sono granularità etichettate e k, l due numeri interi tali che $k \neq 0$ e $l > 0$, l'operazione $SelectDown^l_k(G,H)$ genera una granularità etichettata $\langle \mathcal{L}_{H'}, H'\rangle$ tale che $\mathcal{L}_{H'} = \bigcup_{i\in\mathcal{L}_H} \Delta^l_k(\{z \in \mathcal{L}_G.\emptyset \neq G(z) \subseteq H(i)\})$ e $H'(z) = G(z)$ per ogni $z \in \mathcal{L}_{H'}$.

Intuitivamente l'operazione *SelectDown* seleziona l granuli a partire dal k-esimo di ogni insieme di granuli di G contenuto in un granulo di H. Per esempio, le granularità *Lunedì*, *Martedì*, ..., *Domenica* sono rappresentabili usando *SelectDown* nel modo seguente:

$$\begin{cases} Lunedì = SelectDown^1_1(Giorno, Settimana), \\ Martedì = SelectDown^1_2(Giorno, Settimana), \\ \ldots \\ Domenica = SelectDown^1_7(Giorno, Settimana). \end{cases}$$

Definizione 1.45 (Operatore *SelectUp*). Se $\langle \mathcal{L}_G, G\rangle$ e $\langle \mathcal{L}_H, H\rangle$ sono granularità etichettate, l'operazione $SelectUp(G,H)$ genera una granularità etichettata $\langle \mathcal{L}_{H'}, H'\rangle$ tale che $\mathcal{L}_{H'} = \{z \in \mathcal{L}_G.\exists i \in \mathcal{L}_H.\emptyset \neq H(i) \subseteq G(z)\}$ e $H'(z) = G(z)$ per ogni $z \in \mathcal{L}_{H'}$.

Ad esempio, se *Compleanno* è la granularità che contiene, per ogni anno, il giorno in cui una persona compie gli anni, la granularità che rappresenta il mese del compleanno (o più precisamente che contiene per ogni anno il mese del compleanno) è $SelectUp(Mese, Compleanno)$.

Definizione 1.46 (Operatore *SelectByIntersect*). Se $\langle \mathcal{L}_G, G\rangle$ e $\langle \mathcal{L}_H, H\rangle$ sono granularità etichettate e k, l due numeri interi tali che $k \neq 0$ e $l > 0$, l'operazione $SelectByIntersect^l_k(G,H)$ genera una granularità etichettata $\langle \mathcal{L}_{H'}, H'\rangle$ tale che $\mathcal{L}_{H'} = \bigcup_{i\in\mathcal{L}_H} \Delta^l_k(\{z \in \mathcal{L}_G.\emptyset G(z) \cap H(i) \neq \emptyset\})$ e $H'(z) = G(z)$ per ogni $z \in \mathcal{L}_{H'}$.

Per esempio, la granularità che contiene le prime settimane di ogni mese è rappresentabile dall'espressione $SelectByIntersect^1_1(Settimana, Mese)$.

Definiamo infine tre operatori *insiemistici* (*Union*, *Intersection* e *Difference*) che corrispondono alle operazioni insiemistiche di unione, intersezione e differenza. Una granularità può essere vista infatti come insieme di granuli, ma non è detto che tutte le coppie di granularità siano compatibili con gli operatori insiemistici (per esempio la coppia *Mese*, *Settimana*) ed è quindi necessario porre la seguente restrizione: per convenzione, *gli operatori insiemistici sono definiti solamente sulle coppie di granularità etichettate G, H per le quali esiste una granularità H' tale che G e H sono sottogranularità allineate di H'.*

Definizione 1.47 (Operatore *Union*). Se $\langle \mathcal{L}_G, G\rangle$ e $\langle \mathcal{L}_H, H\rangle$ sono sottogranularità etichettate di $\langle \mathcal{L}_{H'}, H'\rangle$, l'operazione $Union(G,H)$ genera una granularità etichettata $\langle \mathcal{L}_{H''}, H''\rangle$ tale che $\mathcal{L}_{H''} = \mathcal{L}_G \cup \mathcal{L}_H$ e

$$H''(z) = \begin{cases} G(z) & \text{se } z \in \mathcal{L}_G \\ H(z) & \text{se } z \in \mathcal{L}_H \setminus \mathcal{L}_G \end{cases}$$

Definizione 1.48 (Operatore *Intersection*). Se $\langle \mathcal{L}_G, G\rangle$ e $\langle \mathcal{L}_H, H\rangle$ sono sottogranularità etichettate di $\langle \mathcal{L}_{H'}, H'\rangle$, l'operazione *Intersection*(G, H) genera una granularità etichettata $\langle \mathcal{L}_{H''}, H''\rangle$ tale che $\mathcal{L}_{H''} = \mathcal{L}_G \cap \mathcal{L}_H$ e $H''(z) = G(z)$ per ogni $z \in \mathcal{L}_{H''}$.

Definizione 1.49 (Operatore *Difference*). Se $\langle \mathcal{L}_G, G\rangle$ e $\langle \mathcal{L}_H, H\rangle$ sono sottogranularità etichettate di $\langle \mathcal{L}_{H'}, H'\rangle$, l'operazione *Difference*(G, H) genera una granularità etichettata $\langle \mathcal{L}_{H''}, H''\rangle$ tale che $\mathcal{L}_{H''} = \mathcal{L}_G \setminus \mathcal{L}_H$ e $H''(z) = G(z)$ per ogni $z \in \mathcal{L}_{H''}$.

1.5.4 Espressività

È importante ricordare che alcuni operatori della Calendar Algebra sono definiti solamente se valgono precise relazioni tra gli operandi: per esempio le operazioni insiemistiche possono essere effettuate solamente su granularità che sono sottogranularità allineate di una stessa granularità. Il problema di decidere (in generale) se due granularità qualsiasi soddisfano le condizioni imposte dalle operazioni è chiaramente indecidibile, ed è quindi necessario introdurre alcune restrizioni sugli insiemi di granularità rappresentabili. In particolare, se si assume che esista una *granularità minima* $G_\perp$ i cui granuli appartengono ad un intervallo convesso (anche infinito) di interi (per esempio $\mathbb{N}^+$) il problema della verifica delle precondizioni degli operatori della Calendar Algebra diventa decidibile.

La verifica effettiva di tali condizioni può tuttavia essere molto costosa in termini di tempo e spazio, e risultare quindi inapplicabile nella pratica. Per risolvere il problema Ning, Jajodia e Wang (cfr. [NJW02]) introducono opportune restrizioni sintattiche sulle espressioni che possono essere costruite, attraverso la definizione di tre classi di granularità e la deduzione di condizioni sufficienti per l'applicazione delle operazioni della Calendar Algebra.

Diamo ora la definizione di calendario nel formalismo della Calendar Algebra. Tale definizione è ovviamente diversa da quella data in precedenza (cfr. Definizione 1.31) poiché riguarda solamente le *rappresentazioni finite* di granularità che sono esprimibili mediante gli operatori visti nelle sezioni precedenti.

Definizione 1.50. Fissata una granularità iniziale $G_\perp$, un *calendario* è definito ricorsivamente dalle seguenti espressioni:

- $\langle \{G_\perp\}, \emptyset\rangle$ è un calendario;
- se $\langle \mathcal{G}, \mathcal{E}\rangle$ è un calendario, e è una espressione costruita applicando gli operatori della Calendar Algebra alle granularità di $\mathcal{G}$ e $G \notin \mathcal{G}$ è la granularità rappresentata da e, allora anche $\langle \mathcal{G} \cup \{G\}, \mathcal{E} \cup \{G = e\}\rangle$ è un calendario.

Mostriamo ora quali sono le granularità esprimibili attraverso i termini della Calendar Algebra, definendo prima le nozioni di granularità periodica e granularità finita, ed enunciando poi il teorema di espressività della Calendar Algebra

Definizione 1.51. Una granularità G è detta *periodica* (*rispetto alla granularità iniziale* $G_\perp$) se e solo se $G_\perp \trianglelefteq^p G$ (cioè se e solo se G raggruppa periodicamente $G_\perp$).

Definizione 1.52. Una granularità G è detta *finita* (*rispetto alla granularità iniziale* $G_\perp$) se e solo se $G_\perp \trianglelefteq G$ (cioè se e solo se G raggruppa $G_\perp$) e G ha un numero finito di granuli.

Teorema 1.53 (Espressività della Calendar Algebra). *Gli operatori della Calendar Algebra permettono di rappresentare tutte le granularità finite o periodiche (rispetto ad una granularità iniziale* $G_\perp$*).*

Le granularità periodiche o finite non sono tuttavia le sole granularità rappresentabili mediante il formalismo della Calendar Algebra: per esempio, l'operatore *AlterTick* può generare granularità che raggruppano periodicamente *con finite eccezioni* la granularità iniziale $G_\perp$. Inoltre, il Teorema 1.53 stabilisce che la Calendar Algebra è strettamente più espressiva di altri due formalismi utilizzati per rappresentare le granularità, le *Slice Expression* e le *Collection Expression*, descritti in [BDS99, LMF86, NS92]. In questa tesi mostreremo come i formalismi basati sulle stringhe e quelli basati su automi siano espressivi almeno quanto la Calendar Algebra, e come risultino più efficaci per gestire alcuni problemi che il formalismo della Calendar Algebra non affronta. In particolare il problema dell'equivalenza tra le rappresentazioni delle granularità non viene affrontato nella Calendar Algebra, mentre è risolvibile facilmente nei formalismi che presenteremo nei prossimi capitoli.

Capitolo 2

Approccio basato su stringhe

In questo capitolo presentiamo il formalismo per la rappresentazione delle granularità basato su stringhe infinite introdotto da Wijsen in [Wij00]. Abbiamo già mostrato, nel capitolo precedente, come sia possibile sfruttare le proprietà delle relazioni $\trianglelefteq^p$ e $\trianglelefteq^{fp}$ (cfr. Osservazioni 1.20 e 1.23) per costruire rappresentazioni finite di granularità temporali. Se si assume l'esistenza di una granularità minima $G_\perp$ è possibile infatti rappresentare ogni granularità che raggruppa periodicamente $G_\perp$ nei termini delle relazioni tra i granuli. In alternativa, si può supporre l'esistenza di un *unico dominio temporale discreto* (come, ad esempio, $\langle \mathbb{N}, \leq \rangle$) e rappresentare le granularità descrivendo il modo in cui i granuli raggruppano i singoli istanti temporali.

L'approccio basato su stringhe che viene proposto in questo capitolo utilizza parole fondamentalmente periodiche sull'alfabeto $\{\blacksquare, \square, \wr\}$ per rappresentare il modo in cui i granuli di $G_\perp$ si raggruppano nella granularità G. In particolare, il simbolo $\blacksquare$ indica la presenza di un granulo di $G_\perp$ coperto da G, il simbolo $\square$ indica i granuli di $G_\perp$ non coperti da G, ed il simbolo $\wr$ separa i granuli di G con indici contigui. Per esempio, la ω-parola $\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\square\square\wr\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\square\square\wr\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\square\square\wr\ldots$ rappresenta la granularità *SettimanaLavorativa* rispetto al dominio temporale dei giorni o, alternativamente, la relazione *Giorni* $\trianglelefteq$ *SettimanaLavorativa*.

Poiché il formalismo usa parole fondamentalmente periodiche, esse possono essere codificate mediante un prefisso iniziale finito ed un periodo finito che si ripete infinite volte. Queste codifiche sono dette *Granspec* e in [Wij00] Wijsen ne studia i problemi di minimizzazione ed equivalenza. Tuttavia la nozione di granularità a cui si fa riferimento è diversa da quella proposta da Bettini in [BJW00] ed utilizzata in questa tesi. In particolare, Wijsen definisce una granularità come una *partizione monotona* di un sottoinsieme dei numeri naturali, cioè come una relazione di equivalenza $G \subseteq T \times T$ dove $T \subseteq \mathbb{N}$ e per ogni coppia di classi di equivalenza $E_1, E_2 \subseteq \mathbb{N}$ si ha che per ogni $i \in E_1$ e per ogni $j \in E_2$, $i < j$, oppure per ogni $i \in E_1$ e per ogni $j \in E_2$, $j < i$. Questa definizione non fa riferimento agli indici dei granuli (cfr. Definizione 1.3) né ad eventuali etichettature sui granuli (cfr. Definizione 1.35), ed è quindi necessario adattare i risultati ottenuti da Wijsen alla definizione di granularità etichettata per permettere un confronto tra la Calendar Algebra e le Granspec.

2.1 Definizione di Granspec

Una *Granspec* è una coppia ordinata di stringhe finite sull'alfabeto $\{\blacksquare, \square, \wr\}$, in cui la prima componente rappresenta il prefisso e la seconda il periodo della parola fondamentalmente periodica rappresentata. Formalmente, la definizione di Granspec è la seguente.

Definizione 2.1. Una *Granspec* è una coppia ordinata $\alpha = \langle u, v\rangle$ di parole finite sull'alfabeto $\{\blacksquare, \square, \wr\}$ tale che $\#_{\blacksquare,\square}(v) \geq 1$. La parola u è detta *prefisso* della Granspec, mentre la parola v è detta *periodo*.

Il periodo di una Granspec contiene, per definizione, almeno un'occorrenza del simbolo $\blacksquare$ o del simbolo $\square$ e quindi ogni Granspec codifica una ω-parola fondamentalmente periodica che contiene infinite occorrenze dei simboli in $\{\blacksquare, \square\}$. La parola fondamentalmente periodica codificata da una Granspec si dice *traccia*: formalmente, se $\alpha = \langle u, v\rangle$ è una Granspec, indicheremo con $\alpha^\infty = u \cdot v^\omega$ la traccia codificata da α.

Due Granspec α, β possono codificare la stessa ω-parola: in questo caso scriveremo che $\alpha \equiv_T \beta$. Per esempio $\alpha = \langle \varepsilon, \square\blacksquare\blacksquare\wr\rangle$ e $\beta = \langle \square, \blacksquare\blacksquare \wr \square\rangle$ sono due Granspec che codificano entrambe la parola $\square\blacksquare\blacksquare \wr \square\blacksquare\blacksquare \wr \square\blacksquare\blacksquare \wr \ldots$, e quindi $\alpha \equiv_T \beta$.

In questo formalismo le parole sull'alfabeto $\{\blacksquare, \square, \wr\}$ che contengono infinite occorrenze dei simboli non separatori $\blacksquare$ e $\square$ sono utilizzate per rappresentare le granularità temporali. Abbiamo già osservato che in [Wij00] si utilizza una definizione diversa di granularità dove si indicano con questo termine tutte le partizioni monotone di un sottoinsieme dei naturali. Per poter confrontare i risultati ottenuti da Wijsen con il formalismo della Calendar Algebra, è necessario quindi specificare il modo in cui le Granspec rappresentano le granularità etichettate, secondo la definizione seguente.

Definizione 2.2. Se $\alpha = \langle u, v\rangle$ è una Granspec, e $\langle \mathcal{L}_G, G\rangle$ e $\langle \mathcal{L}_H, H\rangle$ sono due granularità etichettate tali che $\mathcal{L}_G \subseteq \mathbb{N}$, $\mathcal{L}_H \subseteq \mathbb{N}$ e $G \trianglelefteq H$, diciamo che α *rappresenta la relazione* $G \trianglelefteq H$ (oppure che α *rappresenta* H *rispetto a* G) se e solo se, per ogni $i \in \mathcal{L}_G$, $z \in \mathcal{L}_H$, $G(i) \subseteq H(z) \Leftrightarrow \alpha^\infty(i + z - 1) = \blacksquare \wedge \#_\wr(\alpha^\infty[1, i + z - 1]) = z - 1$.

Intuitivamente, la definizione stabilisce che il granulo $G(i)$ è coperto dal granulo $H(z)$ se l'i-esima occorrenza dei simboli non separatori di α^∞ coincide con $\blacksquare$, mentre non è coperto da $H(z)$ se l'i-esima occorrenza dei simboli non separatori coincide con $\square$. L'etichetta del granulo $H(z)$ è invece determinata dal numero di occorrenze di $\wr$ che precedono la i-esima occorrenza di simboli non separatori.

Come vedremo in seguito, ogni Granspec consente di rappresentare una istanza della relazione $\trianglelefteq$, mentre ogni istanza della relazione $\trianglelefteq^{fp}$ può essere rappresentata mediante una opportuna Granspec.

La definizione permette l'esistenza di Granspec che, pur codificando parole fondamentalmente periodiche diverse, rappresentano la stessa granularità. Per esempio, $\alpha = \langle \varepsilon, \blacksquare\blacksquare\blacksquare\blacksquare\square\square\wr\rangle$ e $\beta = \langle \varepsilon, \blacksquare\blacksquare\blacksquare\blacksquare \wr \square\square\rangle$ codificano le ω-parole $\blacksquare\blacksquare\blacksquare\blacksquare\square\square\wr\blacksquare\blacksquare\blacksquare\blacksquare\square\square\wr\ldots$ e $\blacksquare\blacksquare\blacksquare\blacksquare\wr\square\square\blacksquare\blacksquare\blacksquare\blacksquare\wr\square\square\ldots$ che sono distinte ma rappresentano entrambe la granularità *SettimanaLavorativa*. Si noti come, in generale, sostituendo in una stringa $\square\wr$ con $\wr\square$ o viceversa, la granularità

rappresentata rimanga invariata. Useremo la notazione $\alpha \equiv_G \beta$ per indicare che le Granspec α e β rappresentano la stessa istanza della relazione $\trianglelefteq$, o, analogamente, che rappresentano la stessa granularità G rispetto alla granularità di base $G_\perp$.

2.1.1 Espressività delle Granspec

In questa sezione mostreremo come il formalismo delle Granspec permetta di rappresentare tutte le istanze della relazione $\trianglelefteq^{fp}$ tra le granularità etichettate su interi positivi e come ogni Granspec rappresenti una istanza della relazione $\trianglelefteq$. Non è possibile tuttavia affermare che le Granspec rappresentano tutte e sole le istanze della relazione $\trianglelefteq^{fp}$: per esempio la Granspec $\langle \varepsilon, \blacksquare \rangle$ rappresenta la relazione $G \trianglelefteq H$, dove $H(1) = \bigcup_{i \in \mathbb{Z}} G(i)$ e per ogni indice $z \neq 1$, $H(z) = \emptyset$; tuttavia $G \not\trianglelefteq^{fp} H$.

Teorema 2.3 (Espressività delle Granspec). *Se $\langle \mathcal{L}_G, G \rangle$ e $\langle \mathcal{L}_H, H \rangle$ sono granularità etichettate su interi positivi tali che $G \trianglelefteq^{fp} H$, allora esiste una Granspec α che rappresenta la relazione $G \trianglelefteq^{fp} H$. Viceversa, se α è una Granspec, allora esistono due granularità etichettare su interi positivi $\langle \mathcal{L}_G, G \rangle$ e $\langle \mathcal{L}_H, H \rangle$ tali che $G \trianglelefteq H$ e α rappresenta la loro relazione.*

Dimostrazione. Siano $\langle \mathcal{L}_G, G \rangle$ e $\langle \mathcal{L}_H, H \rangle$ due granularità etichettate su interi positivi tali che $G \trianglelefteq^{fp} H$: costruiamo la Granspec $\langle u, v \rangle$ che rappresenta H rispetto a G.

Siano quindi q e r i periodi e siano $E_1, E_2, \ldots, E_s$ le eccezioni dell'istanza della relazione $\trianglelefteq^{fp}$ considerata (cfr. Definizione 1.21). Sia inoltre z l'etichetta massima contenuta nelle eccezioni $E_1, E_2, \ldots, E_s$, cioè sia $z = Max(\bigcup_{x \in [1,s]} E_x)$, e sia i l'ultima etichetta di G tale che $G(i) \subseteq H(z)$. Dalla definizione della relazione $\trianglelefteq^{fp}$ segue che, per ogni indice $z' > z$, esiste una sequenza di indici $i_1, i_2, \ldots i_n$ che rende vera l'asserzione $\forall h \in \mathbb{N}. H(z' + h \cdot r) \neq \emptyset \Rightarrow H(z' + h \cdot r) = \bigcup_{k \in [1,n]} G(i_k + h \cdot q)$. Definiamo quindi la sequenza infinita di parole (eventualmente vuote o infinite) $w_1, w_2, w_3, \ldots$ tale che:

$$w_{z'}(i') = \begin{cases} \blacksquare & \text{se } G(i') \subseteq H(z'), \\ \square & \text{altrimenti,} \end{cases} \qquad \text{per ogni } z' \in \mathbb{N}^+ \text{ e } i' \in [1, b_{z'+1} - b_{z'}],$$

dove $b_{z'}$ è il massimo valore contenuto nell'insieme $\{j \in \mathcal{L}_G | G(j) \subseteq H(z'-1)\}$, oppure $b_{z'-1}$ se tale insieme è vuoto.

Poiché H raggruppa periodicamente e con finite eccezioni G, la parola $w = w_1 \wr w_2 \wr w_3 \wr \ldots$ è fondamentalmente periodica ed esistono quindi un prefisso u ed un periodo v tali che $w = u \cdot v^\omega$. Combinando le parole finite u e v (sostituendo v con la parola $\square$ nel caso in cui sia composta solamente da separatori) otteniamo la Granspec $\langle u, v \rangle$ che rappresenta la relazione $G \trianglelefteq^{fp} H$.

Fissiamo ora una Granspec $\alpha = \langle u, v \rangle$ ed una generica granularità G. Sia $S_z = \{i \in \mathbb{N} | \alpha^\infty(i + z - 1) = \blacksquare \wedge \#_\wr(\alpha^\infty[1, i + z - 1]) = z - 1\}$ per ogni $z \in \mathbb{Z}$, e poniamo quindi $H(z) = \bigcup_{i \in S_z} G(i)$. Si dimostra facilmente che H è una granularità che raggruppa G e che la Granspec α rappresenta tale relazione. $\square$

2.2 Equivalenza tra Granspec

Abbiamo già osservato che, se α e β sono due Granspec, allora $\alpha = \beta$ implica $\alpha \equiv_T \beta$, e $\alpha \equiv_T \beta$ implica $\alpha \equiv_G \beta$; le implicazioni inverse non sono tuttavia valide, come dimostrato dagli esempi forniti all'inizio del capitolo. Nel formalismo delle Granspec si pone quindi il problema di stabilire se due Granspec qualsiasi rappresentano la stessa granularità. Questo problema è detto *problema dell'equivalenza tra Granspec* e viene risolto mediante la definizione delle forme *allineate* e *canoniche* per le Granspec, che permetteranno di dimostrare le seguenti implicazioni:

- se α e β sono Granspec in forma allineata, allora $\alpha \equiv_G \beta \Rightarrow \alpha \equiv_T \beta$;
- se α e β sono Granspec in forma canonica, allora $\alpha \equiv_G \beta \Rightarrow \alpha = \beta$.

Inoltre le forme allineate e canoniche delle Granspec sono effettivamente computabili mediante opportuni algoritmi, consentendo così di risolvere il problema dell'equivalenza. Si dimostra infatti che valgono le seguenti proprietà:

- α e β rappresentano la stessa granularità se e solo se le rispettive forme allineate codificano la stessa ω-parola;
- α e β codificano la stessa ω-parola se e solo se le rispettive forme canoniche coincidono.

2.2.1 Forma allineata

In questa sezione e nella successiva definiremo le forme allineate e le forme canoniche per le Granspec, adattando quanto riportato in [Wij00] alla nozione di granularità etichettata. In particolare, ammetteremo la possibilità di avere occorrenze consecutive del simbolo $\wr$ nelle parole codificate da Granspec in forma allineata, in modo da poter rappresentare valori non contigui nelle etichettature delle granularità.

Definizione 2.4. Una Granspec α è in *forma allineata* se e solo se per ogni intero positivo i valgono le seguenti condizioni:

- $\alpha^\infty(i) = \wr$ implica che $i = 1$ oppure $\alpha^\infty(i-1) \neq \square$;
- $\alpha^\infty(i) = \wr$ implica che $\exists j > i.\alpha^\infty(j) = \blacksquare$.

La definizione stabilisce, nella prima condizione, che ogni occorrenza del simbolo separatore $\wr$ è immediatamente preceduta da un'occorrenza di $\blacksquare$ o da un'altra occorrenza di $\wr$. La seconda condizione stabilisce invece che a ogni occorrenza di $\wr$ segue, prima o poi, un'occorrenza di $\blacksquare$. Per esempio, la Granspec $\langle \wr\square, \blacksquare \wr \wr \rangle$ è allineata, mentre $\langle \square\wr, \blacksquare \wr \wr \rangle$ non lo è perché la prima occorrenza del simbolo $\wr$ è preceduta dal simbolo $\square$.

Le forme allineate possono essere ottenute a partire da generiche Granspec sfruttando le seguenti equivalenze:

1. $\langle u'\square \wr u'', v \rangle \equiv_G \langle u' \wr \square u'', v \rangle$;
2. $\langle u, v'\square \wr v'' \rangle \equiv_G \langle u, v' \wr \square v'' \rangle$;
3. $\langle u, \wr v'\square \rangle \equiv_G \langle u\wr, v' \wr \square \rangle$;

4. $\langle u'\square, \wr v' \rangle \equiv_G \langle u'\square \wr, v' \wr \rangle$;
5. $\langle u' \wr \square^n, \square^m \rangle \equiv_G \langle u'\square^n, \square^m \rangle$, per ogni $n, m \in \mathbb{N}^+$.

Per ottenere una forma allineata a partire da una generica Granspec è sufficiente sostituire le rappresentazioni non allineate con le rappresentazioni poste a destra del simbolo $\equiv_G$: la seguente proposizione dimostra l'esistenza di una forma allineata per ogni Granspec affermando che il processo di sostituzione descritto termina producendo una rappresentazione allineata.

Proposizione 2.5. *Per ogni Granspec α esiste una forma allineata β tale che $\alpha \equiv_G \beta$.*

Dimostrazione. Dimostriamo innanzi tutto che ogni Granspec non allineata è nella forma di almeno una delle parti sinistre delle equivalenze 1-5 Se $\alpha = \langle u, v \rangle$ è una Granspec non allineata allora vale uno dei due casi seguenti:

- un'occorrenza di $\wr$ in $\alpha^\infty = u \cdot v^\omega$ è immediatamente preceduta dal simbolo $\square$: in tal caso $u = u'\square \wr u''$, oppure $v = v'\square \wr v''$, oppure $v = \wr v'\square$, oppure $u = u'\square$ e $v = \wr v'$; quindi è possibile applicare l'equivalenza 1, 2, 3, o 4;
- un'occorrenza di $\wr$ in $\alpha^\infty = u \cdot v^\omega$ è immediatamente preceduta dal simbolo $\blacksquare$ o dal simbolo $\wr$, ma non è seguita da alcuna occorrenza di $\blacksquare$: in tal caso il periodo di α è costituito unicamente da ripetizioni del simbolo $\square$ ed è pertanto possibile applicare l'equivalenza 5.

Dimostriamo ora che la procedura di sostituzione delle Granspec non allineate con i membri destri delle equivalenze 1-5 termina in un numero finito di passi. Sia $\alpha = \langle u, v \rangle$ una generica Granspec e sia n la somma delle lunghezze del prefisso e del periodo di α più uno, cioè $n = |u| + |v| + 1$. Definiamo la funzione $\mathcal{F}_n : \{\blacksquare, \square, \wr\}^\omega \to \mathbb{N}$ che associa ad ogni ω-parola w il valore $\sum_{i \in [1,n]} i \cdot \#_\wr(w(i))$ (somma delle posizioni del simbolo $\wr$ in $w[1, n]$). La funzione $\mathcal{F}_n$ è limitata inferiormente da 0 e superiormente da n^2, ed inoltre si verifica che ognuna delle cinque equivalenze $\alpha \equiv_G \beta$ implica $\mathcal{F}_n(\alpha^\infty) > \mathcal{F}_n(\beta^\infty)$. Di conseguenza si può affermare che continuando ad applicare le equivalenze 1-5 si ottiene, dopo un numero finito di passi, una rappresentazione allineata. □

Proposizione 2.6. *Se α e β sono due Granspec in forma allineata, allora $\alpha \equiv_G \beta \Rightarrow \alpha \equiv_T \beta$.*

Dimostrazione. Per dimostrare la proposizione supponiamo che α e β siano due Granspec allineate tali che $\alpha \not\equiv_T \beta$ e proviamo che $\alpha \not\equiv_G \beta$, cioè che per ogni coppia di istanze (G, H) e (G, H') della relazione $\trianglelefteq$, α rappresenta H rispetto ad G e β rappresenta H' rispetto a G solo se $H \neq H'$.

Sia G una generica granularità che si raggruppa nelle granularità etichettate $\langle \mathcal{L}_H, H \rangle$ e $\langle \mathcal{L}_{H'}, H' \rangle$, rappresentate rispettivamente dalle Granspec allineate α e β; siano inoltre, n il minimo intero positivo tale che $\alpha^\infty(n) \neq \beta^\infty(n)$ (quindi, per ogni $j < n$, $\alpha^\infty(j) = \beta^\infty(j)$) e i e z gli interi tali che:

$$\begin{cases} i = \#_{\blacksquare,\square}(\alpha^\infty[1, n[) = \#_{\blacksquare,\square}(\beta^\infty[1, n[), \\ z = \#_\wr(\alpha^\infty[1, n[) + 1 = \#_\wr(\beta^\infty[1, n[) + 1. \end{cases}$$

Dimostriamo ora che $\langle \mathcal{L}_H, H \rangle \neq \langle \mathcal{L}_{H'}, H' \rangle$ in funzione dei simboli che precedono l'n-esima posizione in α e β, distinguendo i seguenti casi.

- Se $\alpha^\infty(n) = \blacksquare$ e $\beta^\infty(n) = \wr$ (o viceversa), allora $G(i+1) \subseteq H(z)$, ma il granulo $H'(z)$, se esiste, non include $G(i+1)$. Quindi si può concludere che $\langle \mathcal{L}_H, H\rangle \neq \langle \mathcal{L}_{H'}, H'\rangle$.
- Se $\alpha^\infty(n) = \square$, $\beta^\infty(n) = \wr$ e $\exists n' > n.\alpha^\infty(n') = \blacksquare$ (o viceversa), allora esiste un granulo di G con indice $i' > i$ incluso in $H(z)$ ma non in $H'(z)$. Anche in questo caso $\langle \mathcal{L}_H, H\rangle \neq \langle \mathcal{L}_{H'}, H'\rangle$.
- Se $\forall n' \geq n.\alpha^\infty(n') = \square$ e $\beta^\infty(n) = \wr$ (o viceversa), allora esiste $n' > n$ tale che $\beta^\infty(n') = \blacksquare$ (altrimenti β non sarebbe in forma allineata) e quindi esiste un granulo di H' etichettato con $z' > z$. Ma le etichette di H non contengono alcun intero maggiore di z, e di conseguenza $\langle \mathcal{L}_H, H\rangle \neq \langle \mathcal{L}_{H'}, H'\rangle$.
- Se $\alpha^\infty(n) = \blacksquare$ e $\beta^\infty(n) = \square$ (o viceversa), allora $G(i+1) \subseteq H(z)$ e $G(i+1) \nsubseteq H'(z)$. Si ricava dunque che vale la disuguaglianza $\langle \mathcal{L}_H, H\rangle \neq \langle \mathcal{L}_{H'}, H'\rangle$.

□

Osservazione 2.7. Per i risultati appena dimostrati due Granspec α e β rappresentano la stessa granularità se e solo se le rispettive forme allineate codificano la stessa ω-parola.

Presentiamo ora brevemente l'Algoritmo 2.1, che computa la forma allineata di una Granspec $\alpha = \langle u, v\rangle$ in tempo polinomiale rispetto alla dimensione dell'input (più precisamente con complessità $O(|uv|^3)$). Esso simula i passi di sostituzione della dimostrazione della Proposizione 2.5 e termina dopo al più $|uv|^2$ sostituzioni, ognuna delle quali richiede tempo $O(|uv|)$. Per la semplicità dell'algoritmo, non presenteremo alcuna dimostrazione di correttezza.

Algoritmo 2.1 Calcolo della forma allineata della Granspec α

```
1: ⟨u, v⟩ ← α
2: while true do
3:    if ∃u', u''.u = u'□ ≀ u'' then
4:       u ← u' ≀ □u''
5:    else if ∃v', v''.v = v'□ ≀ v'' then
6:       v ← v' ≀ □v''
7:    else if ∃v'.v = ≀v'□ then
8:       u ← u≀
9:       v ← v' ≀ □
10:   else if ∃u'.u = u'□ e ∃v'.v = ≀v' then
11:      u ← u' ≀ □
12:      v ← v'≀
13:   else if ∃u', n.u = u' ≀ □^n e ∃m.v = □^m then
14:      u ← u'□^n
15:      v ← □^m
16:   else
17:      return ⟨u, v⟩
18:   end if
19: end while
```

2.2.2 Forma canonica

Abbiamo già osservato che le Granspec codificano parole fondamentalmente periodiche, rappresentandole mediante un prefisso ed un periodo finiti. Prima di introdurre il concetto di forma canonica per le Granspec, presentiamo la definizione generale di *codifica* di una parola (finita o infinita).

Definizione 2.8. Una coppia (u, v) di parole finite sull'alfabeto A è una *codifica di una parola (finita o infinita)* $w \in A^\omega \cup A^*$ se e solo se $w = u \cdot v^\omega$.

Le parole u e v di una codifica (u, v) sono indicate rispettivamente con i termini *prefisso* e *periodo*.

Si noti che la Definizione 2.8 riguarda sia le parole finite che le parole fondamentalmente periodiche: ad esempio, (abc, ε) e (ε, ab) codificano rispettivamente la parola finita abc e la parola fondamentalmente periodica $abababababab\ldots$. In questo capitolo considereremo solamente le codifiche di parole fondamentalmente periodiche; nei capitoli successivi verranno invece considerate anche le codifiche di parole finite.

Due codifiche di parole (u, v) e (u', v') si dicono *equivalenti* se e solo se $u \cdot v^\omega = u' \cdot v'^\omega$. Useremo la notazione $(u, v) \equiv_W (u', v')$ per indicare che le due codifiche (u, v) e (u', v') sono equivalenti.

Definizione 2.9. Una codifica (u, v) della parola fondamentalmente periodica $w = u \cdot v^\omega$ è detta *minima* se e solo se

- il periodo v è primitivo, cioè $v = w^k$ implica $k = 1$ e $w = v$;
- il prefisso u è tale che $u \neq \varepsilon$ implica $u(|u|) \neq v(|v|)$.

Per esempio, (ε, abc) è una codifica minima per la parola fondamentalmente periodica $w = abcabcabc\ldots$, mentre (a, bca) è sempre una codifica di w, ma non è minima.

Per determinare una codifica minima per una parola fondamentalmente periodica si sfruttano le seguenti equivalenze:

1. $(u, v^k) \equiv_W (u, v)$ per ogni $k > 1$;
2. $(ux, vx) \equiv_W (u, xv)$ per ogni $x \neq \varepsilon$.

Riportiamo ora alcune proprietà delle parole fondamentalmente periodiche che saranno utili per stabilire l'esistenza e l'unicità della loro codifica minima, e che verranno utilizzate sia in questo capitolo (per definire la forma canonica delle Granspec) che nei capitoli successivi, in particolare nei formalismi per rappresentare insiemi di granularità basati su automi.

Lemma 2.10 (Lemma di Fine-Wilf [FW65]). *Se una parola v ha periodi di lunghezza q e q' e lunghezza $|v| \geq q + q' - M.C.D.(q, q')$, allora v possiede un periodo di lunghezza pari a $M.C.D.(q, q')$.*

Dimostrazione. Dimostriamo il lemma per induzione sulla somma di q e q'. Se $q = q' = 1$ allora $M.C.D.(q, q') = 1$ e la proprietà affermata dal lemma è banalmente verificata.

Se invece $q + q' > 2$ assumiamo $q < q'$ e fissiamo il prefisso v' di v di lunghezza $q + q' - M.C.D.(q, q')$. Per ogni $i \in [1, q - M.C.D.(q, q')]$ possiamo scrivere $v'(i) = v(i) = v(i + q') = v(i + q' - q) = v'(i + q' - q)$

e quindi v' ha sia un periodo di lunghezza q che un periodo di lunghezza $q' - q$. Poiché $M.C.D.(q', q) = M.C.D.(q, q' - q)$, per l'ipotesi induttiva v' possiede un periodo di lunghezza $M.C.D.(q, q')$. Vale inoltre la disuguaglianza $q' \leq |v'|$ e quindi il prefisso di v di lunghezza q' ha un periodo di lunghezza $M.C.D.(q, q')$. Osservando che v possiede un periodo di lunghezza q' multipla di $M.C.D.(q, q')$ si ricava immediatamente che v possiede anche un periodo di lunghezza $M.C.D.(q, q')$. □

Definizione 2.11. Se u e v sono parole finite, diciamo che v *occorre in* u *con offset* i se e solo se $v = u[i + 1, i + |v|]$.

Lemma 2.12. *Se $v = w^k$, allora v occorre in $v \cdot v$ con offset $|w|$. Viceversa, se v occorre in $v \cdot v$ con offset i che divide $|v|$, allora v è ripetizione di $v[1, i]$.*

Dimostrazione. La prima implicazione si dimostra immediatamente osservando che w^k occorre in $v \cdot v = w \cdot w^k \cdot w^{k-1}$ con offset $|w|$. Viceversa, se v occorre in $v \cdot v$ con offset i, allora, per ogni $j \in [1, |v|]$, $v(j) = (v \cdot v)(j + i)$. Inoltre, se i divide $|v|$, $v[1, i]$ è un periodo di v, e ciò conclude la dimostrazione del lemma. □

Proposizione 2.13. *Per ogni parola fondamentalmente periodica w esiste una codifica minima (u, v) tale che $w = u \cdot v^\omega$.*

Dimostrazione. Sia w una parola fondamentalmente periodica e sia (u, v) una sua codifica. Se (u, v) non soddisfa i requisiti di codifica minima, allora vale una tra le seguenti condizioni:

- $v = x^k$, con $k > 1$;
- $u \neq \varepsilon$ e $u(|u|) = v(|v|)$.

È possibile quindi applicare una delle equivalenze 1 o 2, ottenendo nel primo caso la codifica (u, x), e nel secondo la codifica $(u[1, |u| - 1], v(|v|) \cdot v[1, |v| - 1])$. In entrambi i casi la rappresentazione ottenuta è equivalente a (u, v) ed ha dimensione inferiore (intendendo per dimensione di una codifica la somma delle lunghezze del prefisso e del periodo). Iterando questa procedura si ottiene, dopo un numero finito di passi, una codifica minima per w. □

Proposizione 2.14. *Se (u, v) e (u', v') sono due codifiche minime di parole fondamentalmente periodiche, allora $(u, v) \equiv_W (u', v') \Rightarrow (u, v) = (u', v')$.*

Dimostrazione. Siano (u, v) e (u', v') due codifiche minime di parole fondamentalmente periodiche tali che $u \cdot v^\omega = u' \cdot v'^\omega$.

Dimostriamo quindi che $u = u'$ e $v = v'$. Se per assurdo fosse $|u| < |u'|$ allora u sarebbe prefisso di u' ed esisterebbe $x \neq \varepsilon$ tale che $u' = ux$. Dall'uguaglianza $u \cdot v^\omega = u' \cdot v'^\omega$ si ricava l'esistenza di una parola y tale che $v = xy$ e $(yx)^\omega = v'^\omega$. Di conseguenza, vale $(yx)^{|v'|} = v'^{|v|}$ e u' e v' terminano con lo stesso simbolo, contro la definizione di codifica minima. In modo analogo si dimostra che la disuguaglianza $|u| > |u'|$ viola le ipotesi.

Quindi si ha che $|u| = |u'|$, dunque $u = u'$ e $v^{|v'|} = v'^{|v|}$. Per il Lemma 2.10 la parola $v^{|v'|} = v'^{|v|}$ ha un periodo di lunghezza $M.C.D.(|v|, |v'|)$; quindi, se fosse $|v| \neq |v'|$ uno dei due periodi non sarebbe primitivo e violerebbe la definizione di codifica minima. Si conclude quindi che anche $v = v'$ e $(u, v) = (u', v')$. □

Nella dimostrazione della proposizione 2.13 si mostra come sia possibile costruire una codifica minima per la parola fondamentalmente periodica w a partire da una codifica qualsiasi applicando le equivalenze 1 e 2. Si noti che dopo ogni passo la codifica ha dimensione strettamente inferiore al passo precedente: questo (assieme al risultato di unicità della Proposizione 2.14) consente di enunciare il seguente corollario, che spiega perché si sia usato il termine "codifica minima" nella Definizione 2.9.

Corollario 2.15. *Se w è una parola fondamentalmente periodica e (u, v) è la sua codifica minima, allora, per ogni altra codifica (u', v') di w, vale la disuguaglianza $|u| + |v| \leq |u'| + |v'|$.*

La nozione di forma canonica per le Granspec è strettamente correlata con il concetto codifica minima delle parole fondamentalmente periodiche. La sua definizione è infatti la seguente.

Definizione 2.16. Una Granspec $\alpha = \langle u, v \rangle$ è in *forma canonica* se e solo se

- α è allineata;
- (u, v) è la codifica minima di α^∞.

Per esempio, la forma canonica di $\langle \blacksquare\blacksquare, \wr\blacksquare\blacksquare \wr \blacksquare\blacksquare \rangle$ è $\langle \varepsilon, \blacksquare\blacksquare\wr \rangle$. Le proprietà di esistenza e di unicità delle codifiche minime per le parole fondamentalmente periodiche permettono inoltre di dimostrare le seguenti proposizioni.

Proposizione 2.17. *Per ogni Granspec α esiste una forma canonica β tale che $\alpha \equiv_G \beta$.*

Dimostrazione. Per la Proposizione 2.5 esiste una forma allineata $\alpha' = \langle u, v \rangle$ di α tale che $\alpha \equiv_G \alpha'$. Se α' non soddisfa i requisiti della forma canonica, la Proposizione 2.13 afferma che esiste una codifica minima (u', v') equivalente a (u, v). La Granspec $\beta = \langle u', v' \rangle$ è quindi in forma canonica e codifica la stessa ω-parola di α', perciò $\beta \equiv_G \alpha' \equiv_G \alpha$. □

Proposizione 2.18. *Se α e β sono due Granspec in forma canonica, allora $\alpha \equiv_G \beta \Rightarrow \alpha = \beta$.*

Dimostrazione. Siano $\alpha = \langle u, v \rangle$ e $\beta = \langle u', v' \rangle$ due Granspec in forma canonica. Per la Proposizione 2.6, $\alpha \equiv_G \beta$ implica $\alpha \equiv_T \beta$ e dunque $u \cdot v^\omega = u' \cdot v'^\omega$.

Quindi (u, v) e (u', v') sono codifiche minime della stessa ω-parola e, per la Proposizione 2.14, $(u, v) = (u', v')$: di conseguenza anche $\alpha = \beta$. □

Osservazione 2.19. I risultati appena dimostrati permettono di affermare che due Granspec α e β codificano la stessa ω-parola se e solo se le rispettive forme canoniche coincidono.

L'Algoritmo 2.2, presentato di seguito, computa la forma canonica di una Granspec allineata α secondo la procedura mostrata nella dimostrazione della Proposizione 2.13. Esso applica, fino a quando è possibile, le sostituzioni secondo l'equivalenza 1. Terminate tali sostituzioni, se $\langle u, v \rangle$ non è in forma canonica è sufficiente applicare una sola volta la sostituzione corrispondente all'equivalenza 2, scegliendo l'intero k e la parola w di lunghezza minima. Infatti, se w è la parola di lunghezza minima tale che $v = w^k$, allora w è primitiva e $\langle u, w \rangle$ è una Granspec in forma canonica equivalente a $\langle u, v \rangle$.

Per il Lemma 2.12 v è ripetizione della parola primitiva w se e solo se v occorre in $v \cdot v$ con offset $|w| > 0$ *minimo*. Per esempio "abcabc" occorre in "abcabcabcabc" con offset 0, 3, 6, 9; ed essendo 3 il minimo offset diverso da zero si conclude che "abc" è la minima parola (e dunque primitiva) che si ripete in "abcabc".

La ricerca della sottostringa w primitiva che si ripete in v è quindi riconducibile ai problemi di string-matching classici, ed esistono algoritmi che determinano la prima occorrenza non banale di una parola v in $v \cdot v$ in tempo lineare rispetto alla dimensione dell'input, come l'algoritmo di Knuth-Morris-Pratt (cfr. [KMP77]). L'operazione di riduzione in forma canonica di una Granspec allineata $\alpha = \langle u, v \rangle$ è quindi effettuata dall'Algoritmo 2.2 in tempo $O(|uv|)$.

Algoritmo 2.2 Calcolo della forma canonica della Granspec α

1: $\langle u, v \rangle \leftarrow \alpha$
2: **while** $u \neq \varepsilon$ e $u(|u|) = v(|v|)$ **do**
3: $\quad u \leftarrow u[1, |u| - 1]$
4: $\quad v \leftarrow v(|v|) \cdot v[1, |v| - 1]$
5: **end while**
6: $i \leftarrow$ offset della prima occorrenza non banale di v in $v \cdot v$.
7: $v \leftarrow v[1, i]$
8: **return** $\langle u, v \rangle$

Riportiamo infine l'Algoritmo 2.3 che verifica l'equivalenza tra due Granspec qualsiasi α e β sfruttando le proprietà delle forme canoniche.

Algoritmo 2.3 Verifica dell'equivalenza tra le Granspec α e β

1: $\alpha' \leftarrow$ forma allineata di α {Algoritmo 2.1}
2: $\alpha'' \leftarrow$ forma canonica di α' {Algoritmo 2.2}
3: $\beta' \leftarrow$ forma allineata di β
4: $\beta'' \leftarrow$ forma canonica di β'
5: **return** $\alpha'' = \beta''$

Capitolo 3

Approcci basati su automi

Il formalismo delle Granspec, proposto da Wijsen e illustrato nel capitolo precedente, fa uso di parole fondamentalmente periodiche per rappresentare le granularità temporali. Risulta quindi naturale collegare questo approccio con la teoria degli automi a stati finiti, i cui risultati (come la chiusura rispetto alla concatenazione o rispetto alle operazioni insiemistiche, oppure l'esistenza e l'unicità dell'automa minimo) permettono di risolvere alcuni problemi salienti di rappresentazione e manipolazione delle strutture temporali in modo semplice ed efficace. L'esistenza di questo collegamento è riconosciuta, anche se non approfondita sistematicamente, da Wijsen in [Wij00]. Dal Lago e Montanari, in [DM01], propongono un primo approccio alla rappresentazione delle granularità mediante automi a stati finiti, e ne studiano l'espressività e il problema dell'equivalenza. In [DMP03a], Dal Lago, Montanari e Puppis estendono questo formalismo, definendo una classe di automi che riconosce sia parole finite che parole fondamentalmente periodiche, e consente di ottenere (mediante opportuni algoritmi) rappresentazioni basate su automi equivalenti ai termini della Calendar Algebra.

In questo capitolo studieremo in modo approfondito i formalismi basati su automi proposti in [DM01] e [DMP03a], in particolare rispetto alla loro espressività e rispetto al problema dell'equivalenza tra le rappresentazioni.

3.1 Automi single-string

Gli automi single-string sono il più semplice ed immediato collegamento tra l'approccio basato su stringhe per rappresentare le (relazioni tra) granularità e la teoria degli automi. La loro definizione è infatti la seguente.

Definizione 3.1. Un *automa single-string* è una tupla $\mathcal{A} = (S, \Sigma, \delta, s_I)$, dove:

- S è un insieme finito di stati;
- Σ è un insieme finito di simboli;
- $\delta : S \to \Sigma \times S$ è la *funzione di transizione*, che codifica le transizioni dell'automa;
- s_I è lo *stato iniziale*.

La definizione data è simile a quella di automa deterministico di Büchi. Le principali differenze sono l'assenza dell'insieme degli stati finali e il fatto che

la nozione di transizione per gli automi single-string prevede una sola istanza (s, a, s') per ogni stato $s \in S$.

Da questo momento in poi, denotiamo con π_i la *funzione di proiezione* che ritorna l'i-esima componente di una tupla qualsiasi. Se ρ è una generica sequenza di tuple (finita o infinita), denotiamo con $\pi_i(\rho)$ la sequenza $\pi_i(\rho(1))\pi_i(\rho(2))$ $\pi_i(\rho(3))\ldots$.

Se $\mathcal{A} = (S, \Sigma, \delta, s_I)$ è un automa single-string, diremo che uno stato r *è raggiungibile dallo stato* s se e solo se esistono un intero $n \in \mathbb{N}$, una sequenza finita di simboli $a_1, a_2, \ldots, a_n$ e una sequenza finita di stati $s_0, s_1, \ldots, s_n$ tali che $s_0 = s$, $s_n = r$ e $\forall i \in [0, n[.\delta(s_i) = \langle a_{i+1}, s_{i+1}\rangle$. Inoltre, per ogni automa single-string $\mathcal{A} = (S, \Sigma, \delta, s_I)$ esiste sempre uno stato s_{loop} raggiungibile da s_I e da se stesso; chiameremo tali stati *loop-state* di $\mathcal{A}$.

3.1.1 Espressività degli automi single-string

Definizione 3.2. Se $\mathcal{A} = (S, \Sigma, \delta, s_I)$ è un automa single-string, definiamo *run di* $\mathcal{A}$ la sequenza infinita $\rho \in (S \times \Sigma)^\omega$ tale che:

- $\pi_1(\rho(1)) = s_I$;
- $\delta(\pi_1(\rho(n))) = \langle \pi_2(\rho(n)), \pi_1(\rho(n+1))\rangle$, per ogni $n \in \mathbb{N}^+$.

Diremo che $\mathcal{A}$ *riconosce la* ω*-parola* w se e solo se il run $\rho \in (S \times \Sigma)^\omega$ di $\mathcal{A}$ è tale che $w = \pi_2(\rho)$.

Gli automi single-string, analogamente ad altri tipi di automi, possono essere rappresentati da grafi orientati. In questa tesi utilizzeremo le seguenti convenzioni per rappresentare gli automi:

- ciascuno stato è rappresentato da un arco etichettato con un nome (ad esempio, $s_I, s_1, s_2, \ldots$);
- lo stato iniziale è indicato da una freccia;
- le transizioni sono rappresentate da archi orientati etichettati con un simbolo.

Nel caso degli automi single-string, i loop-state sono rappresentati dai nodi appartenenti al ciclo raggiungibile a partire dallo stato iniziale.

Esempio 3.3. La Figura 3.1 rappresenta un automa single-string che riconosce la parola fondamentalmente periodica □□□□□■■ ≀ □□□□□■■ ≀ ..., che rappresenta la relazione tra granularità *Giorni* $\trianglelefteq$ *FineSettimana*. In questo caso tutti gli stati dell'automa sono loop-state.

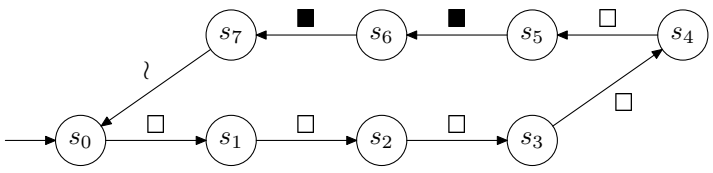

Figura 3.1: Esempio di automa single-string

Poiché le transizioni di un generico automa single-string $\mathcal{A}$ sono funzioni totali che mappano ciascuno stato di $\mathcal{A}$ in una coppia simbolo-stato, si può affermare che il run dell'automa esiste ed è unico. Quindi gli automi single-string riconoscono singole parole, e non linguaggi come gli automi di Büchi.

La seguente proposizione afferma che gli automi single-string sono propriamente meno espressivi degli automi deterministici di Büchi, e che ogni automa single-string riconosce esattamente una parola fondamentalmente periodica.

Proposizione 3.4 (Espressività degli automi single-string). *Un automa single-string $\mathcal{A} = (S, \Sigma, \delta, s_I)$ riconosce la parola w se e solo se w è fondamentalmente periodica. Inoltre, esiste un automa deterministico di Büchi $\mathcal{A}'$ che riconosce il linguaggio $\{w\}$.*

Osservazione 3.5. La Proposizione 3.4 afferma che gli automi single-string riconoscono solamente parole fondamentalmente periodiche, quindi che essi sono codifiche equivalenti alle Granspec. Gli automi single-string permettono dunque di rappresentare tutte le istanze della relazione $\trianglelefteq^{fp}$ tra granularità.

3.1.2 Equivalenza tra automi single-string

Due automi si dicono equivalenti se riconoscono lo stesso linguaggio (o parola, nel caso degli automi single-string). Il problema di riconoscere se due automi sono equivalenti può essere risolto verificando se esiste un *isomorfismo tra automi minimi equivalenti*: la costruzione dell'automa minimo equivalente ad un automa dato può essere ricondotta ai problemi di partizionamento stabile (si vedano al riguardo [HU80] e [PTB85]).

In questa sezione presenteremo un algoritmo che verifica in tempo lineare se due automi single-string sono equivalenti ma che adotta un approccio diverso, che non implica la costruzione dell'automa minimo. Ricordando i risultati di minimizzazione ed equivalenza delle codifiche di parole fondamentalmente periodiche, presentati nella discussione della forma canonica delle Granspec, si può infatti ridurre il problema dell'equivalenza tra due automi al problema dell'equivalenza tra le codifiche delle parole fondamentalmente periodiche riconosciute.

Si può notare infatti (ricordando la definizione di loop-state) che il run di un automa single-string ha la seguente struttura:

$$s_I \xrightarrow{u} s_{loop} \xrightarrow{v} s_{loop} \xrightarrow{v} \dots$$

dove s_I è lo stato iniziale, s_{loop} è un generico loop-state dell'automa e u e v sono due opportune parole finite sull'alfabeto Σ. Quindi l'automa riconosce la parola fondamentalmente periodica $u \cdot v^\omega$ e la codifica (u, v) può essere determinata semplicemente combinando i simboli riconosciuti visitando gli stati da s_I a s_{loop} e da s_{loop} a s_{loop}.

L'Algoritmo 3.1 determina una codifica (u, v) della parola riconosciuta dall'automa single string $\mathcal{A} = (S, \Sigma, \delta, s_I)$ determinando un loop-state di $\mathcal{A}$ e successivamente computando il prefisso u tale che $s_I \xrightarrow{u} s_{loop}$ ed il periodo v tale che $s_{loop} \xrightarrow{v} s_{loop}$. La complessità dell'algoritmo è lineare rispetto al numero di stati.

Per testare l'equivalenza di due automi single-string $\mathcal{A}$ e $\mathcal{A}'$ è sufficiente determinare le codifiche (u, v) e (u', v') delle parole riconosciute rispettivamente da $\mathcal{A}$ e $\mathcal{A}'$, e stabilire quindi se $(u, v) \equiv_W (u', v')$.

Algoritmo 3.1 Calcolo di una codifica della parola riconosciuta da un automa single-string

1: **let** $\mathcal{A} = (S, \Sigma, \delta, s_I)$
2: **for all** $s \in S$ **do**
3: $visited[s] \leftarrow$ false
4: **end for**
5: $s \leftarrow s_I$
6: **while** $\neg visited[s]$ **do**
7: $visited[s] \leftarrow$ true
8: $s \leftarrow \pi_2(\delta(s))$
9: **end while**
10: $s_{loop} \leftarrow s$
11: $s \leftarrow s_I$
12: $u \leftarrow \pi_1(\delta(s))$
13: **while** $s \neq s_{loop}$ **do**
14: $s \leftarrow \pi_2(\delta(s))$
15: $u \leftarrow u \cdot \pi_1(\delta(s))$
16: **end while**
17: $v \leftarrow \pi_1(\delta(s))$
18: **while** $s \neq s_{loop}$ **do**
19: $s \leftarrow \pi_2(\delta(s))$
20: $v \leftarrow v \cdot \pi_1(\delta(s))$
21: **end while**
22: **return** (u, v)

L'Algoritmo 3.2 calcola la codifica minima di (u, v), utilizzando la stessa strategia dell'Algoritmo 2.2 per il calcolo della forma canonica di una Granspec.

Algoritmo 3.2 Calcolo della codifica minima di (u, v)

1: **while** $u \neq \varepsilon$ e $u(|u|) = v(|v|)$ **do**
2: $u \leftarrow u[1, |u| - 1]$
3: $v \leftarrow v(|v|) \cdot v[1, |v| - 1]$
4: **end while**
5: $i \leftarrow$ offset della prima occorrenza non banale di v in $v \cdot v$.
6: $v \leftarrow v[1, i]$
7: **return** (u, v)

Presentiamo infine l'Algoritmo 3.3 che stabilisce l'equivalenza tra due generici automi single-string $\mathcal{A}$ e $\mathcal{A}'$.

Algoritmo 3.3 Verifica dell'equivalenza tra gli automi single-string $\mathcal{A}$ e $\mathcal{A}'$

1: $(u, v) \leftarrow$ codifica di $\mathcal{A}$ {Algoritmo 3.1}
2: $(u, v) \leftarrow$ codifica minima di (u, v) {Algoritmo 3.2}
3: $(u', v') \leftarrow$ codifica di $\mathcal{A}'$
4: $(u', v') \leftarrow$ codifica minima di (u', v')
5: **return** $(u, v) = (u'v')$

3.2 Automi single-string estesi

Nella sezione precedente si è mostrato come gli automi single-string possano essere utilizzati per rappresentare parole fondamentalmente periodiche e quindi istanze della relazione $\trianglelefteq$ tra granularità. Molte granularità, pur avendo strutture relativamente semplici, necessitano comunque di automi con un numero elevato di stati per essere rappresentate. Per esempio, la relazione *Giorno* $\trianglelefteq$ *Anno* (considerando il calendario Gregoriano), può essere rappresentata solamente da automi single-string contenenti almeno $(366+1)\cdot 96+(365+1)\cdot 304+1$ stati. In questa sezione mostriamo una possibile soluzione al problema, modificando gli automi single-string introducendo *contatori* e *transizioni alternative* che possono essere attivate in determinate condizioni.

Dato un insieme finito di variabili V valutate sul dominio $\mathbb{N}$, denotiamo con L_V un generico linguaggio logico tale che:

- le variabili libere delle formule di L_V appartengono a V;
- i simboli funzionali e relazionali delle formule di L_V sono interpretati in $\mathbb{N}$ nel modo naturale.

Chiameremo *valutazione* ogni funzione $C : V \to \mathbb{N}$, e denoteremo con $\mathcal{C}$ l'insieme $\mathbb{N}^V$ di tutte le valutazioni sulle variabili di V. Le valutazioni $C \in \mathcal{C}$ verranno utilizzate per interpretare le formule del linguaggio L_V, definendo *modello* di una formula $\varphi \in L_V$ ogni valutazione C che rende vera φ. Useremo la notazione $C \models \varphi$ per indicare che C è un modello di φ. Con $\mathcal{C}^{\mathcal{C}}$ denotiamo inoltre l'insieme di tutte le funzioni ψ da $\mathcal{C}$ a $\mathcal{C}$.

Definizione 3.6. Un *automa single-string esteso* è una tupla $\mathcal{A} = (S, V, \Sigma, \delta, \gamma, s_I, C_I)$, dove:

- S è un insieme finito di stati;
- V è un insieme finito di variabili;
- Σ è un insieme finito di simboli;
- $\delta : S \to \mathcal{C}^{\mathcal{C}} \times \Sigma \times S$ è la *funzione totale di transizione primaria* di $\mathcal{A}$;
- $\gamma : S \to L_V \times \mathcal{C}^{\mathcal{C}} \times \Sigma \times S$ è la *funzione parziale di transizione secondaria* di $\mathcal{A}$;
- $s_I \in S$ è lo stato iniziale;
- $C_I \in \mathcal{C}$ è la *valutazione iniziale* delle variabili di V.

Negli automi single-string estesi esistono due tipi di transizioni: le *transizioni primarie*, codificate dalla funzione totale δ, e le *transizioni secondarie* codificate dalla funzione parziale γ. Per ogni automa single-string esteso $\mathcal{A} = (S, V, \Sigma, \delta, \gamma, s_I, C_I)$ diremo che *la transizione secondaria è attiva sullo stato s e sulla valutazione C* (o, più semplicemente che γ *è attiva in* $\langle s, C\rangle$) se e solo se $\gamma(s) = \langle \varphi, \psi, a, s'\rangle$ e $C \models \varphi$.

Definizione 3.7. Se $\mathcal{A} = (S, V, \Sigma, \delta, \gamma, s_I, C_I)$ è un automa single-string esteso, definiamo la relazione $\rightarrow_{\mathcal{A}}$: $(S \times \mathcal{C}) \times (S \times \mathcal{C} \times \Sigma)$ tale che $\langle s, C\rangle \rightarrow_{\mathcal{A}} \langle s', C', a'\rangle$ se e solo se vale una delle seguenti condizioni:

- $\gamma(s) = \langle \varphi, \psi, a'', s''\rangle$, $C \models \varphi$ (cioè γ è attiva in $\langle s, C\rangle$), $C' = \psi(C)$, $s' = s''$ e $a' = a''$;

- $\delta(s) = \langle \psi, a'', s'' \rangle$, $C \not\models \varphi$ (cioè γ non è attiva in $\langle s, C \rangle$), $C' = \psi(C)$, $s' = s''$ e $a' = a''$.

La relazione $\rightarrow_{\mathcal{A}}$ è una *funzione totale* da $S \times \mathcal{C}$ a $S \times \mathcal{C} \times \Sigma$: infatti le condizioni imposte dalla definizione sono in mutua esclusione e determinano esattamente uno stato s', una valutazione C' ed un simbolo a' tali che $\langle s, C \rangle \rightarrow_{\mathcal{A}} \langle s', C', a' \rangle$. Inoltre essa permette di stabilire in maniera precisa la semantica degli automi single-string estesi, secondo la definizione seguente.

Definizione 3.8. Se $\mathcal{A} = (S, V, \Sigma, \delta, \gamma, s_I, C_I)$ è un automa single-string esteso, definiamo *run di* $\mathcal{A}$ la sequenza infinita $\rho \in (S \times \mathcal{C} \times \Sigma)^{\omega}$ tale che:

- $\pi_1(\rho(1)) = s_I$;
- $\pi_2(\rho(1)) = C_I$;
- $\langle \pi_1(\rho(n)), \pi_2(\rho(n)) \rangle \rightarrow_{\mathcal{A}} \langle \pi_1(\rho(n+1)), \pi_2(\rho(n+1)), \pi_3(\rho(n)) \rangle$ per ogni $n \in \mathbb{N}^+$.

Diremo che $\mathcal{A}$ *riconosce la* ω*-parola* w se e solo se il run ρ è tale che $\pi_3(\rho) = w$.

Nella definizione delle funzioni di transizione di un automa single-string esteso è necessario indicare gli *operatori di modifica delle valutazioni*, ossia le funzioni ψ da $\mathcal{C}$ in $\mathcal{C}$. In questa tesi utilizzeremo una notazione intuitiva per rappresentare tali operatori: per esempio $i \leftarrow i + 1$ rappresenta l'operatore ψ tale che $\psi(C) = C'$ se e solo se $C'(i) = C(i) + 1$ e, per ogni altra variabile $v \neq i$, $C'(v) = C(v)$.

Un automa single-string esteso può essere rappresentato da un grafo orientato in maniera analoga agli automi single-string, con l'ulteriore convenzione di rappresentare le transizioni primarie mediante archi continui etichettati con un operatore $\psi \in \mathcal{C}^{\mathcal{C}}$ e un simbolo di Σ, e le transizioni secondarie mediante archi tratteggiati etichettati con una formula di L_V, un operatore di $\mathcal{C}^{\mathcal{C}}$ e un simbolo di Σ. La valutazione iniziale C_I viene indicata a parte, anche se generalmente si utilizza la valutazione C_0 tale che $C_0(v) = 0$ per ogni $v \in V$.

Esempio 3.9. La Figura 3.2 rappresenta un automa single-string che riconosce la parola ■■■■■□□ ≀ ■■■■■□□ ≀ .... L'automa riconosce quindi la stessa parola dell'automa single-string di Figura 3.1, utilizzando solamente due stati ed una variabile contatore i. Si assume che la valutazione iniziale sia C_0.

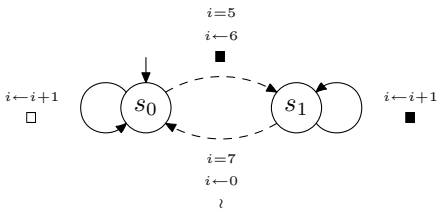

Figura 3.2: Esempio di automa single-string esteso

3.2.1 Automi single-string estesi riducibili

La definizione di automa single-string esteso data nella sezione precedente è molto generale e non effettiva: infatti non si pongono vincoli né sulle formule di

L_V né sull'estensione dell'insieme $\mathcal{C}^{\mathcal{C}}$ (che comprende anche funzioni non computabili), ed è possibile quindi che gli stessi passi di transizione dell'automa non siano computabili. Inoltre, molti problemi sugli automi single-string estesi non sono (in generale) decidibili, come il problema dell'equivalenza. Per poter utilizzare effettivamente gli automi single-string estesi nella rappresentazione delle granularità è quindi necessario porre opportune restrizioni nella loro definizione, per ottenere un compromesso tra la loro espressività e la decidibilità di alcuni problemi fondamentali.

In [DM01] vengono proposte alcune limitazioni sintattiche sulle formule di L_V e sugli operatori di $\mathcal{C}^{\mathcal{C}}$ utilizzabili per definire gli automi single-string estesi; in questo modo è sempre possibile costruire un automa single-string che riconosce la stessa parola e che permette di risolvere il problema dell'equivalenza. In questa sezione daremo la definizione di *automa single-string esteso riducibile* e studieremo una opportuna classe di automi che consente di risolvere molti problemi fondamentali in maniera semplice.

Definizione 3.10. Data una generica funzione $\mathcal{F} : A \to A \times B$, diciamo che una relazione di equivalenza θ sull'insieme A *rispetta la funzione* $\mathcal{F}$ se e solo se, per ogni coppia di elementi $a_1, a_2 \in A$, vale l'implicazione $a_1 \theta a_2 \wedge \mathcal{F}(a_1) = \langle c_1, b_1 \rangle \wedge \mathcal{F}(a_2) = \langle c_2, b_2 \rangle \Rightarrow c_1 \theta c_2$.

Definizione 3.11. Un automa single-string esteso $\mathcal{A} = (S, V, \Sigma, \delta, \gamma, s_I, C_I)$ si dice *riducibile* se e solo se esiste una relazione di equivalenza θ su $S \times \mathcal{C}$ che rispetta la funzione $\rightarrow_{\mathcal{A}}$ e ha indice finito (cioè il numero delle classi di equivalenza di θ è finito).

Il seguente teorema afferma che gli automi single-string estesi riducibili sono equivalenti, come espressività, agli automi single-string. Tuttavia gli automi single-string estesi riducibili consentono, in generale, di rappresentare le parole fondamentalmente periodiche mediante un numero inferiore di stati.

Teorema 3.12 (Espressività degli automi riducibili). *Se w è una parola riconosciuta da un automa single-string esteso riducibile, allora esiste un automa single-string che riconosce w.*

Dimostrazione. Sia $\mathcal{A} = (S, V, \Sigma, \delta, \gamma, s_I, C_I)$ un automa single-string esteso riducibile e sia θ la relazione di equivalenza su $S \times \mathcal{C}$ che rispetta $\rightarrow_{\mathcal{A}}$ e ha indice finito. Costruiamo l'automa single-string $\mathcal{A}' = (S', \Sigma, \delta', s'_I)$ tale che:

- $S' = \{[\langle s, C \rangle]_\theta \text{ t.c. } s \in S, C \in \mathcal{C}\}$ (dove l'insieme $[\langle s, C \rangle]_\theta = \{\langle s', C' \rangle \text{ t.c. } \langle s, C \rangle \theta \langle s', C' \rangle\}$ è la classe di equivalenza di $\langle s, C \rangle$ rispetto a θ);
- $\delta'([\langle s, C \rangle]_\theta) = \langle a, [\langle s', C' \rangle]_\theta \rangle$ se e solo se $\langle s, C \rangle \rightarrow_{\mathcal{A}} \langle s', C', a \rangle$ (poiché θ rispetta $\rightarrow_{\mathcal{A}}$ la funzione δ' è ben definita);
- $s'_I = [\langle s_I, C_I \rangle]_\theta$.

Dimostriamo che $\mathcal{A}$ e $\mathcal{A}'$ riconoscono la stessa parola: siano ρ e ρ' i run degli automi $\mathcal{A}$ e $\mathcal{A}'$. Se l'n-esimo elemento del run di $\mathcal{A}'$ è tale che $\pi_1(\rho'(n)) = [\pi_{1,2}(\rho(n))]_\theta$, allora $\rho'(n+1) = \langle [\langle s, C \rangle]_\theta, a \rangle$ se e solo se $\rho(n+1) = \langle s, C, a \rangle$. Poiché $\pi_1(\rho'(1)) = [\langle s_I, C_I \rangle]_\theta = [\pi_{1,2}(\rho(1))]_\theta$, allora $\pi_2(\rho') = \pi_3(\rho)$ e quindi $\mathcal{A}$ e $\mathcal{A}'$ riconoscono la stessa parola. □

3.2.2 Equivalenza tra automi effettivamente riducibili

È importante notare come, in generale, il problema dell'equivalenza tra automi (single-string estesi) riducibili non è decidibile. Infatti la funzione ξ, che restituisce per ogni automa riducibile l'automa single-string equivalente, non è computabile. Se $\mathcal{R}$ è una classe di automi *effettivamente riducibili*, cioè se la funzione $\xi|_{\mathcal{R}}$ è computabile, allora il problema dell'equivalenza è decidibile in $\mathcal{R}$.

Per ottenere una classe di automi effettivamente riducibili è possibile definire alcuni vincoli sintattici sulle formule di L_V e sulle funzioni che modificano le valutazioni di V. Sia dunque $\mathcal{D}$ la classe di automi estesi tali che:

- il linguaggio L_V, usato per definire la funzione di transizione secondaria, è costituito da formule del tipo

  $$\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n,$$

  dove, per ogni $i \in [1, n]$, φ_i è una uguaglianza o una disuguaglianza tra costanti o espressioni del tipo v *mod* c (dove $v \in V$ e c è una costante);
- gli operatori di $\mathcal{C}^{\mathcal{C}}$ utilizzati per modificare le valutazioni delle variabili di V sono composizioni di operatori come $v \leftarrow c$ o $v \leftarrow v + c$ (dove $v \in V$ e c è una costante).

Esempio 3.13. La Figura 3.3 rappresenta un automa appartenente alla classe $\mathcal{D}$ che riconosce la parola periodica
$\blacksquare^{31} \wr \blacksquare^{28} \wr \blacksquare^{31} \wr \blacksquare^{30} \wr \blacksquare^{31} \wr \blacksquare^{30} \wr \blacksquare^{31} \wr \blacksquare^{31} \wr \blacksquare^{30} \wr \blacksquare^{31} \wr \blacksquare^{30} \wr \blacksquare^{31} \wr \blacksquare^{31} \wr \blacksquare^{28} \wr \ldots$,
che rappresenta la relazione *Giorno* $\trianglelefteq^{fp}$ *PseudoMese* utilizzando solamente cinque stati e due contatori. Si noti che una Granspec avrebbe richiesto un periodo di lunghezza almeno $365 + 12 = 377$ simboli.

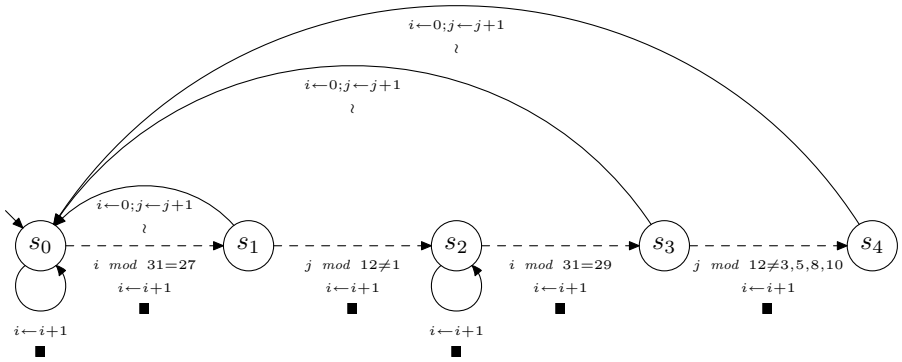

Figura 3.3: Esempio di automa effettivamente riducibile

La seguente proposizione, assieme ai risultati del Teorema 3.12, implica che il problema dell'equivalenza per la classe di automi $\mathcal{D}$ è decidibile.

Proposizione 3.14. *Gli automi appartenenti alla classe $\mathcal{D}$ sono effettivamente riducibili, ossia la funzione $\xi|_{\mathcal{D}}$, che restituisce per ogni automa $\mathcal{A}$ appartenente a $\mathcal{D}$ l'automa single-string equivalente, è computabile.*

Dimostrazione. Dimostriamo la proposizione fornendo, per ogni automa $\mathcal{A} \in \mathcal{D}$, una relazione di equivalenza θ su $S \times \mathcal{C}$ che rispetta la funzione $\rightarrow_{\mathcal{A}}$ e ha indice finito.

Sia $\mathcal{A} = (S, V, \Sigma, \delta, \gamma, s_I, C_I)$ un automa appartenente alla classe $\mathcal{D}$. Indichiamo con $v_1, v_2, \ldots, v_n$ le variabili di V e con d_i, al variare di i in $[1, n]$, il minimo comune multiplo tra tutte le costanti che compaiono nei termini del tipo $v_i \ mod \ c$ delle formule di $\mathcal{A}$. Definiamo la relazione θ su $S \times \mathcal{C}$ tale che $\langle s, C\rangle \theta \langle s', C'\rangle$ se e solo se $s = s'$ e, per ogni $i \in [1, n]$, $C(v_i) \ mod \ d_i = C'(v_i) \ mod \ d_i$.

L'equivalenza θ è chiaramente di indice finito; verifichiamo quindi che rispetta $\rightarrow_{\mathcal{A}}$. Siano s, s', s'', s''' stati appartenenti a S, C, C', C'', C''' valutazioni di V e a'', a''' simboli di Σ tali che:

- $\langle s, C\rangle \rightarrow_{\mathcal{A}} \langle s'', C'', a''\rangle$;
- $\langle s', C'\rangle \rightarrow_{\mathcal{A}} \langle s''', C''', a'''\rangle$;
- $\langle s, C\rangle \theta \langle s', C'\rangle$.

Se $\phi = \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_m$ è la formula utilizzata per definire la transizione secondaria $\gamma(s)$ allora, per ogni $j \in [1, m]$, $C \models \varphi_j$ se e solo se $C' \models \varphi_j$ (infatti i termini nella forma $v \ mod \ d$ hanno interpretazioni coincidenti in C e C'). Si ricava quindi che γ è attiva in $\langle s, C\rangle$ se e solo se è attiva in $\langle s, C'\rangle = \langle s', C'\rangle$ e quindi $s'' = s'''$. Inoltre, se $\psi = f_1 \cdot f_2 \cdot \ldots \cdot f_k$ è l'operatore utilizzato per definire la transizione $\delta(s)$ o $\gamma(s)$ allora, per ogni $i \in [1, n]$ e per ogni $j \in [1, k]$, vale l'implicazione $C(v_i) \ mod \ d_i = C'(v_i) \ mod \ d_i \Rightarrow f_j(C(v_i)) \ mod \ d_i = f_j(C'(v_i)) \ mod \ d_i$. Possiamo quindi concludere che $C''(v_i) \ mod \ d_i = C'''(v_i) \ mod \ d_i$ e quindi $\langle s, C\rangle \theta \langle s', C'\rangle$. In questo modo si dimostra che θ rispetta la funzione $\rightarrow_{\mathcal{A}}$.

Dalla definizione della relazione θ segue inoltre immediatamente che la funzione $\xi|_{\mathcal{D}}$, che restituisce per ogni automa $\mathcal{A}$ appartenente a $\mathcal{D}$ l'automa single-string equivalente, è computabile. □

Concludiamo la discussione sugli automi effettivamente riducibili presentando l'Algoritmo 3.4 che riceve in input l'automa $\mathcal{A}$ appartenente alla classe $\mathcal{D}$ e computa l'automa single-string $\mathcal{A}'$ equivalente. Per stabilire se due automi appartenenti alla classe $\mathcal{D}$ riconoscono la stessa parola è quindi sufficiente costruire i due automi single-string equivalenti e verificare se riconoscono la stessa parola mediante l'Algoritmo 3.3.

Data la semplicità dell'algoritmo, non verrà data alcuna dimostrazione di correttezza. Si osservi invece come la complessità dell'algoritmo di conversione sia $O(|V| \cdot |Q'|)$, dove V è l'insieme delle variabili dell'automa $\mathcal{A}$ fornito in input e Q' è l'insieme degli stati dell'automa $\mathcal{A}'$ risultante. Non è possibile fornire una limitazione rispetto alla dimensione dell'input; tuttavia, se non vi sono ridondanze nelle valutazioni di $\mathcal{A}$, la dimensione dell'automa $\mathcal{A}'$ fornito in output è comparabile alla lunghezza del prefisso e del periodo della parola riconosciuta.

Algoritmo 3.4 Calcolo dell'automa single-string $\mathcal{A}'$ equivalente all'automa $\mathcal{A} \in \mathcal{D}$

let $\mathcal{A} = (S, V, \Sigma, \delta, \gamma, s_I, C_I)$
let $\mathcal{A}' = (S, \Sigma, \delta, s_I)$
for all $v_i \in V$ **do**
  $d_i \leftarrow m.c.m.\{$costanti nei termini $v_i \ mod \ d$ delle formule di $\gamma\}$
end for
$S' \leftarrow S \times [0, d_1[\times [0, d_2[\times \ldots \times [0, d_n[$
for all $\bar{s} \in S'$ **do**
  let $\bar{s} = \langle s, \langle x_1, x_2, \ldots, x_n \rangle \rangle$
  for all $i \in [1, n]$ **do**
    $C[v_i] \leftarrow x_i$
  end for
  let $\langle s, C \rangle \rightarrow_{\mathcal{A}} \langle s', C', a' \rangle$
  for all $i \in [1, n]$ **do**
    $x'_i \leftarrow C'[v_i]$
  end for
  let $\bar{s}' = \langle s', \langle x'_1, x'_2, \ldots, x'_n \rangle \rangle$
  $\delta'[\bar{s}] \rightarrow \langle a', \bar{s}' \rangle$
end for
for all $i \in [1, n]$ **do**
  $x_i \leftarrow C_I[v_i] \ mod \ d_i$
end for
let $\bar{s}_I = \langle s_I, \langle x_1, x_2, \ldots, x_n \rangle \rangle$
$\mathcal{A}' = (S', \Sigma, \delta', \bar{s}_I)$
return $\mathcal{A}'$

3.3 Automi etichettati ristretti

In questa sezione presentiamo una ulteriore classe di automi, detti *automi single-string etichettati ristretti* (o, più semplicemente, automi etichettati ristretti), proposta da Dal Lago, Montanari e Puppis in [DMP03a] per estendere gli automi single-string. La differenza principale con le altre classi di automi presentate è la struttura delle funzioni di transizione, che risulta più ristretta. Tuttavia ciò rende possibile la risoluzione di molti problemi interessanti (riguardanti, in particolare, le granularità temporali) in modo efficiente. In questa tesi mostreremo in dettaglio come affrontare il problema dell'equivalenza; in [DMP03a] viene inoltre illustrato come gli operatori della Calendar Algebra possano essere effettivamente implementati in questo formalismo.

Dato un insieme finito S di stati, chiameremo *valutazione su* S ogni funzione $C : S \rightarrow \mathbb{N}$. La definizione di automa etichettato ristretto è quindi la seguente.

Definizione 3.15. Un *automa (single-string) etichettato ristretto* (o RLA, restricted labelled automata) è una tupla $\mathcal{A} = (S_\Sigma, S_\varepsilon, \Sigma, \mathcal{L}, \delta, \gamma, s_I, C_I)$, dove:

- S_Σ e S_ε sono insiemi disgiunti di *stati*; con S denotiamo l'insieme $S_\Sigma \cup S_\varepsilon$;
- Σ è un alfabeto di simboli;
- $\mathcal{L} : S_\Sigma \rightarrow \Sigma$ è la *funzione di etichettatura*;
- $\delta : S \rightarrow S$ è la *funzione di transizione primaria*;

- $\gamma : S_\varepsilon \to S$ è la *funzione di transizione secondaria* tale che:
  1. per ogni $s \in S_\varepsilon$, $(\gamma(s), s)$ appartiene alla chiusura transitiva e riflessiva δ^* di δ; il minimo $n \in \mathbb{N}$ tale che $(\gamma(s), s) \in \delta^n(s)$ è detto *grado di* s e $\Gamma_\mathcal{A} \subseteq S \times S_\varepsilon$ è la relazione tale che $(s, r) \in \Gamma_\mathcal{A}$ se e solo se $r = \delta^i(\gamma(s))$, con i minore o uguale al grado di s;
  2. la chiusura riflessiva e transitiva $\Gamma^*_\mathcal{A}$ di $\Gamma_\mathcal{A}$ è antisimmetrica;
- $s_I \in S$ è lo *stato iniziale*;
- $C_I : S_\varepsilon \to \mathbb{N}$ è la *valutazione iniziale*.

Si noti come la funzione di etichettatura sia definita sugli stati dell'automa (da cui il nome di automa *etichettato*): questo significa che un simbolo deve essere letto ogni volta che l'automa raggiunge uno stato etichettato, e non ogni volta che viene attivata una transizione, come accade generalmente negli automi e nei sistemi di transizione etichettati. Le restrizioni sulla funzione di transizione secondaria (condizioni 1. e 2.) sono state introdotte per ottenere un'utile ordinamento sulla struttura delle transizioni, come risulterà chiaro nella sezione successiva.

Nel caso degli automi etichettati ristretti denotiamo con $\mathcal{C}$ la classe $\mathbb{N}^{S_\varepsilon}$ di tutte le valutazioni dell'automa.

Definizione 3.16. Se $\mathcal{A} = (S_\Sigma, S_\varepsilon, \Sigma, \mathcal{L}, \delta, \gamma, s_I, C_I)$ è un automa etichettato ristretto, definiamo la funzione $\to_A$: $S \times \mathcal{C} \to S \times \mathcal{C}$ tale che $\langle s, C\rangle \to_A \langle s', C'\rangle$ se e solo se vale una delle seguenti condizioni:

1. $s \in S_\Sigma$, $s' = \delta(s)$ e $\forall t \in S. C'(t) = C(t)$;
2. $s \in S_\varepsilon$, $C(s) \neq 0$, $s' = \gamma(s)$, $C'(s) = C(s) - 1$ e $\forall t \neq s. C'(t) = C(t)$;
3. $s \in S_\varepsilon$, $C(s) = 0$, $s' = \delta(s)$, $C'(s) = C_I(s)$ e $\forall t \neq s. C'(t) = C(t)$.

Se $C(s) \neq 0$ diremo che *la funzione di transizione* γ *è attiva in* $\langle s, C\rangle$.

Le condizioni imposte dalla definizione sono in mutua esclusione e determinano al più uno stato s' e una valutazione C' tale che $\langle s, C\rangle \to_\mathcal{A} \langle s', C'\rangle$, quindi la funzione $\to_\mathcal{A}$ è ben fondata ed è una *funzione totale*. Ciò permette di definire in maniera precisa la semantica degli automi single-string etichettati ristretti. Con $\to^*_\mathcal{A}$ denotiamo la chiusura transitiva di $\to_\mathcal{A}$.

Definizione 3.17. Se $\mathcal{A} = (S_\Sigma, S_\varepsilon, \Sigma, \mathcal{L}, \delta, \gamma, s_I, C_I)$ è un automa etichettato ristretto, definiamo *run di* $\mathcal{A}$ la sequenza infinita $\rho \in (S \times \mathcal{C})^\omega$ tale che:

- $\rho(1) = \langle s_I, C_I\rangle$;
- $\rho(i) \to_\mathcal{A} \rho(i+1)$, per ogni $i \in \mathbb{N}^+$.

Se ρ è il run dell'automa $\mathcal{A}$, definiamo $\rho_\Sigma \in S^*_\Sigma \cup S^\omega_\Sigma$ la sequenza di stati ottenuta a partire da ρ eliminando le configurazioni i cui stati appartengono a S_ε. Diremo che $\mathcal{A}$ *riconosce la parola (finita o infinita)* w se e solo se $w = \mathcal{L}(\rho_\Sigma)$.

Si noti che, mentre il run di $\mathcal{A}$ è sempre una sequenza infinita di configurazioni, la sequenza ρ_Σ, e quindi la parola riconosciuta da $\mathcal{A}$, può essere anche una sequenza finita. Di conseguenza, gli automi etichettati ristretti riconoscono sia *parole fondamentalmente periodiche* che *parole finite*. Nell'ultimo caso si può definire un insieme F di *stati finali* tali che $F \subseteq S_\varepsilon$, $\delta(F) = F$ (cioè il grafo (F, δ) è un insieme di cicli disgiunti) e $\gamma(F) \subseteq F$ (cioè le transizioni secondarie che partono da F non possono uscire da F). Inoltre, quando si ha a che fare con

automi etichettati ristretti che riconoscono parole finite, i dettagli sulla struttura degli stati finali possono essere ignorati: si può dimostrare infatti che tutte le strutture possibili sono equivalenti.

Nella rappresentazione degli automi etichettati ristretti mediante grafi orientati si adottano le seguenti convenzioni:

- gli stati di S_Σ sono rappresentati da cerchi etichettati con il nome dello stato o con il simbolo di Σ corrispondente alla loro etichettatura;
- gli stati di S_ε sono rappresentati da triangoli;
- le transizioni primarie sono rappresentate da archi continui;
- le transizioni secondarie sono rappresentate da archi tratteggiati;
- il valore della valutazione iniziale è mostrato vicino al corrispondente stato di S_ε;
- lo stato iniziale è identificato da una freccia, mentre gli (eventuali) stati finali sono identificati da una doppia cerchiatura.

Esempio 3.18. L'automa etichettato ristretto di Figura 3.4 riconosce la parola infinita $(\blacksquare \wr (\square\wr)^6)^\omega$, che rappresenta la relazione *Giorni* $\trianglelefteq$ *Lunedì*. Si noti che sono sufficienti quattro stati per rappresentare una parola che possiede periodo minimo di lunghezza 14.

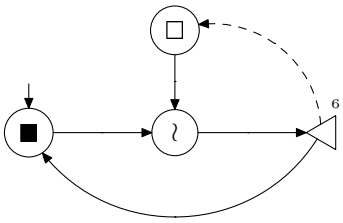

Figura 3.4: Esempio di automa etichettato ristretto

3.3.1 Proprietà degli automi etichettati ristretti

Per semplificare la descrizione delle proprietà degli automi etichettati ristretti, introduciamo le notazioni $[s,t]_\delta$, $[s,t[_\delta$ e $[s,\infty[_\delta$ che denotano, rispettivamente, gli insiemi finiti di stati $\{s,\delta(s),\delta^2(s),\ldots,\delta^n(s)\}$, $\{s,\delta(s),\delta^2(s),\ldots,\delta^{n-1}(s)\}$ e $\{s,\delta(s),\delta^2(s),\delta^3(s),\ldots\}$, dove δ è la funzione di transizione primaria dell'automa $\mathcal{A}$ e s e t sono due stati tali che $\delta^n(s)=t$ e $\forall i<n.\delta^i(s)\neq t$.

Sia $\mathcal{A}=(S_\Sigma,S_\varepsilon,\Sigma,\mathcal{L},\delta,\gamma,s_I,C_I)$ un automa etichettato ristretto; definiamo ricorsivamente la sequenza $\rho_s\in S^*\cup S^\omega$ come segue:

1. $\forall s\in S_\Sigma.\rho_s=s$;
2. $\forall s\in S_\varepsilon.\rho_s=\left(\rho_{\gamma(s)}\cdot\rho_{\delta(\gamma(s))}\cdot\rho_{\delta^2(\gamma(s))}\cdot\ldots\rho_{\delta^{n-1}(\gamma(s))}\right)^{C_I(s)}$;

dove n è il minimo intero tale che $s = \delta^n(\gamma(s))$. Si noti come la definizione risulti ben fondata grazie ai requisiti posti dalla Definizione 3.15 alle funzioni di transizione primaria e secondaria.

Il teorema seguente sfrutta invece tali requisiti per caratterizzare le parole riconosciute dagli RLA mediante espressioni del tipo $(abc)^3$, $a^2(bc)^\omega$ o $(a^2b^3)^4$.

Teorema 3.19. *L'automa etichettato ristretto* $\mathcal{A} = (S_\Sigma, S_\varepsilon, \Sigma, \mathcal{L}, \delta, \gamma, s_I, C_I)$ *riconosce la parola (finita o infinita)*

$$u = \mathcal{L}(\rho_{s_I} \cdot \rho_{\delta(s_I)} \cdot \rho_{\delta^2(s_I)} \cdot \ldots).$$

Inoltre gli automi etichettati ristretti, così come gli automi di Büchi, sono chiusi per concatenazione, ripetizione ed iterazione di parole, e per selezione di sottostringhe. Formalmente, dati due automi etichettati ristretti $\mathcal{A}$ e $\mathcal{B}$ che riconoscono rispettivamente le parole u e v, e dati tre parametri $k \in \mathbb{N}$, $m \in \mathbb{N}^+$ e $n \in \mathbb{N}^+ \cup \{\omega\}$, esistono:

- un automa $\mathcal{A} \cdot \mathcal{B}$ (*concatenazione* di $\mathcal{A}$ e $\mathcal{B}$) che riconosce la parola $u \cdot v$;
- un automa $\mathcal{A}^k$ (*ripetizione* di $\mathcal{A}$) che riconosce la parola u^k;
- un automa $\mathcal{A}^\omega$ (*iterazione* di $\mathcal{A}$) che riconosce la parola u^ω;
- un automa $\mathcal{A}[m, n[$ (*selezione di sottostringa* di $\mathcal{A}$) che riconosce la parola $u[m, n[$.

Le proprietà di chiusura così definite sono inoltre effettive, cioè è possibile costruire, mediante appropriati algoritmi, l'automa concatenazione di $\mathcal{A}$ e $\mathcal{B}$, l'automa ripetizione $\mathcal{A}^k$, l'automa iterazione $\mathcal{A}^\omega$ e l'automa selezione di sottostringa $\mathcal{A}[m, n[$ a partire dagli automi $\mathcal{A}$ e $\mathcal{B}$. Per la loro semplicità, il codice di tali algoritmi non viene riportato. Le Figure 3.5, 3.6, 3.7 e 3.8 mostrano alcuni esempi della costruzione di tali automi.

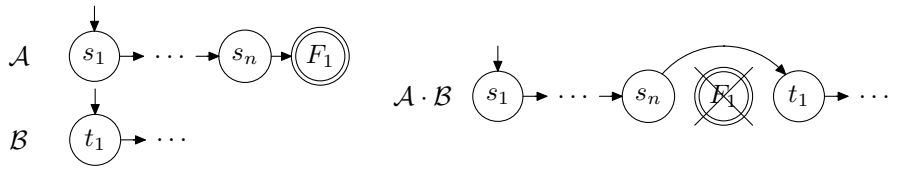

Figura 3.5: Concatenazione di due automi etichettati ristretti

Figura 3.6: Ripetizione di un automa etichettato ristretto

Figura 3.7: Iterazione di un automa etichettato ristretto

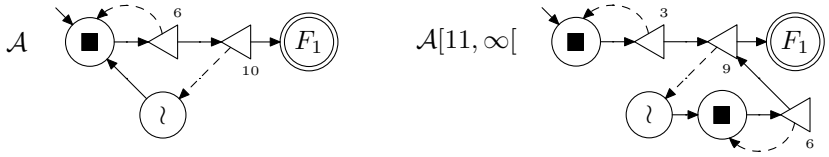

Figura 3.8: Selezione di una sottostringa di un automa etichettato ristretto

3.3.2 Equivalenza tra automi etichettati ristretti

Anche nel caso degli automi etichettati ristretti il problema dell'equivalenza si risolve confrontando le codifiche minime delle parole riconosciute dai due automi. Il metodo per determinare una codifica della parola riconosciuta da un automa etichettato ristretto non necessita però della sua conversione in un automa single-string, ma sfrutta le proprietà dell'automa per determinare la lunghezza del prefisso e del periodo e simula quindi le transizioni per determinare effettivamente le parole u e v tali che (u, v) è una codifica della parola riconosciuta dall'automa.

Definizione 3.20. Se $\mathcal{A} = (S_\Sigma, S_\varepsilon, \Sigma, \mathcal{L}, \delta, \gamma, s_I, C_I)$ è un automa etichettato ristretto, chiamiamo *loop-configuration* ogni coppia ordinata $\langle s, C\rangle \in S \times \mathcal{C}$ tale che:

- $\langle s_I, C_I\rangle \rightarrow^*_{\mathcal{A}} \langle s, C\rangle$, cioè $\langle s, C\rangle$ è raggiungibile a partire dalla configurazione iniziale;
- $\langle s, C\rangle \rightarrow^*_{\mathcal{A}} \langle s, C\rangle$.

Chiaramente, se $\langle s, C\rangle$ è una loop-configuration di $\mathcal{A}$, ogni simbolo della sequenza $\mathcal{L}(\rho_s)$ occorre infinite volte nella parola riconosciuta, con periodo esattamente uguale a $\sum_{t\in[s,\omega[_\delta} |\rho_t|$. Se si conoscono i valori di $|\rho_s|$ per ogni stato s dell'automa la lunghezza del prefisso e del periodo di una codifica della parola riconosciuta da $\mathcal{A}$ può quindi facilmente essere calcolata, una volta che si è determinata una loop-configuration.

Utilizzando le equazioni 1 e 2 date nella definizione di ρ_s il valore di $|\rho_s|$ può essere facilmente calcolato, per ogni stato $s \in S$, in tempo polinomiale. In questa tesi si assumerà invece che questi valori vengano calcolati durante la costruzione dell'automa $\mathcal{A}$ e memorizzati in un'apposita struttura dati: più precisamente, per ogni stato $s \in S$ il valore di $|\rho_s|$ è memorizzato in $\mathcal{A}.rholength[s]$. L'Algoritmo 3.5 sfrutta tali valori per calcolare la lunghezza del prefisso e del periodo di una codifica della parola riconosciuta da $\mathcal{A}$.

Algoritmo 3.5 Calcolo della lunghezza del prefisso e del periodo di una codifica dell'RLA $\mathcal{A}$

1: **let** $\mathcal{A} = (S_\Sigma, S_\varepsilon, \Sigma, \mathcal{L}, \delta, \gamma, s_I, C_I)$
2: **for all** $s \in S$ **do**
3: $\quad p[s] \leftarrow 0$
4: **end for**
5: $i \leftarrow 1$
6: $s \leftarrow s_I$
7: **while** $p[s] = 0$ **do**
8: $\quad p[s] \leftarrow i$
9: $\quad i \leftarrow i + \mathcal{A}.rholength[s]$
10: $\quad s \leftarrow \delta(s)$
11: **end while**
12: **return** $(p[s], i - p[s])$

Dopo aver determinato la lunghezza del prefisso e la lunghezza del periodo, l'Algoritmo 3.6, riportato di seguito, determina una codifica (u, v) della parola riconosciuta da $\mathcal{A}$ simulando le transizioni dell'automa.

Algoritmo 3.6 Calcolo di una codifica della parola riconosciuta dall'RLA $\mathcal{A}$

1: **let** $\mathcal{A} = (S_\Sigma, S_\varepsilon, \Sigma, \mathcal{L}, \delta, \gamma, s_I, C_I)$
2: $(l_1, l_2) \leftarrow$ lunghezza del prefisso e del periodo {Algoritmo 3.5}
3: $\langle s, C\rangle \leftarrow \langle s_I, C_I\rangle$
4: $i \leftarrow 1$
5: $u \leftarrow \varepsilon$
6: **while** $i \leq l_1$ **do**
7: $\quad$ **if** $s \in S_\Sigma$ **then**
8: $\quad\quad u \leftarrow u \cdot \mathcal{L}(s)$
9: $\quad\quad i \leftarrow i + 1$
10: $\quad$ **end if**
11: $\quad$ **let** $\langle s', C'\rangle$ t.c. $\langle s, C\rangle \rightarrow_{\mathcal{A}} \langle s', C'\rangle$
12: $\quad \langle s, C\rangle \leftarrow \langle s', C'\rangle$
13: **end while**
14: $v \leftarrow \varepsilon$
15: **while** $i \leq l_1 + l_2$ **do**
16: $\quad$ **if** $s \in S_\Sigma$ **then**
17: $\quad\quad v \leftarrow v \cdot \mathcal{L}(s)$
18: $\quad\quad i \leftarrow i + 1$
19: $\quad$ **end if**
20: $\quad$ **let** $\langle s', C'\rangle$ t.c. $\langle s, C\rangle \rightarrow_{\mathcal{A}} \langle s', C'\rangle$
21: $\quad \langle s, C\rangle \leftarrow \langle s', C'\rangle$
22: **end while**
23: **return** (u, v)

Le condizioni imposte dalla Definizione 3.16 alla funzione di transizione $\rightarrow_A$ permettono di calcolare effettivamente, a partire da una configurazione $\langle s, C\rangle \in S \times \mathcal{C}$, la configurazione $\langle s', C'\rangle$ tale che $\langle s, C\rangle \rightarrow_{\mathcal{A}} \langle s', C'\rangle$. L'Algoritmo 3.6 è quindi effettivo e determina una codifica $\langle u, v\rangle$ della parola riconosciuta da $\mathcal{A}$ in tempo lineare rispetto alla lunghezza della codifica.

Mostriamo infine l'Algoritmo 3.7, che verifica l'equivalenza tra due automi etichettati ristretti.

Algoritmo 3.7 Verifica dell'equivalenza tra gli automi etichettati ristretti $\mathcal{A}$ e $\mathcal{A}'$

1: $(u, v) \leftarrow$ codifica di $\mathcal{A}$ {Algoritmo 3.6}
2: $(u, v) \leftarrow$ codifica minima di (u, v) {Algoritmo 3.2}
3: $(u', v') \leftarrow$ codifica di $\mathcal{A}'$
4: $(u', v') \leftarrow$ codifica minima di (u', v')
5: **return** $(u, v) = (u'v')$

3.3.3 Implementazione della Calendar Algebra

Abbiamo già menzionato il fatto che i formalismi basati su automi proposti in questo capitolo sono espressivi almeno quanto il formalismo basato su stringhe proposto da Wijsen. Dal Lago, Montanari e Puppis mostrano come gli automi etichettati ristretti (oppure, equivalentemente, gli automi single-string o gli automi single-string riducibili) permettano di rappresentare tutte le granularità catturate dal formalismo della Calendar Algebra.

In [DMP03a] sono infatti proposti alcuni algoritmi che permettono di costruire, a partire da un'espressione della Calendar Algebra, un automa etichettato ristretto equivalente che rappresenta la stessa granularità nei termini della granularità di base. È importante notare come il formalismo della Calendar Algebra non permetta di decidere se due espressioni sono equivalenti (cioè se rappresentano la stessa granularità): la conversione delle espressioni della Calendar Algebra in automi permette invece di risolvere tale problema utilizzando gli algoritmi per la verifica dell'equivalenza tra automi illustrati in questo capitolo.

Anche le operazioni di conversione tra granularità (cfr. Sezione 1.4.4) possono essere implementate negli automi etichettati ristretti mediante opportuni algoritmi che restituiscono i valori delle funzioni $\lceil i \rceil_G^H$ e $\lfloor z \rfloor_G^H$ in tempo polinomiale rispetto al numero di stati degli automi coinvolti. I formalismi basati su automi presentati in questo capitolo permettono quindi una facile ed efficiente rappresentazione e manipolazione delle granularità temporali e delle loro informazioni associate.

Il limite principale di questi formalismi è che essi possono rappresentare solamente una singola granularità per ogni automa, mentre in alcuni casi risulta utile rappresentare insiemi (anche infiniti) di granularità, come le cosiddette *granularità dinamiche*, cioè granularità non ancorate al dominio temporale sottostante. Nel prossimo capitolo si mostrerà come la teoria degli automi (in particolare gli automi di Büchi) consenta di definire formalismi che permettono di rappresentare insiemi infiniti di granularità.

Capitolo 4

Automi fondamentalmente periodici

La discussione sugli automi single-string e sulle loro evoluzioni, come gli automi etichettati ristretti, mostra come la teoria degli automi possa essere utilizzata efficacemente per rappresentare istanze della relazione $\trianglelefteq$ tra granularità etichettate su interi positivi ed eseguire operazioni su di esse. Tuttavia tali automi consentono di rappresentare una sola istanza della relazione tra granularità, mentre in diverse applicazioni si può presentare la necessità di operare su insiemi di (relazioni tra) granularità e, conseguentemente, di trovare adeguati metodi di rappresentazione di tali insiemi.

Gli automi di Büchi ed i linguaggi ω-regolari da essi riconosciuti sono lo strumento principale utilizzato per rappresentare insiemi di parole infinite nella teoria degli automi (si veda in proposito [Tho90]). In questo capitolo si stabilisce quali insiemi di relazioni tra granularità (rappresentate con parole fondamentalmente periodiche) possono essere riconosciuti dagli automi di Büchi e si definisce una classe di automi che riconoscono esattamente tali insiemi.

4.1 Automi di Büchi e linguaggi ω-regolari

Presentiamo in questa sezione alcuni risultati di base della teoria degli automi di Büchi che verranno utilizzati successivamente per caratterizzare i linguaggi di sole parole fondamentalmente periodiche.

Gli automi di Büchi sono automi a stati finiti non deterministici che riconoscono parole infinite che attraverso computazioni (sequenze di stati) in cui almeno uno stato finale occorre infinite volte.

Definizione 4.1 (Automa di Büchi). Un *automa di Büchi* su un alfabeto A è una quadrupla $\mathcal{A} = (Q, \Delta, q_0, F)$ dove:

- Q è un insieme finito di *stati*;
- $\Delta \subseteq Q \times A \times Q$ è la *relazione di transizione* dell'automa;
- $q_0 \in Q$ è lo *stato iniziale* dell'automa;
- F è l'insieme finito degli *stati finali*.

Un *run* di $\mathcal{A}$ su una parola (finita o infinita) $w = w(1)w(2)\ldots$ è una sequenza

(finita o infinita) $\rho = \rho(1)\rho(2)\ldots$ di stati tale che $(\rho(i), w(i), \rho(i+1)) \in \Delta$ per ogni $1 \leq i \leq |w|$, se w è finita, e per ogni $i \geq 1$ se w è infinita.

Se w è una parola finita, non vuota, e $p, q \in Q$, scriviamo $p \xrightarrow[\mathcal{A}]{w} q$ per indicare che esiste un run ρ di $\mathcal{A}$ sulla parola w che inizia in p e termina in q. Se ρ contiene uno stato finale si scrive $p \xrightarrow[\mathcal{A}]{w} q$, in caso contrario si scrive $p \xrightarrow[\mathcal{A}]{w} q$. Dati due stati p, q di $\mathcal{A}$, indichiamo con $p \longrightarrow^* q$ il fatto che lo stato q è raggiungibile a partire da p nell'automa $\mathcal{A}$ (ossia che esiste una parola v tale che $p \xrightarrow[\mathcal{A}]{v} q$). Per definizione, ogni stato è raggiungibile da se stesso, cioè $p \rightarrow^* p$ per ogni $p \in Q$.

Un run di $\mathcal{A}$ su una parola infinita w è detto *accettante* se $\rho(1) = q_0$ e $\mathrm{Inf}(\rho) \cap F \neq \emptyset$, cioè se il run inizia dallo stato iniziale e contiene almeno uno stato finale che si ripete infinite volte. L'automa $\mathcal{A}$ *accetta* la parola w se esiste un run accettante di $\mathcal{A}$ su w. L'ω-linguaggio $L(\mathcal{A})$ *riconosciuto* da $\mathcal{A}$ è definito come segue:

$$L(\mathcal{A}) = \{w \in A^\omega | \mathcal{A} \text{ accetta } w\}$$

Se $L \subseteq A^\omega$ è uguale a $L(\mathcal{A})$, per un opportuno automa di Büchi $\mathcal{A}$, L è detto *ω-regolare*.

I linguaggi ω-regolari sono chiusi rispetto alle principali operazioni sui linguaggi: chiusura di Kleene infinita, concatenazione, unione, intersezione e complementazione.

Lemma 4.2 (Proprietà di chiusura degli automi di Büchi).

(a) *Se $V \subseteq A^*$ è un linguaggio regolare, allora V^ω è un linguaggio ω-regolare.*

(b) *Se $U \subseteq A^*$ è regolare e $V \subseteq A^\omega$ è un linguaggio ω-regolare, allora il linguaggio $U \cdot V$ è ω-regolare.*

(c) *Se $L_1, L_2 \subseteq A^\omega$ sono linguaggi ω-regolari, allora $L_1 \cap L_2$ e $L_1 \cup L_2$ sono linguaggi ω-regolari.*

(d) *Se $L \subseteq A^\omega$ è un linguaggio ω-regolare, allora anche $\overline{L} = A^\omega \setminus L$ è un linguaggio ω-regolare.*

Dimostrazione. Descriviamo brevemente il metodo di costruzione degli automi che riconoscono i linguaggi definiti dal punto *(a)*, poiché essi verranno utilizzati successivamente. Per la dimostrazione della correttezza di tale costruzione e per la descrizione dei metodi di costruzione degli automi che riconoscono i linguaggi dei punti *(b)*, *(c)* e *(d)*, si veda [Tho90].

Poiché $V^\omega = (V \setminus \{\varepsilon\})^\omega$ si assuma, senza perdita di generalità, che V non contenga la parola vuota. Si assuma, inoltre, che l'automa regolare $\mathcal{A} = (Q, \Delta, q_0, F)$ che riconosce V sia tale che q_0 non possieda transizioni entranti. L'automa di Büchi $\mathcal{A}'$ che riconosce V^ω si ottiene aggiungendo una transizione (q, a, q_0) per ogni transizione (q, a, q') tale che $q' \in F$ e ponendo q_0 come l'unico stato finale di $\mathcal{A}'$. Formalmente $\mathcal{A}' = (Q', \Delta', q_0, F')$ è definito come segue:

- $Q' = Q$,
- $\Delta' = \Delta \cup \{(q, a, q_0) | \exists q' \in F.(q, a, q') \in \Delta\}$,
- $F' = \{q_0\}$.

È importante osservare come ogni run accettante dell'automa $\mathcal{A}'$ contenga infinite occorrenze dello stato q_0 e come tra un'occorrenza di q_0 e la successiva l'automa debba necessariamente riconoscere una parola v appartenente al linguaggio V che genera V^ω. □

I linguaggi ω-regolari godono quindi delle proprietà di chiusura rispetto alle principali operazioni. Inoltre, tali operazioni sono effettive, ossia è possibile costruire l'automa che riconosce la composizione di linguaggi ω-regolari a partire dagli automi che riconoscono i linguaggi di partenza. Infine, le operazioni oggetto del Lemma 4.2 consentono di caratterizzare ogni linguaggio riconosciuto da un automa di Büchi con una *espressione ω-regolare* che lo descrive come composizione di linguaggi regolari di parole finite.

Teorema 4.3 (Caratterizzazione dei linguaggi ω-regolari). *Un ω-linguaggio $L \subseteq A^\omega$ è riconosciuto da un automa di Büchi se e solo se L è un unione finita di insiemi $U_i \cdot V_i^\omega$, dove $U_i, V_i \subseteq A^*$ sono insiemi regolari di parole finite:*

$$L = \bigcup_{i=1}^{n} U_i \cdot V_i^\omega.$$

Il *teorema di caratterizzazione dei linguaggi ω-regolari* appena enunciato è la base di partenza per le caratterizzazioni dei linguaggi di parole fondamentalmente periodiche riconoscibili da automi di Büchi che verranno introdotte successivamente; inoltre, consente di stabilire direttamente due importanti risultati sul rapporto tra i linguaggi ω-regolari e le parole fondamentalmente periodiche in essi contenute. Indichiamo con $\mathrm{UP}(L)$ l'insieme delle parole fondamentalmente periodiche del linguaggio ω-regolare L. Valgono le seguenti proprietà.

Proposizione 4.4. (a) *Ogni linguaggio ω-regolare non vuoto contiene almeno una parola fondamentalmente periodica.*

(b) *L'insieme delle parole fondamentalmente periodiche di un linguaggio ω-regolare caratterizza il linguaggio: se $L_1, L_2 \subseteq A^\omega$ sono linguaggi ω-regolari, allora $\mathrm{UP}(L_1) = \mathrm{UP}(L_2)$ se e solo se $L_1 = L_2$.*

Dimostrazione. (a) Se $L \subseteq A^\omega$ è un linguaggio ω-regolare può essere caratterizzato (per il Teorema 4.3) da un'opportuna espressione ω-regolare:

$$L = \bigcup_{i=1}^{n} U_i \cdot V_i^\omega.$$

Poiché L è non vuoto, esiste almeno una parola $w \in L$ tale che $w = u \cdot v_1 \cdot v_2 \cdot \ldots$, con $u \in U_i$ e $v_1, v_2, \ldots \in V_i$ per un indice i opportuno.

Allora anche la parola fondamentalmente periodica $w' = u \cdot v_1^\omega = u \cdot v_1 \cdot v_1 \cdot \ldots$ appartiene a $U_i \cdot V_i^\omega$ e quindi w' è una parola fondamentalmente periodica che appartiene al linguaggio L.

(b) La dimostrazione segue direttamente dal punto precedente: siano $L_1, L_2 \subseteq A^\omega$ due insiemi ω-regolari che possiedono le stesse parole fondamentalmente periodiche. L'insieme $\Delta(L_1, L_2) = (L_1 \setminus L_2) \cup (L_2 \setminus L_1)$ è un insieme ω-regolare (per le proprietà di chiusura degli automi di Büchi) e non contiene parole fondamentalmente periodiche, quindi $\Delta(L_1, L_2) = \emptyset$ e $L_1 = L_2$. □

4.2 Linguaggi di parole fondamentalmente periodiche

In questa sezione forniamo una caratterizzazione dei linguaggi ω-regolari composti di sole parole fondamentalmente periodiche, al fine di identificare la corrispondente sottoclasse della classe degli automi di Büchi. Nella prima parte si

esaminano le proprietà di chiusura dei linguaggi di sole parole fondamentalmente periodiche rispetto alle operazioni già considerate per i linguaggi ω-regolari. Successivamente viene definita una relazione di equivalenza sui periodi delle parole fondamentalmente periodiche che consente di stabilire le condizioni in virtù delle quali un linguaggio ω-regolare contiene solamente parole fondamentalmente periodiche. Infine, viene introdotta una caratterizzazione generale dei linguaggi di sole parole fondamentalmente periodiche riconoscibili dagli automi di Büchi in termini dell'espressione ω-regolare che li definisce.

4.2.1 Proprietà di chiusura

Detta $\mathcal{F}$ la classe dei linguaggi ω-regolari composti da sole parole fondamentalmente periodiche, studiamo le proprietà di chiusura di $\mathcal{F}$ rispetto alle principali operazioni sui linguaggi. È importante notare come non tutti i linguaggi di parole fondamentalmente periodiche siano ω-regolari. Per esempio, il linguaggio F tale che

$$F = \{u \cdot v^\omega | u, v \in A^*\},$$

ossia tale che F è l'insieme di tutte le parole fondamentalmente periodiche sull'alfabeto A, non è un linguaggio ω-regolare, come dimostrato dalla Proposizione 4.7.

Proposizione 4.5 (Chiusura per concatenazione). *Sia $U \subseteq A^*$ un linguaggio regolare, e sia $V \in \mathcal{F}$ un linguaggio ω-regolare di sole parole fondamentalmente periodiche, allora il linguaggio*

$$U \cdot V = \{u \cdot v | u \in U, v \in V\}$$

è un linguaggio ω-regolare di parole fondamentalmente periodiche.

Dimostrazione. Dalla proprietà di chiusura per concatenazione dei linguaggi ω-regolari segue che $U \cdot V$ è un linguaggio ω-regolare, per dimostrare che è composto da sole parole fondamentalmente periodiche basta notare come ogni parola $v \in V$ possa essere scomposta in un prefisso finito ed un periodo che si ripete infinite volte:

$$v = w \cdot z^\omega$$

Il linguaggio $U \cdot V$ può quindi essere definito anche come

$$U \cdot V = \{u \cdot w \cdot z^\omega | u \in U, w \cdot z^\omega \in V\}$$

che è un linguaggio di sole parole fondamentalmente periodiche. □

Proposizione 4.6 (Chiusura per unione ed intersezione). *Siano $U, V \in \mathcal{F}$ due linguaggi ω-regolari di sole parole fondamentalmente periodiche, allora anche i linguaggi $U \cup V$ e $U \cap V$ sono ω-regolari e composti di sole parole fondamentalmente periodiche.*

Dimostrazione. Per le proprietà di chiusura per le operazioni booleane dei linguaggi ω-regolari gli insiemi $U \cup V$ e $U \cap V$ sono ω-regolari, e poiché sono composti di sole parole che appartengono a U o a V (o a entrambi) sono anche composti di sole parole fondamentalmente periodiche. □

I linguaggi ω-regolari di sole parole fondamentalmente periodiche non godono delle proprietà di chiusura per complementazione ed iterazione ω così come definite normalmente per i linguaggi ω-regolari: si può dimostrare come tali operazioni permettano di creare insiemi contenenti parole non fondamentalmente periodiche a partire da insiemi di sole parole fondamentalmente periodiche.

Per esempio, $L = \{a \cdot b^\omega\}$ è un linguaggio ω-regolare composto dalla sola parola fondamentalmente periodica $a \cdot b^\omega$, ed appartiene quindi a $\mathcal{F}$, mentre il suo complementare $\overline{L} = A^\omega \setminus \{a \cdot b^\omega\}$ è un linguaggio ω-regolare che contiene, in particolare, tutte le parole non fondamentalmente periodiche di A^ω, e quindi non può appartenere a $\mathcal{F}$. Nel caso della iterazione ω è sufficiente osservare che tale operazione permette di costruire l'insieme A^ω, che contiene tutte le parole infinite sull'alfabeto A, ma che non appartiene a $\mathcal{F}$ perché contiene anche parole non fondamentalmente periodiche.

Tali operazioni possono tuttavia essere ridefinite in modo da generare solamente insiemi di parole fondamentalmente periodiche nel modo seguente:

- la complementazione può essere definita rispetto all'insieme F di tutte e sole le parole fondamentalmente periodiche:
$$\overline{U} = F \setminus U;$$
- l'iterazione ω può essere definita rispettando la periodicità delle parole:
$$U^{inf} = \{u^\omega | u \in U\}$$

Le operazioni così ridefinite portano tuttavia alla generazione di insiemi di parole fondamentalmente periodiche che non sono ω-regolari.

Proposizione 4.7. *L'insieme F di tutte e sole le parole fondamentalmente periodiche non è ω-regolare.*

Dimostrazione. La dimostrazione procede per assurdo sfruttando il fatto che due insiemi ω-regolari con le stesse parole fondamentalmente periodiche sono identici.

Supponiamo per assurdo che F sia ω-regolare. Si può osservare come F e l'insieme ω-regolare A^ω contengano le stesse parole fondamentalmente periodiche, da cui segue, in virtù della Proposizione 4.4(b), $F = A^\omega$ (contraddizione). □

Corollario 4.8. *Gli insiemi ω-regolari di sole parole fondamentalmente periodiche non sono chiusi per le operazioni di complementazione e iterazione infinita ridefinite per rispettare la periodicità delle parole.*

Dimostrazione. Si procede per assurdo, dimostrando che in entrambi i casi si può arrivare alla costruzione dell'insieme F che non è ω-regolare.

Supponiamo per assurdo che gli insiemi ω-regolari siano chiusi per complementazione ed iterazione infinita.

Sia $U \in \mathcal{F}$ un insieme non vuoto, allora
$$\overline{U} \cup U = F$$
che non è un insieme ω-regolare (contraddizione).

L'insieme F può, inoltre, essere definito anche come
$$F = A^* \cdot (A^+)^{inf}$$
contraddicendo l'ipotesi che gli insiemi ω-regolari di sole parole fondamentalmente periodiche siano chiusi per iterazione infinita. □

4.2.2 Relazione di equivalenza sui periodi

Poiché i possibili periodi di una parola fondamentalmente periodica sono infiniti, risulta opportuno definire una relazione di equivalenza tra di essi e costruire l'insieme delle classi di equivalenza rispetto ad un ω-linguaggio dato. In questo modo sarà possibile caratterizzare gli insiemi ω-regolari un base alla cardinalità dell'insieme delle classi di equivalenza di tale relazione, e studiare la presenza (o l'assenza) di parole non fondamentalmente periodiche nel linguaggio.

Definiamo una relazione di equivalenza $\cong$ su A^* che associa alla stessa classe tutti e soli i possibili periodi di una parola fondamentalmente periodica.

Definizione 4.9 (Relazione di equivalenza sui periodi). Date due parole $u, v \in A^*$, la *relazione di equivalenza sui periodi* è la relazione $\cong$ tale che $u \cong v$ se e solo se le parole fondamentalmente periodiche u^ω e v^ω possiedono almeno un periodo in comune, cioè se esistono tre opportune parole finite w, x e y tali che $u^\omega = w \cdot x^\omega$ e $v^\omega = y \cdot x^\omega$.

Due periodi p_1 e p_2 di due parole fondamentalmente periodiche si dicono *distinti* se $p_1 \ncong p_2$.

Proposizione 4.10. *Tutti gli infiniti possibili periodi di una parola fondamentalmente periodica sono equivalenti rispetto alla relazione* $\cong$.

Dimostrazione. Sia w una parola fondamentalmente periodica e siano p_1 e p_2 due suoi periodi diversi. Per definizione, esistono due prefissi u e v tali che $w = u \cdot p_1^\omega = v \cdot p_2^\omega$.

Supponiamo, senza perdita di generalità, che $|u| \leq |v|$ e spezziamo v in due parti v_1 e v_2 tali che $|v_1| = |u|$ e $v = v_1 \cdot v_2$. La parola w può quindi essere riscritta come

$$w = u \cdot p_1^\omega = v_1 \cdot v_2 \cdot p_2^\omega.$$

Poiché $|u| = |v_1|$, si ha che $p_1^\omega = v_2 \cdot p_2^\omega$. Ne segue che p_2 è periodo sia di p_2^ω sia di p_1^ω, dimostrando che $p_1 \cong p_2$. □

Definizione 4.11 (Insieme dei periodi distinti). Dato un linguaggio ω-regolare L, l'*insieme dei periodi distinti* P *di* L è l'insieme delle classi di equivalenza di $\cong$ rispetto a tutti i possibili periodi delle parole fondamentalmente periodiche contenute in L.

Studiamo ora alcune proprietà dei periodi che appartengono alla stessa classe di equivalenza. Intuitivamente, se w è una parola fondamentalmente periodica, allora è possibile costruire i periodi di w attraverso:

1. la scelta del punto in cui inizia il periodo (il prefisso di w può infatti essere fissato con una qualsiasi lunghezza uguale o superiore a quella del prefisso minimo);
2. la scelta della lunghezza del periodo (se p è un periodo di w, allora anche $p^2, p^3, \ldots, p^n, \ldots$ sono periodi di w).

Questa osservazione porta a definire i seguenti due criteri per i quali due periodi della stessa classe di equivalenza possono differire.

Definizione 4.12 (Differenza per allineamento). Due periodi p_1 e p_2 si dicono *differenti per allineamento* se p_1^ω è un suffisso proprio di p_2^ω o viceversa (cioè se $p_1^\omega \neq p_2^\omega$ ed esiste $v \neq \varepsilon$ tale che $p_2^\omega = v \cdot p_1^\omega$ o viceversa).

Definizione 4.13 (Differenza per molteplicità). Due periodi p_1 e p_2 si dicono *differenti per molteplicità* se esistono due interi positivi e maggiori di zero n ed m tali che $p_1^n = p_2^m$.

È facile dimostrare che due periodi che differiscono per allineamento o per molteplicità sono equivalenti: il seguente teorema dimostra come valga anche il viceversa.

Teorema 4.14. *Due periodi p_1 e p_2 tali che $p_1 \cong p_2$ e $p_1 \neq p_2$ differiscono per allineamento o per molteplicità.*

Dimostrazione. Poiché $p_1 \cong p_2$ si ha che le parole p_1^ω e p_2^ω possiedono almeno un periodo comune x: siano u, v tali che $p_1^\omega = u \cdot x^\omega$ e $p_2^\omega = v \cdot x^\omega$. Dimostriamo ora che p_1 è un periodo per x^ω: sia k il minimo intero tale che $k \cdot |p_1| \geq |u|$, allora esiste una opportuna parola y tale che

$$p_1^k = u \cdot y \text{ e } x^\omega = y \cdot p_1^\omega;$$

quindi p_1 è periodo anche per p_2^ω:

$$p_2^\omega = v \cdot x^\omega = v \cdot y \cdot p_1^\omega = w_1 \cdot p_1^\omega, \text{ posto } w_1 = v \cdot y.$$

Si possono quindi presentare due casi:

- se $p_1^\omega \neq p_2^\omega$ allora p_1^ω è un suffisso proprio di p_2^ω, e quindi p_1 e p_2 differiscono per allineamento,
- se $p_1^\omega = p_2^\omega$ esistono due interi positivi n, m tali che $n \cdot |p_1| = m \cdot |p_2|$ e $p_1^n = p_2^m$, quindi p_1 e p_2 differiscono per molteplicità.

□

Inoltre dalla Definizione 4.13 di differenza per molteplicità segue direttamente, per il Lemma di Fine-Wilf (Lemma 2.10), che se due periodi differiscono per molteplicità allora sono nella forma $p_1 = p^m$ e $p_2 = p^n$, cioè possono essere riscritti come sequenze di un unico sottoperiodo comune.

Proposizione 4.15. *Se due periodi p_1 e p_2 differiscono solo per molteplicità esiste un periodo $p \in A^*$ e due interi positivi $n, m \in \mathbb{N}$ tali che $p_1 = p^n$ e $p_2 = p^m$.*

Dimostrazione. Poiché p_1 e p_2 differiscono solo per molteplicità esistono due interi $k, l \in \mathbb{N}$ tali che $p_1^k = p_2^l = v$ e $|v| \geq |p_1| + |p_2| - M.C.D.(|p_1|, |p_2|)$.

Per il lemma di Fine-Wilf v ha un periodo p di lunghezza $M.C.D.(|p_1|, |p_2|)$. Inoltre p è anche un periodo di p_1 e p_2, cioè esistono $n, m \in \mathbb{N}^+$ tali che $p_1 = p^n$ e $p_2 = p^m$. □

4.2.3 Periodi dei linguaggi V^ω

La particolare struttura dei linguaggi ω-regolari (che sono rappresentabili come unioni finite di insiemi del tipo $U \cdot V^\omega$) fa sì che ci si possa limitare, nello studio dei periodi delle parole fondamentalmente periodiche contenute in un linguaggio ω-regolare, ai soli insiemi del tipo V^ω.

In questa sezione si approfondisce la struttura dei linguaggi ω-regolari del tipo V^ω rispetto alle parole fondamentalmente periodiche che essi generano, stabilendo sotto quali condizioni il linguaggio è composto di sole parole fondamentalmente periodiche. I risultati ottenuti saranno poi generalizzati a linguaggi ω-regolari qualsiasi.

Nella prima parte vengono introdotte alcune proprietà dei periodi presenti nei linguaggi V^ω ed un teorema generale che stabilisce le condizioni sufficienti affinché il linguaggio contenga almeno una parola non fondamentalmente periodica.

Vediamo ora due lemmi che permettono di caratterizzare i periodi contenuti in V^ω rispetto alle parole contenute nell'insieme V che genera il linguaggio.

Lemma 4.16. *Sia $L = V^\omega$ un linguaggio ω-regolare che contiene la parola fondamentalmente periodica $u \cdot v^\omega$. Allora L contiene anche una opportuna parola fondamentalmente periodica w^ω, con w che differisce da v per allineamento.*

Dimostrazione. Costruiamo l'automa $\mathcal{A}$ che riconosce il linguaggio L a partire dall'automa regolare che riconosce il linguaggio V, secondo il metodo utilizzato nella dimostrazione del Lemma 4.2, che prevede il solo stato iniziale q_0 come finale.

Sia ρ un run accettante di $\mathcal{A}$ per $u \cdot v^\omega$: esso è nella forma

$$\rho = q_0 \overset{u(1)}{\to} q_1 \overset{u(2)}{\to} \dots \overset{u(|u|)}{\to} q_{|u|} \overset{v(1)}{\to} q_{|u|+1} \overset{v(2)}{\to} \dots$$

Consideriamo ora il suffisso $\sigma = \rho[|u|, \infty[$ di ρ. Poiché ρ è accettante esistono infinite occorrenze di q_0. Sia j la posizione della prima occorrenza di q_0 in σ: il suffisso infinito $\sigma[j, \infty[$ di σ è un run accettante per l'automa $\mathcal{A}$ perché inizia nello stato q_0 e contiene infinite occorrenze dello stato finale q_0. Inoltre esiste $1 \leq k \leq |v|$ tale che la struttura di $\sigma[j, \infty[$ è la seguente:

$$\sigma[j, \infty[= p_j \overset{v(k)}{\to} p_{j+1} \overset{v(k+1)}{\to} \dots \overset{v(|v|)}{\to} p_{j+|v|+1} \overset{v(1)}{\to} \dots \overset{v(k-1)}{\to} p_{j+|v|+k} \overset{v(k)}{\to} \dots$$

Quindi il run $\sigma[j, \infty[$ riconosce la parola fondamentalmente periodica

$$w^\omega = (v[k, |v|] \cdot v[1, k-1])^\omega$$

Il periodo cercato che differisce da v solo per allineamento è quindi $w = v[k, |v|] \cdot v[1, k-1]$. □

Corollario 4.17. *Sia $L = V^\omega$ e sia P l'insieme dei periodi distinti di L: per ogni classe di equivalenza di P esiste almeno un testimone* allineato*, cioè esiste almeno un testimone w tale che $w^\omega \in L$.*

Lemma 4.18. *Sia $L = V^\omega$ e sia $P_A = \{p_1, p_2, \dots\}$ l'insieme dei periodi allineati presenti in L. Allora per ogni $p \in P_A$ esistono $v_1, v_2, \dots v_k \in V$ tali che $p \cong v_1 \cdot v_2 \cdot \dots \cdot v_k$.*

Dimostrazione. Costruiamo l'automa $\mathcal{A}$ che riconosce il linguaggio L a partire dall'automa regolare che riconosce il linguaggio V, secondo il metodo utilizzato nella dimostrazione del Lemma 4.2, che prevede il solo stato iniziale q_0 come finale. La struttura dell'automa è tale che tra due occorrenze successive di q_0 in ogni run di $\mathcal{A}$ viene riconosciuta una parola $v \in V$, poiché si assume che nell'automa che riconosce V non vi siano transizioni entranti in q_0.

Sia $p \in P_A$. Poiché p è allineato, sia $\rho = q_0 q_1 q_2 \ldots$ un run accettante per p^ω. Consideriamo ora gli stati di ρ in cui l'automa legge il carattere iniziale di p: essi formano la sequenza infinita

$$\rho_{|p|} = q_0 q_{|p|} q_{2|p|} \cdots q_{n|p|} \cdots$$

Sia q uno stato che si ripete infinite volte in $\rho_{|p|}$, e siano i e j gli indici di due occorrenze distinte di q tali che

1. $q_{i|p|}$ è la prima occorrenza di q in $\rho_{|p|}$;
2. $q_{j|p|}$ è la prima occorrenza successiva a i tale che esiste almeno un'occorrenza dello stato finale q_0 nel segmento $\rho[i|p|, j|p|]$ del run accettante ρ.

Consideriamo ora il segmento $\rho[i|p|, j|p|]$ del run accettante ρ: esso riconosce il periodo p^{j-i} di p^ω, che differisce da p per molteplicità.

Sia l l'indice della prima occorrenza di q_0 in $\rho[i|p|, j|p|]$: costruiamo per rotazione un nuovo segmento di run σ che inizi e termini in q_0:

$$\sigma = \rho[l, j|p|] \cdot \rho[i|p| + 1, l].$$

Il segmento σ riconosce un periodo di p^ω che differisce per allineamento da p^{j-i} ed inoltre, poiché σ inizia e termina con q_0, esistono k parole $v_1, v_2, \ldots, v_k \in V$ (una per ogni occorrenza di q_0 in σ) tali che σ riconosce il periodo $v_1 \cdot v_2 \cdot \ldots \cdot v_k$.

Quindi $v_1 \cdot v_2 \cdot \ldots \cdot v_k \cong p$ è il periodo cercato. □

Corollario 4.19. *Sia $L = V^\omega$ e sia P l'insieme dei periodi distinti di L: per ogni classe di equivalenza di P esiste almeno un testimone* che rispetti V, *cioè esiste almeno un testimone w tale che $w = v_1 \cdot v_2 \cdot \ldots \cdot v_k$, con $v_1, v_2, \ldots, v_k \in V$.*

I due lemmi appena introdotti, ed i relativi corollari, consentono di dimostrare il seguente teorema, che stabilisce le condizioni sufficienti per la presenza di parole non fondamentalmente periodiche nei linguaggi del tipo V^ω.

Teorema 4.20. *Ogni linguaggio ω-regolare nella forma $L = V^\omega$ che contiene almeno due periodi distinti contiene almeno una parola non fondamentalmente periodica.*

Dimostrazione. Sia $L = V^\omega$ e sia $P_V = \{p_1, p_2, \ldots\}$ un insieme contenente per ogni classe di equivalenza di $\cong$ un testimone che rispetti V. Per le ipotesi del teorema, P_V contiene almeno due elementi distinti p_1 e p_2.

Costruiamo una parola non fondamentalmente periodica che appartenga a L alternando sequenze di p_1 e p_2 di lunghezza crescente:

$$w = p_1 \cdot p_2 \cdot (p_1)^2 \cdot (p_2)^2 \cdot \ldots \cdot (p_1)^k \cdot (p_2)^k \cdot (p_1)^{k+1} \cdot (p_2)^{k+1} \ldots$$

Poiché p_1 e p_2 rispettano V, la parola w appartiene ad L: supponiamo per assurdo che sia fondamentalmente periodica, e che possa quindi essere scritta nella forma $w = u \cdot v^\omega$.

Siano n, m due interi positivi tali che $n|p_1| = m|v|$. Per la particolare costruzione di w è possibile trovare un punto in cui m occorrenze successive di v sono contenute in una sequenza di soli periodi p_1, quindi v differisce da p_1 per molteplicità o allineamento e $v \cong p_1$.

Analogamente, siano i e j due interi positivi tali che $i|p_2| = j|v|$. Per la particolare costruzione di w è possibile trovare un punto in cui j occorrenze successive di v sono contenute in una sequenza di soli periodi p_2, quindi v differisce da p_2 per molteplicità o allineamento e $v \cong p_2$.

Questo porta all'assurdo $p_1 \cong p_2$, contro la definizione dell'insieme P_V. Quindi w è una parola non fondamentalmente periodica che appartiene a L. □

La dimostrazione del teorema evidenzia una proprietà interessante dei periodi che rispettano l'insieme V: possono essere combinati insieme per ottenere nuove parole appartenenti all'insieme ma con periodo diverso da quelli di partenza. Questa osservazione suggerisce il fatto che un insieme con due periodi distinti contenga in realtà un numero infinito di periodi distinti ottenuti per combinazione.

Lemma 4.21. *Ogni linguaggio ω-regolare nella forma $L = V^\omega$ che contiene almeno due periodi distinti contiene infiniti periodi distinti.*

Dimostrazione. Sia $L = V^\omega$ e sia $P_V = \{p_1, p_2, \ldots\}$ un insieme contenente per ogni classe di equivalenza di $\cong$ un testimone che rispetti V. Per le ipotesi del teorema, P_V contiene almeno due elementi distinti p_1 e p_2.

Costruiamo quindi per combinazione di p_1 e p_2 una sequenza infinita di periodi sempre più lunghi:

$$\begin{gathered} p_1 \cdot p_2 \\ p_1 \cdot p_1 \cdot p_2 \cdot p_2 \\ (p_1)^3 \cdot (p_2)^3 \\ \ldots \\ (p_1)^n \cdot (p_2)^n \\ \ldots \end{gathered}$$

Poiché ogni elemento della sequenza è distinto dagli altri, l'insieme L contiene infiniti periodi distinti. □

Stabilite le condizioni per le quali un linguaggio del tipo V^ω contiene parole non fondamentalmente periodiche, non resta che descrivere le caratteristiche che deve avere per contenere solamente parole fondamentalmente periodiche. Visti i risultati ottenuti è utile sviluppare la struttura degli insiemi del tipo V^ω che contengono un unico periodo distinto.

Teorema 4.22. *Ogni linguaggio ω-regolare nella forma $L = V^\omega$ che contiene un solo periodo distinto contiene una sola parola infinita nella forma v^ω, con v opportuno.*

Dimostrazione. Per le ipotesi del teorema l'insieme $V = \{v_1, v_2, \ldots\}$ che genera il linguaggio L è tale che per ogni $v_i, v_j \in V$, $v_i \cong v_j$, quindi ogni parola di V differisce dalle altre solamente per molteplicità o per allineamento.

Dimostriamo che due parole appartenenti a V non possono differire per allineamento in quanto genererebbero un nuovo periodo distinto di L: siano per assurdo $v_1, v_2 \in V$ distinti per allineamento. Supponiamo, senza perdita di generalità, che v_1^ω sia un suffisso proprio di v_2^ω e sia u la minima parola tale che

$$v_2^\omega = u \cdot v_1^\omega.$$

Allora esistono due interi positivi n, m tali che $n \cdot |v_1| = m \cdot |v_2| > |u|$: siano quindi $w_1 = v_1^n$ e $w_2 = v_2^m$. Chiaramente, anche w_1 e w_2 differiscono per allineamento e

$$w_2^\omega = u \cdot w_1^\omega,$$

da ciò segue che w_2 può essere ottenuta a partire da w_1 per rotazione, infatti

$$w_2 \cdot u = u \cdot w_1$$

ed esiste quindi una opportuna parola x tale che

$$w_2 = u \cdot x \text{ e } w_1 = x \cdot u.$$

Consideriamo ora la parola fondamentalmente periodica $(w_1 \cdot w_2)^\omega = (x \cdot u \cdot u \cdot x)^\omega$: essa appartiene ad L ma il suo periodo $x \cdot u \cdot u \cdot x$ è distinto da w_1 e w_2, contro l'ipotesi che L contenga un solo periodo distinto.

Quindi le parole di V differiscono per molteplicità: per la Proposizione 4.15 esiste un periodo v di lunghezza $|v| = M.C.D.(|v_1|, |v_2|, \ldots)$ tale che per ogni $v_i \in V$ esiste $k_i \in \mathbb{N}$ tale che $v_i = v^{k_i}$. Quindi ogni possibile sequenza infinita di parole di V altro non è che la ripetizione infinita del prefisso minimale v, e la sola parola che appartiene a L è v^ω. □

Questo teorema, assieme agli altri risultati raggiunti, stabilisce che gli insiemi del tipo V^ω che contengono solamente parole fondamentalmente periodiche sono tutti e soli quelli che contengono un unico periodo distinto, cioè quelli formati da un'unica parola infinita.

4.2.4 Caratterizzazione generale

Abbiamo già osservato che i linguaggi ω-regolari possono essere definiti come unioni di insiemi del tipo $U_i \cdot V_i^\omega$, e quindi che i responsabili della generazione dei periodi delle parole fondamentalmente periodiche sono solamente le componenti del tipo V_i^ω.

I teoremi dimostrati nella sezione precedente sulla struttura degli insiemi V^ω possono quindi essere estesi ai linguaggi ω-regolari qualsiasi, fornendo una caratterizzazione ai linguaggi ω-regolari di sole parole fondamentalmente periodiche rispetto all'espressione ω-regolare che li definisce.

Teorema 4.23. *Ogni insieme ω-regolare che contiene infiniti periodi distinti contiene almeno una parola non fondamentalmente periodica.*

Dimostrazione. Sia $L = \bigcup_{i=1}^{n} U_i \cdot V_i^\omega$ un linguaggio ω regolare tale che la relazione di equivalenza sui periodi $\cong$ è di ordine infinito.

Esiste quindi una componente $U_i \cdot V_i^\omega$ che contiene anch'essa infiniti periodi distinti: poiché i periodi delle parole fondamentalmente periodiche vengono generati dalla sola componente V_i^ω, anche V_i^ω contiene infiniti periodi distinti.

Quindi V_i^ω contiene almeno una parola non fondamentalmente periodica, e di conseguenza anche L contiene almeno una parola non fondamentalmente periodica. □

Il Teorema 4.22, che caratterizza i linguaggi V^ω con un solo periodo distinto, ci permette inoltre di stabilire le condizioni necessarie e sufficienti perché un insieme ω-regolare contenga solamente parole fondamentalmente periodiche.

Teorema 4.24 (Teorema di caratterizzazione dei linguaggi ω-regolari di sole parole fondamentalmente periodiche). *Gli insiemi ω-regolari che contengono solamente parole fondamentalmente periodiche sono tutti e soli quelli che contengono un numero finito di periodi distinti, cioè quelli che possono essere scritti nella forma $\bigcup_{i=1}^{n} U_i \cdot V_i^\omega$, con V_i insieme singoletto.*

Dimostrazione. È facile osservare come un insieme ω-regolare di sole parole fondamentalmente periodiche contenga un numero finito di periodi distinti: la presenza di infiniti periodi distinti implicherebbe la presenza di almeno una parola non fondamentalmente periodica.

Dimostriamo invece come un insieme ω-regolare che contiene un numero finito di periodi distinti contenga solo parole fondamentalmente periodiche attraverso la struttura dell'espressione ω-regolare che lo definisce: essa è nella forma

$$L = \bigcup_{i=1}^{n} U_i \cdot V_i^\omega.$$

Il Lemma 4.21 dà due sole possibilità per la cardinalità dell'insieme dei periodi distinti delle componenti V_i^ω: essa può essere uno o infinita. Poiché L contiene un numero finito di periodi distinti ogni V_i^ω contiene un solo periodo distinto, ed è quindi nella forma $V_i^\omega = \{v_i^\omega\}$, con v_i opportuno.

Ogni componente $U_i \cdot V_i^\omega$ di L può quindi essere scritta nella forma

$$U_i \cdot V_i^\omega = \{u \cdot v_i^\omega | u \in U_i\}$$

ed è quindi un insieme che contiene solamente parole fondamentalmente periodiche aventi prefisso appartenente a U_i ed un unico periodo possibile v_i.

L'insieme L è quindi un'unione finita di insiemi di sole parole fondamentalmente periodiche, quindi è un insieme di sole parole fondamentalmente periodiche. □

Osservazione 4.25. Il Teorema 4.24 appena dimostrato permette di osservare come gli automi di Büchi riescano a catturare solamente una piccola classe di insiemi di parole fondamentalmente periodiche, e più esattamente gli insiemi di parole fondamentalmente periodiche aventi un *numero finito* di periodi distinti ed un *numero infinito* di prefissi. Questi linguaggi consentono tuttavia di rappresentare una classe significativa di granularità, le cosiddette *granularità dinamiche*, cioè quelle granularità il cui inizio può essere spostato a piacere lungo la scala temporale adottata. Gli insiemi ω-regolari di parole fondamentalmente periodiche consentono infatti di rappresentare parole aventi un prefisso di lunghezza arbitraria, che stabilisce l'istante iniziale della granularità, ed un periodo scelto tra un insieme finito di possibilità. In questo modo si possono quindi rappresentare *insiemi finiti di granularità dinamiche*.

4.3 Definizione della classe di automi

Nella sezione precedente è stata data una caratterizzazione generale ai linguaggi ω-regolari di sole parole fondamentalmente periodiche; in questa sezione si procede con la costruzione effettiva degli automi di Büchi che riconoscono tali linguaggi a partire da componenti più semplici. Verrà quindi definita una sottoclasse degli automi ω-regolari che riconosce solamente parole fondamentalmente

periodiche e che gode delle stesse proprietà di chiusura dei linguaggi ω-regolari di sole parole fondamentalmente periodiche descritte nella Sezione 4.2.1. Tale sottoclasse può essere impiegata per rappresentare insiemi di parole fondamentalmente periodiche, come insiemi di granularità, e per eseguire operazioni su di essi.

4.3.1 Operazioni di base per la costruzione degli automi

Il teorema di caratterizzazione dei linguaggi ω-regolari di sole parole fondamentalmente periodiche (Teorema 4.24) descrive tali linguaggi come generati da espressioni ω-regolari nella forma

$$L = \bigcup_{i=1}^{n} U_i \cdot \{v_i^\omega\}$$

e stabilisce quindi implicitamente le operazioni di base che consentono la costruzione effettiva degli automi che li riconoscono. Tali operazioni sono:

1. la costruzione dell'automa che riconosce la parola infinita v_i^ω a partire da v_i;
2. la concatenazione con il linguaggio regolare U_i;
3. l'unione finita di linguaggi di sole parole fondamentalmente periodiche.

Le operazioni elencate sono effettive rispetto agli automi di Büchi e consentono quindi di costruire l'automa che riconosce un linguaggio di sole parole fondamentalmente periodiche a partire dalla espressione ω-regolare che lo definisce. Le tre proposizioni seguenti descrivono come costruire effettivamente tale automa.

Proposizione 4.26. *A partire dalla parola finita $v \in A^*$ è possibile costruire l'automa di Büchi $\mathcal{A}_v$ che riconosce solamente la parola infinita v^ω.*

Dimostrazione. Sia $v = a_0 a_1 a_2 \dots a_n$ una parola finita di lunghezza n. Allora l'automa di Büchi $\mathcal{A}_v = (Q, \Delta, q_0, F)$ che riconosce solamente la parola v^ω è definito come segue:

- $Q = \{q_0, q_1, \dots, q_n\}$;
- $\Delta = \{(q_i, a_i, q_{i+1}) | 0 \leq i \leq n-1\} \cup \{(q_n, a_n, q_0\}$;
- $F = \{q_0\}$.

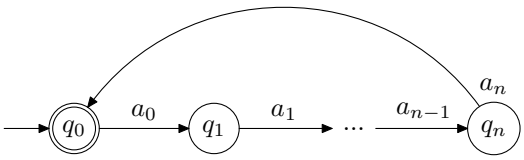

Figura 4.1: Automa $\mathcal{A}_v$ che riconosce la parola v^ω

Graficamente l'automa può essere rappresentato come in Figura 4.1: esso è strutturato in modo da possedere uno stato per ogni lettera della parola v e riconoscere in ogni stato q_i solamente la lettera corrispondente a_i. Dallo stato q_n l'automa ritorna poi nello stato iniziale per riconoscere una nuova occorrenza di v.

L'unico run possibile per l'automa è quindi $\rho = q_0q_1q_2 \dots q_{n-1}q_nq_0 \dots$ a cui corrisponde la parola infinita $v^\omega = a_0a_1a_2 \dots a_{n-1}a_na_0 \dots$. L'automa $\mathcal{A}_v$ riconosce quindi il linguaggio $L(\mathcal{A}_v) = \{v^\omega\}$. □

Proposizione 4.27. *Sia L un linguaggio ω-regolare di sole parole fondamentalmente periodiche riconosciuto dall'automa di Büchi $\mathcal{A}_L$ e sia U un linguaggio regolare riconosciuto dall'automa regolare $\mathcal{A}_U$. Allora è possibile costruire l'automa di Büchi $\mathcal{A}_{U \cdot L}$ che riconosce il linguaggio ω-regolare di sole parole fondamentalmente periodiche $U \cdot L$.*

Dimostrazione. La costruzione dell'automa avviene come nel caso generale della concatenazione negli automi di Büchi (cfr. Lemma 4.2). Se $\mathcal{A}_L = (Q_L, \Delta_L, q_0^L, F_L)$ e $\mathcal{A}_U = (Q_U, \Delta_U, q_0^U, F_U)$ l'automa $\mathcal{A}_{U \cdot L} = (Q, \Delta, q_0, F)$ è così definito:

- $Q = Q_U \cup Q_L$, nell'ipotesi che gli insiemi di stati Q_U e Q_L siano disgiunti;
- $\Delta = \Delta_U \cup \Delta_L \cup \{(q, a, q') | q \in F_U \text{ e } (q_0^L, a, q') \in \Delta_L\}$, cioè le transizioni sono date dall'unione dei due insiemi di transizioni più un insieme di transizioni "di passaggio" dagli stati finali del primo automa agli stati del secondo automa;
- $q_0 = q_0^U$;
- $F = F_L$.

□

Proposizione 4.28. *Siano L_1 e L_2 due linguaggi ω-regolari di sole parole fondamentalmente periodiche riconosciuti dagli automi di Büchi $\mathcal{A}_1$ e $\mathcal{A}_2$; allora è possibile costruire l'automa di Büchi $\mathcal{A}_3$ che riconosce il linguaggio $L_3 = L_1 \cup L_2$.*

Dimostrazione. Se $\mathcal{A}_1 = (Q_1, \Delta_1, q_0^1, F_1)$ e $\mathcal{A}_2 = (Q_2, \Delta_2, q_0^2, F_2)$, l'automa $\mathcal{A}_3 = (Q, \Delta, q_0, F)$ viene costruito come automa unione dei due linguaggi come nel caso generale (cfr. Lemma 4.2):

- l'insieme degli stati Q è definito come unione dei due insiemi di stati Q_1 e Q_2 più un nuovo stato iniziale q_0;
- la relazione di transizione è ottenuta aggiungendo all'unione delle due relazioni Δ_1 e Δ_2 l'insieme di transizioni iniziali $\{(q_0, a, q) | (q_0^1, a, q) \in \Delta_1 \text{ o } (q_0^2, a, q) \in \Delta_2\}$;
- l'insieme degli stati finali F è ottenuto come unione dei due insiemi di stati finali F_1 e F_2.

□

4.3.2 Automi di Büchi fondamentalmente periodici

Le operazioni di base viste in precedenza consentono di costruire l'automa che riconosce un linguaggio di sole parole fondamentalmente periodiche a partire dalla espressione ω-regolare che lo definisce. I metodi impiegati per costruire effettivamente l'automa suggeriscono inoltre una possibile caratterizzazione per una classe di automi di Büchi che riconoscano solamente linguaggi di parole fondamentalmente periodiche.

Si può osservare come gli automi costruiti come descritto nelle dimostrazioni delle Proposizioni 4.26, 4.27 e 4.28 possiedano le seguenti caratteristiche particolari:

1. quando l'automa entra nella parte che riconosce il suffisso infinito v^ω di una parola fondamentalmente periodica si comporta come un automa single-string: esiste un unico run possibile ed esso riconosce esattamente v^ω;

2. gli stati finali dell'automa si trovano solamente nelle parti che riconoscono i suffissi infiniti del tipo v^ω.

Possiamo quindi definire una classe di automi di Büchi, detta classe degli *automi di Büchi fondamentalmente periodici* (o più semplicemente *automi fondamentalmente periodici*), che generalizzi queste due caratteristiche peculiari imponendo che dall'istante in cui un automa entra in uno stato finale esso si comporti come un automa single-string che riconosce la parola infinita v^ω, con v opportuno. Ogni stato finale dell'automa deve quindi appartenere ad un solo ciclo che non possiede transizioni uscenti e che riconosce solamente il periodo v. Questo tipo di automi sono detti fondamentalmente periodici perché ogni loro run accettante è fondamentalmente periodico rispetto agli stati dell'automa.

Definizione 4.29 (Automa di Büchi fondamentalmente periodico). Un automa di Büchi $\mathcal{A} = (Q, \Delta, q_0, F)$ è detto *fondamentalmente periodico* se ogni stato finale appartiene ad un solo ciclo senza transizioni uscenti che riconosce un unico periodo v opportuno, cioè se per ogni $f \in F$:

1. esiste un ciclo che ritorna in f: $f \rightarrow^* f$,
2. lo stato f ed ogni stato q raggiungibile da f possiedono esattamente una transizione uscente: per ogni $q \in Q$ tale che $f \rightarrow^* q$ esiste un'unica coppia $(a, q') \in A \times Q$ tale che $(q, a, q') \in \Delta$.

Un automa (di Büchi) fondamentalmente periodico può essere rappresentato da un grafo orientato, secondo le seguenti convenzioni:

- i nodi sono etichettati con il nome dello stato che rappresentano;
- lo stato iniziale è identificato da una freccia;
- gli stati iniziali sono identificati da una doppia cerchiatura;
- le transizioni sono rappresentate da archi etichettati con simboli dell'alfabeto A.

Esempio 4.30. La Figura 4.2 rappresenta due automi fondamentalmente periodici. Il primo automa riconosce il linguaggio $\{\wr^n \cdot (\blacksquare\wr)^\omega | n \in \mathbb{N}\}$, corrispondente, nel formalismo della Calendar Algebra, all'insieme di granularità $\{G =$

$Shift_n(Giorni)|n \in \mathbb{N}\}$ di tutte le trasposizioni (con indice zero o positivo) della granularità *Giorni*. Il secondo automa riconosce il linguaggio $\{(\square\blacksquare\wr)^\omega, (\blacksquare\blacksquare\wr)^\omega\}$, che possiede due periodi distinti per le parole fondamentalmente periodiche contenute.

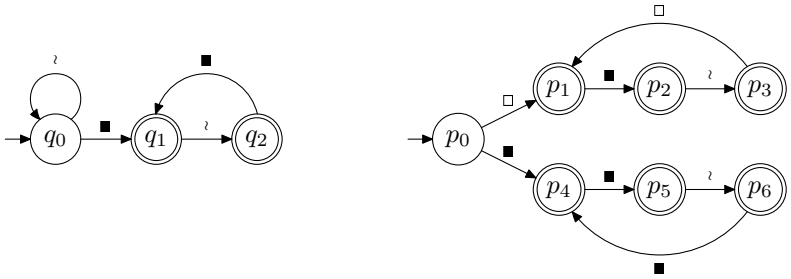

Figura 4.2: Esempi di automi fondamentalmente periodici

Dalla definizione segue immediatamente che gli automi costruiti per rappresentare insiemi di sole parole fondamentalmente periodiche secondo i metodi presentati nelle Proposizioni 4.26, 4.27 e 4.28 sono fondamentalmente periodici e quindi che per ogni linguaggio ω-regolare di sole parole fondamentalmente periodiche esiste un automa fondamentalmente periodico che lo riconosce. Dimostriamo inoltre come gli automi fondamentalmente periodici riconoscano solamente parole fondamentalmente periodiche.

Teorema 4.31. *Gli automi di Büchi fondamentalmente periodici riconoscono solamente linguaggi di parole fondamentalmente periodiche.*

Dimostrazione. Dimostriamo il teorema osservando che per ogni stato finale f di un automa fondamentalmente periodico esiste un solo run infinito possibile, e tale run riconosce esattamente la parola v_f^ω, con v_f periodo riconosciuto dall'unico ciclo dell'automa che inizia e termina in f.

Sia $\mathcal{A} = (Q, \Delta, q_0, F)$ un automa di Büchi fondamentalmente periodico e sia $f \in F$ uno stato finale. Dalla definizione di automa fondamentalmente periodico segue che esiste una sola transizione uscente da f e da ogni stato raggiungibile da f, quindi esiste un solo run infinito possibile ρ che inizi in f ed esso riconosce solamente la parola infinita w:

$$\rho = \rho(1) \xrightarrow{w(1)} \rho(2) \xrightarrow{w(2)} \ldots \xrightarrow{w(n-1)} \rho(n) \xrightarrow{w(n)} \ldots$$

Infatti ρ è tale che $\rho(1) = f$ ed esiste una sola transizione $(\rho(n), w(n), \rho(n+1) \in \Delta$ per ogni $n \geq 1$.

Poiché $f \rightarrow^* f$ si ha che esiste un indice $k > 1$ tale che $\rho(k) = f$ e per ogni $1 < i < k$, $\rho(i) \neq f$, quindi

$$\rho = (\rho(1) \cdot \ldots \cdot \rho(k-1))^\omega.$$

Di conseguenza esiste un unico ciclo $C_f = \rho(1) \cdot \rho(2) \cdot \ldots \cdot \rho(k-1)$ che inizia e termina in f, tale ciclo riconosce solamente un periodo v_f opportuno e l'unica parola infinita riconoscibile dall'automa a partire da f è v_f^ω. Ogni

run accettante dell'automa è quindi formato da un prefisso di stati non finali e da un suffisso C_f^ω corrispondente al ciclo finale del primo stato finale raggiunto e quindi può riconoscere solamente parole fondamentalmente periodiche aventi periodo v_f. L'automa $\mathcal{A}$ riconosce quindi linguaggi composti unicamente da parole fondamentalmente periodiche. □

4.3.3 Automi fondamentalmente periodici in forma normale

La struttura particolare degli automi fondamentalmente periodici, in cui ogni stato finale appartiene ad un solo ciclo da cui l'automa non può uscire e che riconosce solamente un periodo, fa si che ogni run accettante dell'automa sia composto da un prefisso di stati non finali seguito da un ciclo finale (cioè un ciclo che contiene almeno uno stato finale) che si ripete infinite volte. Nel caso degli automi fondamentalmente periodici, ciò che consente di riconoscere una parola è in realtà il *ciclo finale* che il run raggiunge, piuttosto gli stati finali considerati singolarmente.

Questa osservazione permette di definire una *forma normale* per gli automi fondamentalmente periodici che consente di semplificare la struttura degli automi e di effettuare operazioni su di essi in maniera più semplice, senza tuttavia diminuirne l'espressività. La seguente definizione introduce quindi la nozione di *automa di Büchi fondamentalmente periodico in forma normale.*

Definizione 4.32. Un automa di Büchi fondamentalmente periodico $\mathcal{A} = (Q, \Delta, q_0, F)$ è *in forma normale* se valgono le seguenti condizioni:

- tutti gli stati raggiungibili da uno stato finale sono finali (cioè $\forall f \in F, q \in Q. f \rightarrow^* q \Rightarrow q \in F$).
- lo stato iniziale q_0 non è uno stato finale (cioè $q_0 \notin F$),

Il primo punto della Definizione 4.32 stabilisce che tutti gli stati che compongono un ciclo finale sono finali e consente quindi di distinguere facilmente le parti di automa che riconoscono i periodi delle parole riconosciute dalle parti di automa che riconoscono i prefissi. Il secondo punto impone invece che l'automa riconosca almeno un carattere prima di entrare in un ciclo finale e risulterà utile nelle procedure di minimizzazione degli automi fondamentalmente periodici che presenteremo nel capitolo successivo.

Osservazione 4.33. La struttura di un automa fondamentalmente periodico in forma normale è tale che ogni run accettante dell'automa è composto da un prefisso finito composto di soli stati non finali e da un suffisso infinito composto di soli stati finali (che sono esattamente gli stati del ciclo finale raggiunto dall'automa).

Il seguente teorema dimostra che gli automi fondamentalmente periodici in forma normale riconoscono esattamente gli stessi linguaggi degli automi fondamentalmente periodici generali. Da questo punto in poi (salvo dove diversamente specificato) si assume che ogni automa fondamentalmente periodico citato sia in forma normale.

Teorema 4.34. *Sia $\mathcal{A} = (Q, \Delta, q_0, F)$ un automa di Büchi fondamentalmente periodico non in forma normale. Allora esiste un automa di Büchi fondamentalmente periodico in forma normale che riconosce lo stesso linguaggio di $\mathcal{A}$.*

Dimostrazione. Fissato un automa di Büchi fondamentalmente periodico non in forma normale $\mathcal{A} = (Q, \Delta, q_0, F)$, si costruisce l'automa di Büchi $\mathcal{A}' = (Q', \Delta', q_0', F')$ in forma normale definito come segue:

- se $\nexists f \in F. f \rightarrow^* q_0$, $\mathcal{A}'$ è tale che:
  - $Q' = Q$,
  - $\Delta' = \Delta$,
  - $q_0' = q_0$,
  - $F' = F \cup \{q \in Q | \exists f \in F. f \rightarrow^* q\}$.
- se $\exists f \in F. f \rightarrow^* q_0$ è tale che:
  - $Q' = Q \cup \{q_0'\}$, con $q_0' \notin Q$,
  - $\Delta' = \Delta \cup \{(q_0', a, q) | (q_0, a, q) \in \Delta\}$,
  - $F' = F \cup \{q \in Q | \exists f \in F. f \rightarrow^* q\}$.

Intuitivamente $\mathcal{A}'$ è ottenuto da $\mathcal{A}$ aggiungendo un nuovo stato iniziale se lo stato iniziale di $\mathcal{A}$ è raggiungibile a partire da uno stato finale, e marcando come finali tutti gli stati che compongono un ciclo finale (cioè tutti gli stati raggiungibili da uno stato finale). Si vede immediatamente che $\mathcal{A}'$ rispetta la definizione di automa fondamentalmente periodico in forma normale. Dimostriamo che riconosce lo stesso linguaggio di $\mathcal{A}$.

Sia quindi $w \in L(\mathcal{A})$ e sia ρ un run accettante di $\mathcal{A}$ su w. Se $q_0' = q_0$ allora ρ è anche un run di $\mathcal{A}'$ su w ed è accettante perché ogni stato finale di $\mathcal{A}$ è stato finale anche di $\mathcal{A}'$. Se, invece, $q_0' \neq q_0$ allora possiamo costruire un run ρ' di $\mathcal{A}'$ su w semplicemente sostituendo la transizione iniziale (q_0, a, q) di ρ con la transizione (q_0', a, q) che appartiene a Δ'; anche in questo caso ρ' è accettante perché ogni stato finale di $\mathcal{A}$ e uno stato finale anche di $\mathcal{A}'$. Quindi $L(\mathcal{A}) \subseteq L(\mathcal{A}')$.

Sia ora $w \in L(\mathcal{A}')$ e sia ρ un run accettante di $\mathcal{A}'$ su w. Se $q_0' = q_0$ allora ρ è un run valido anche per $\mathcal{A}$. In caso contrario basta semplicemente sostituire la transizione iniziale (q_0', a, q) con la transizione (q_0, a, q) per ottenere un run valido anche per $\mathcal{A}$. Mostriamo ora che ρ è accettante anche per $\mathcal{A}$ considerando uno degli stati finali $f \in F'$ che si ripete infinite volte in ρ: se $f \in F$ allora ρ è accettante anche per $\mathcal{A}$. Se invece $f \notin F$ significa che esiste uno stato finale f' di F tale che $f' \rightarrow^* f$, e tale stato f' appartiene allo stesso ciclo finale di f; quindi anche f' si ripete infinite volte e ρ è accettante anche per $\mathcal{A}$.

Quindi vale anche l'inclusione $L(\mathcal{A}') \subseteq L(\mathcal{A})$ e i due automi riconoscono lo stesso linguaggio. □

4.3.4 Proprietà effettive di chiusura

Gli automi fondamentalmente periodici sono chiusi per concatenazione ed unione in modo effettivo (come descritto nelle Proposizioni 4.27 e 4.28) e per intersezione poiché l'intersezione di insiemi di sole parole fondamentalmente periodiche è ancora un insieme di sole parole fondamentalmente periodiche. Il metodo effettivo per costruire l'automa intersezione a partire da due automi fondamentalmente periodici (in forma normale) è descritto nel teorema seguente.

Teorema 4.35 (Chiusura effettiva per intersezione). *Dati due automi di Büchi fondamentalmente periodici in forma normale $\mathcal{A}_1$ e $\mathcal{A}_2$ che riconoscono i linguaggi L_1 e L_2 è possibile costruire effettivamente l'automa prodotto $\mathcal{A}_1 \times \mathcal{A}_2$ che riconosce il linguaggio $L_1 \cap L_2$, e tale automa è ancora fondamentalmente periodico ed in forma normale.*

Dimostrazione. La costruzione dell'automa prodotto avviene in modo simile a quello utilizzato per gli automi di Büchi generali: se $\mathcal{A}_1 = (Q_1, \Delta_1, q_0^1, F_1)$ e $\mathcal{A}_2 = (Q_2, \Delta_2, q_0^2, F_2)$ l'automa prodotto $\mathcal{A}_1 \times \mathcal{A}_2 = (Q, \Delta, q_0, F)$ è così definito:

- $Q = Q_1 \times Q_2$;
- Δ è tale che $(\langle q_1, q_2\rangle, a, \langle q_1', q_2'\rangle) \in \Delta$ se e solo se $(q_1, a, q_1') \in \Delta_1$ e $(q_2, a, q_2') \in \Delta_2$, per ogni $q_1, q_1' \in Q_1$, $q_2, q_2' \in Q_2$ e $a \in A$.
- $q_0 = \langle q_0^1, q_0^2\rangle$;
- $F = \{\langle q_1, q_2\rangle | q_1 \in F_1 \text{ e } q_2 \in F_2\}$.

Dimostriamo che l'automa così definito è tale che per ogni coppia di stati $\langle q_1, q_2\rangle, \langle q_1', q_2'\rangle \in Q$ e per ogni parola finita $v \in A^*$ si ha che $\langle q_1, q_2\rangle \xrightarrow[\mathcal{A}_1\times\mathcal{A}_2]{v} \langle q_1', q_2'\rangle$ se e solo se $q_1 \xrightarrow[\mathcal{A}_1]{v} q_1'$ e $q_2 \xrightarrow[\mathcal{A}_2]{v} q_2'$ per induzione sulla lunghezza di v:

- Se $v = \varepsilon$ la proprietà è banalmente verificata.
- Sia $v = u \cdot a$, con $u \in A^*$ e $a \in A$. Siano $\langle q_1, q_2\rangle, \langle q_1', q_2'\rangle \in Q$ due stati tali che $\langle q_1, q_2\rangle \xrightarrow[\mathcal{A}_1\times\mathcal{A}_2]{v} \langle q_1', q_2'\rangle$: allora esiste uno stato $\langle q_1'', q_2''\rangle$ tale che $\langle q_1, q_2\rangle \xrightarrow[\mathcal{A}_1\times\mathcal{A}_2]{u} \langle q_1'', q_2''\rangle \xrightarrow[\mathcal{A}_1\times\mathcal{A}_2]{a} \langle q_1', q_2'\rangle$ e, per ipotesi induttiva, $q_1 \xrightarrow[\mathcal{A}_1]{u} q_1''$ e $q_2 \xrightarrow[\mathcal{A}_2]{u} q_2''$. Per la definizione della relazione di transizione $(\langle q_1'', q_2''\rangle, a, \langle q_1', q_2'\rangle) \in \Delta$ se e solo se $(q_1'', a, q_1') \in \Delta_1$ e $(q_2'', a, q_2') \in \Delta_2$, quindi $q_1 \xrightarrow[\mathcal{A}_1]{v} q_1'$ e $q_2 \xrightarrow[\mathcal{A}_2]{v} q_2'$.

  Siano ora $q_1, q_1' \in Q_1$ e $q_2, q_2' \in Q_2$ tali che $q_1 \xrightarrow[\mathcal{A}_1]{v} q_1'$ e $q_2 \xrightarrow[\mathcal{A}_2]{v} q_2'$. Per ipotesi induttiva esistono due stati q_1'' e q_2'' tali che $q_1 \xrightarrow[\mathcal{A}_1]{u} q_1'' \xrightarrow[\mathcal{A}_1]{a} q_1'$, $q_2 \xrightarrow[\mathcal{A}_2]{u} q_2'' \xrightarrow[\mathcal{A}_2]{a} q_2'$ e $\langle q_1, q_2\rangle \xrightarrow[\mathcal{A}_1\times\mathcal{A}_2]{u} \langle q_1'', q_2''\rangle$. Per la definizione della relazione di transizione $(\langle q_1'', q_2''\rangle, a, \langle q_1', q_2'\rangle) \in \Delta$ e quindi $\langle q_1, q_2\rangle \xrightarrow[\mathcal{A}_1\times\mathcal{A}_2]{v} \langle q_1', q_2'\rangle$.

Sia ora $w \in L_1 \cap L_2$. La parola w è una parola fondamentalmente periodica riconosciuta sia da $\mathcal{A}_1$ che da $\mathcal{A}_2$ ed esistono quindi due run ρ_1 e ρ_2 che riconoscono w rispettivamente in $\mathcal{A}_1$ e in $\mathcal{A}_2$. Per le proprietà degli automi fondamentalmente periodici tali run sono fondamentalmente periodici e composti da un ciclo finale che si ripete infinite volte: poiché negli automi in forma normale tutti gli stati di un ciclo finale sono finali esiste un istante n a partire dal quale i suffissi infiniti $\rho_1[n, +\infty[$ e $\rho_2[n, +\infty[$ sono composti da soli stati finali.

Costruiamo quindi il run prodotto $\rho_1 \times \rho_2$:

$$\rho_1 \times \rho_2 = \langle \rho_1(1), \rho_2(1)\rangle \xrightarrow{w(1)} \dots \xrightarrow{w(n-1)} \langle \rho_1(n), \rho_2(n)\rangle \xrightarrow{w(n)} \dots$$

esso è un run dell'automa $\mathcal{A}_1 \times \mathcal{A}_2$ ed inoltre a partire dall'istante n è composto solamente da coppie di stati finali. Esiste quindi uno stato finale di $\mathcal{A}_1 \times \mathcal{A}_2$ che si ripete infinite volte e $\rho_1 \times \rho_2$ è un run accettante per w.

Sia $w \in L(\mathcal{A}_1 \times \mathcal{A}_2)$ e sia ρ un run accettante per w. Se $\langle q_1, q_2 \rangle$ è uno stato finale che si ripete infinite volte in ρ, il run è così composto:

$$\rho = \langle q_0^1, q_0^2 \rangle \to^* \langle q_1, q_2 \rangle \to^* \langle q_1, q_2 \rangle \to^* \dots$$

Le proiezioni di ρ rispetto alla prima e alla seconda componente sono quindi run validi per $\mathcal{A}_1$ e per $\mathcal{A}_2$ e sono inoltre accettanti perché contengono infinite occorrenze degli stati finali q_1 e q_2, rispettivamente. Quindi $w \in L_1 \cap L_2$.

L'automa $\mathcal{A}_1 \times \mathcal{A}_2$ riconosce quindi il linguaggio $L_1 \cap L_2$: studiamo ora la sua struttura a partire dagli stati finali.

Sia $\langle q_1, q_2 \rangle \in F$: allora $q_1 \in F_1$ e $q_2 \in F_2$. Poiché $\mathcal{A}_1$ e $\mathcal{A}_2$ sono automi fondamentalmente periodici in forma normale si ha che:

- $q_1 \to^* q_1$ e $q_2 \to^* q_2$;
- esiste esattamente una transizione uscente da q_1 e una uscente da q_2: siano (q_1, a_1, q_1') e (q_2, a_2, q_2') tali transizioni;
- tutti gli stati raggiungibili da q_1 in $\mathcal{A}_1$ e da q_2 in $\mathcal{A}_2$ sono finali.

Lo stato $\langle q_1, q_2 \rangle$ è quindi tale che:

- se $a_1 \neq a_2$, $\langle q_1, q_2 \rangle$ non possiede transizioni uscenti;
- se $a_1 = a_2$, $\langle q_1, q_2 \rangle$ possiede una sola transizione uscente ed essa è $(\langle q_1, q_2 \rangle, a_1, \langle q_1', q_2' \rangle)$. Inoltre $\langle q_1', q_2' \rangle$ è uno stato finale perché $q_1' \in F_1$ e $q_2' \in F_2$.

Poiché ogni stato raggiungibile da q_1 o da q_2 è uno stato finale e possiede esattamente una transizione uscente, tutti gli stati raggiungibili da $\langle q_1, q_2 \rangle$ sono finali, possiedono al più una transizione uscente ed esiste un unico run massimale (finito o infinito) possibile a partire da $\langle q_1, q_2 \rangle$.

Sia v_1 il periodo riconosciuto da $\mathcal{A}_1$ a partire da q_1 e sia v_2 il periodo riconosciuto da $\mathcal{A}_2$ a partire da q_2:

- se $v_1^\omega = v_2^\omega$, esistono n, m tali che $v_1^n = v_2^m$. Allora $\langle q_1, q_2 \rangle \to^* \langle q_1, q_2 \rangle$ perché dopo aver riconosciuto v_1^n partendo da q_1 l'automa $\mathcal{A}_1$ ritorna in q_1 e dopo aver riconosciuto v_2^m partendo da q_2 l'automa $\mathcal{A}_2$ ritorna in q_2. Quindi $\langle q_1, q_2 \rangle \xrightarrow[\mathcal{A}_1 \times \mathcal{A}_2]{v_1^n} \langle q_1, q_2 \rangle$ e lo stato $\langle q_1, q_2 \rangle$ rispetta la definizione di automa fondamentalmente periodico in forma normale perché gli unici stati raggiungibili sono quelli che compongono il ciclo che riconosce $v_1^n = v_2^m$ che sono finali e possiedono una sola transizione uscente.
- Se $v_1^\omega \neq v_2^\omega$, esiste uno stato raggiungibile da $\langle q_1, q_2 \rangle$ senza transizioni uscenti (che corrisponde al primo carattere diverso tra v_1^ω e v_2^ω). Dunque l'unico run massimale possibile che inizia in $\langle q_1, q_2 \rangle$ è finito e termina in uno stato senza transizioni uscenti, quindi $\langle q_1, q_2 \rangle \not\to^* \langle q_1, q_2 \rangle$ (in caso contrario si può creare un run ciclico infinito) e lo stato non rispetta la definizione di automa fondamentalmente periodico.

Quindi non tutti gli stati finali di $\mathcal{A}_1 \times \mathcal{A}_2$ rispettano la definizione di automa fondamentalmente periodico. Per creare un automa di Büchi fondamentalmente periodico che riconosca il linguaggio $L_1 \cap L_2$ è però sufficiente eliminare da $\mathcal{A}_1 \times \mathcal{A}_2$ tutti gli stati finali da cui non è possibile costruire un ciclo che ritorni ad essi.

L'automa così creato rispetta la definizione di automa fondamentalmente periodico in forma normale perché tutti gli stati raggiungibili da uno stato finale sono finali e lo stato iniziale $\langle q_0^1, q_0^2 \rangle$ di $\mathcal{A}_1 \times \mathcal{A}_2$ non è uno stato finale perché q_0^1 e q_0^2 non sono finali. □

Capitolo 5

Equivalenza tra automi

Nel capitolo precedente è stata stabilita una classe di automi in grado di rappresentare insiemi di parole fondamentalmente periodiche che può essere utilizzata per rappresentare insiemi di granularità. In questo capitolo si mostra come questa classe di automi consenta di eseguire effettivamente operazioni sugli insiemi di granularità presentando alcuni problemi paradigmatici che possono essere risolti nella classe degli automi fondamentalmente periodici.

Nella prima parte del capitolo verranno affrontati i problemi la cui soluzione può essere ricavata facilmente dalla teoria degli automi di Büchi, mentre nella seconda parte del capitolo verrà studiato in dettaglio il problema dell'equivalenza tra gli automi fondamentalmente periodici.

5.1 Problemi paradigmatici

5.1.1 Problema del vuoto

Il *problema del vuoto* è uno dei problemi cruciali della teoria degli automi, la cui risoluzione consente di affrontare molti altri problemi paradigmatici che si possono ridurre ad esso. Formalmente la sua definizione è la seguente.

Definizione 5.1 (Problema del vuoto). Dato un automa fondamentalmente periodico $\mathcal{A} = (Q, \Delta, q_0, F)$ il *problema del vuoto* consiste nello stabilire se $L(\mathcal{A}) = \emptyset$.

Un esempio di applicazione del problema del vuoto è il *controllo di consistenza* di un insieme di granularità: si immagini di costruire un automa per rappresentare un insieme di granularità che soddisfi determinati vincoli (forniti, per esempio, tramite formule di una opportuna logica); il problema del vuoto consente di stabilire se tale insieme contiene granularità e quindi se esistono granularità che soddisfino i vincoli fissati.

Negli automi di Büchi il problema del vuoto si risolve in tempo lineare effettuando una visita in profondità degli stati dell'automa e cercando un cammino dallo stato iniziale ad uno stato finale ed un ciclo che ritorni sullo stesso stato finale. Nel caso degli automi fondamentalmente periodici è sufficiente determinare un cammino dallo stato iniziale ad uno stato finale: le condizioni imposte dalla Definizione 4.29 garantiscono infatti che per ogni stato finale esiste sempre un cammino che ritorna ad esso.

5.1.2 Problema dell'appartenenza

Definizione 5.2 (Problema dell'appartenenza). Data una parola fondamentalmente periodica w ed un automa fondamentalmente periodico $\mathcal{A} = (Q, \Delta, q_0, F)$ il *problema dell'appartenenza* consiste nello stabilire se $w \in L(\mathcal{A})$.

Questo problema consente di stabilire se una specifica granularità (rappresentata dalla parola w) appartiene all'insieme di granularità rappresentato dall'automa $\mathcal{A}$ e può essere risolto riducendolo al problema del vuoto, come mostrato nella Proposizione seguente.

Proposizione 5.3. *Il problema dell'inclusione per gli automi fondamentalmente periodici può essere ridotto al problema del vuoto.*

Dimostrazione. Sia $w \in A^\omega$ una parola fondamentalmente periodica e sia $\mathcal{A} = (Q, \Delta, q_0, F)$ un automa fondamentalmente periodico. La parola w appartiene al linguaggio di $\mathcal{A}$ se e solo se l'insieme $\{w\} \cap \mathcal{L}(\mathcal{A})$ è non vuoto. Poiché è possibile costruire (data una suddivisione prefisso-periodo di w) l'automa fondamentalmente periodico che riconosce il linguaggio $\{w\}$ e, poiché è possibile costruire effettivamente l'automa che riconosce il linguaggio $\{w\} \cap L(\mathcal{A})$ (cfr. Teorema 4.35), il problema dell'appartenenza per gli automi fondamentalmente periodici è decidibile. □

Il *problema dell'appartenenza* di una parola w ad un linguaggio L consente di risolvere tutti i problemi di appartenenza di una specifica granularità ad un insieme di granularità. Si immagini per esempio di costruire l'insieme di tutte le granularità che soddisfano determinati vincoli: la verifica dell'appartenenza di una granularità all'insieme consente di stabilire se la granularità soddisfa i vincoli fissati.

Osservazione 5.4. Si noti come il problema dell'appartenenza sia (in generale) indecidibile per gli automi di Büchi, a causa del fatto che non è possibile dare una rappresentazione finita per tutte le parole che possono essere contenute nel linguaggio (in particolare per le parole non fondamentalmente periodiche). Negli automi fondamentalmente periodici è invece possibile rappresentare in maniera finita la parola di cui si vuole testare l'appartenenza, ed il problema diventa decidibile in maniera molto semplice.

5.1.3 Problema dell'equivalenza

Il *problema dell'equivalenza* è un altro problema di cruciale della teoria degli automi. La sua risoluzione consente di affrontare altri problemi interessanti, come il problema dell'inclusione, che vedremo in seguito.

Definizione 5.5. Siano $\mathcal{A}_1$ e $\mathcal{A}_2$ due automi fondamentalmente periodici. Il *problema dell'equivalenza* consiste nel determinare se $L(\mathcal{A}_1) = L(\mathcal{A}_2)$.

Il problema dell'equivalenza viene risolto, negli automi di Büchi, riducendolo al problema del vuoto: infatti $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ se e solo se l'insieme $(L(\mathcal{A}_1) \cup L(\mathcal{A}_2)) \setminus (L(\mathcal{A}_1) \cap L(\mathcal{A}_2))$ è vuoto. Tuttavia questo approccio richiede la costruzione dell'automa che riconosce la differenza insiemistica tra due linguaggi e tale passo può essere effettuato solamente se la classe di automi considerata è chiusa rispetto alla complementazione (perché $L_1 \setminus L_2 = L_1 \cap \overline{L}_2$).

Nel caso degli automi fondamentalmente periodici questo non è possibile, quindi il problema dell'equivalenza non può essere risolto in maniera analoga agli automi di Büchi. Nelle sezioni successive mostreremo come sia possibile risolvere il problema dell'equivalenza anche per gli automi fondamentalmente periodici attraverso la costruzione di una opportuna *forma canonica* che garantisca che, se due automi riconoscono lo stesso linguaggio, allora la loro forma canonica è identica (a meno di rinomina degli stati).

Supponendo di riuscire a risolvere il problema dell'equivalenza, vediamo un esempio di un problema interessante riducibile ad esso.

5.1.4 Problema dell'inclusione

Definizione 5.6. Siano $\mathcal{A}_1$ e $\mathcal{A}_2$ due automi fondamentalmente periodici. Il *problema dell'inclusione* consiste nel determinare se $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$.

Il problema dell'inclusione consente di risolvere tutti i problemi di inclusione tra insiemi di granularità. Inoltre questo problema può essere ridotto ad un problema di equivalenza, come mostrato dalla Proposizione seguente.

Proposizione 5.7. *Il problema dell'inclusione per gli automi fondamentalmente periodici può essere ridotto al problema dell'equivalenza.*

Dimostrazione. Siano $\mathcal{A}_1$ e $\mathcal{A}_2$ due automi fondamentalmente periodici, allora $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ se e solo se $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) = \mathcal{A}_1$. Poiché gli automi fondamentalmente periodici sono chiusi per intersezione, il problema dell'inclusione è effettivamente riducibile ad un problema di equivalenza. □

5.2 Forma canonica

Nella sezione precedente abbiamo discusso alcuni problemi interessanti per le applicazioni degli automi fondamentalmente periodici ed abbiamo osservato come la teoria degli automi di Büchi fornisca le soluzioni per alcuni di essi. Per il problema dell'equivalenza si è invece mostrato come non sia possibile utilizzare i risultati preesistenti, rendendo necessario un approccio diverso per determinare una soluzione.

In questa sezione affronteremo il problema dell'equivalenza attraverso la definizione di opportuni algoritmi che consentono di ottenere una *forma canonica* per gli automi fondamentalmente periodici. In questo modo si garantisce che, dopo l'esecuzione degli algoritmi, due automi riconoscono lo stesso linguaggio se e solo se la loro struttura è identica a meno di isomorfismi (ad esempio, rinomina degli stati). Questo approccio è strettamente correlato con la teoria degli automi regolari su parole finite, ed in particolare con il Teorema di Myhill-Nerode e gli algoritmi di minimizzazione per gli automi regolari. Infatti, nel caso degli automi regolari, il problema dell'equivalenza viene risolto attraverso la determinazione dell'*automa minimo* che riconosce il linguaggio: l'unicità di tale automa garantisce che due automi riconoscono lo stesso linguaggio se e solo se i due automi minimi sono identici. Per ulteriori approfondimenti sulla teoria degli automi regolari si veda [HU80].

L'approccio utilizzato in questa sezione non porta tuttavia alla costruzione di un automa minimo per i linguaggi di sole parole fondamentalmente periodiche,

ma sfrutta gli algoritmi di minimizzazione degli automi regolari per ottenere una forma canonica che è unica rispetto al linguaggio riconosciuto, e consente quindi di confrontare direttamente due automi fondamentalmente periodici per stabilirne l'equivalenza.

Presentiamo la nostra soluzione al problema dell'equivalenza definendo due linguaggi regolari di parole finite costruiti a partire da un automa fondamentalmente periodico e fornendo successivamente una serie di algoritmi che operano su tali linguaggi e sulla struttura dell'automa per costruire un automa che è unico (a meno di isomorfismi) rispetto al linguaggio riconosciuto.

5.2.1 Linguaggio dei prefissi e linguaggio dei periodi

La particolare struttura degli automi fondamentalmente periodici ricalca la rappresentazione *prefisso, periodo* di una parola fondamentalmente periodica, ed implicitamente suddivide ogni parola accettata in un prefisso, corrispondente alla parte di parola riconosciuta prima di entrare in un ciclo finale, ed un periodo riconosciuto infinite volte da un ciclo finale.

Questa osservazione permette di definire il *linguaggio dei prefissi* ed il *linguaggio dei periodi* di un automa fondamentalmente periodico, cercando di operare su di essi per ottenere un automa che riconosce lo stesso linguaggio ma in cui l'implicita suddivisione prefisso-periodo sia quella minimale. Questa costruzione non porta alla definizione di un automa minimale per il linguaggio, ma è il primo passo indispensabile per definire una forma canonica per gli automi fondamentalmente periodici.

Definizione 5.8. Se $\mathcal{A} = (Q, \Delta, q_0, F)$ è un automa di Büchi fondamentalmente periodico in forma normale, definiamo il *linguaggio dei prefissi di* $\mathcal{A}$ come l'insieme $L^{pre}(\mathcal{A}) = \{v \in A^* | q_0 \stackrel{v}{\rightsquigarrow} q_f, q_f \in F\}$ ed il *linguaggio dei periodi di* $\mathcal{A}$ come l'insieme $L^{per}(\mathcal{A})$ dei periodi riconosciuti da ognuno dei cicli finali dell'automa a partire da ognuno degli stati finali di $\mathcal{A}$. Formalmente si ha che $L^{per}(\mathcal{A}) = \{v \in A^* | \exists f \in F. f \stackrel{v}{\rightarrow} f \text{ e } \forall u \in A^*. f \stackrel{u}{\rightarrow} f \Rightarrow |v| \leq |u|\}$.

Si può dimostrare facilmente come i linguaggi $L^{pre}(\mathcal{A})$ e $L^{per}(\mathcal{A})$ siano regolari costruendo gli automi che li riconoscono a partire dall'automa $\mathcal{A}$. Il linguaggio dei prefissi $L^{pre}(\mathcal{A})$ è riconosciuto dall'automa $\mathcal{A}^{pre} = (Q, \Delta_{pre}, q_0, F)$, dove la relazione di transizione Δ_{pre} è ottenuta a partire dalla relazione di transizione di $\mathcal{A}$, Δ, eliminando le transizioni uscenti dagli stati finali. La dimostrazione della regolarità di $L^{per}(\mathcal{A})$ si basa invece sull'osservazione che:

1. gli stati finali di $\mathcal{A}$ sono in numero finito;
2. ad ogni stato finale corrisponde un'unico periodo.

Il linguaggio dei periodi di $\mathcal{A}$ ha quindi cardinalità finita, ed è perciò un insieme regolare.

È importante notare che due automi che riconoscono lo stesso linguaggio possono avere linguaggi dei prefissi o dei periodi differenti, poiché la divisione prefisso-periodo delle parole riconosciute può essere diversa; inoltre due automi che possiedono lo stesso linguaggio dei prefissi e lo stesso linguaggio dei periodi possono collegare prefissi e periodi in maniera diversa, riconoscendo quindi linguaggi di parole fondamentalmente periodiche diversi.

La definizione del linguaggio dei prefissi e del linguaggio dei periodi di un automa fondamentalmente periodico evidenzia tuttavia la necessità di un passo preliminare fondamentale per la costruzione di una forma canonica: è necessario che l'implicita suddivisione prefisso-periodo delle parole riconosciute dall'automa sia univoca perché l'automa possa essere trasformato in una forma canonica unica per ogni linguaggio riconosciuto. I passi necessari per poter costruire una forma canonica per un generico automa fondamentalmente periodico sono quindi i seguenti:

1. Il linguaggio dei periodi viene minimizzato in modo da comprendere solamente periodi che non sono ripetizione di sottoperiodi più corti. Dopo questo passo viene inoltre garantito che la struttura dell'automa (limitatamente ai soli stati finali) è unica e minimale.

2. Il linguaggio dei prefissi viene minimizzato, in modo da garantire che per ogni parola riconosciuta l'implicita suddivisione prefisso-periodo sia quella minimale. Al termine di questo passo si garantisce che due automi che riconoscono lo stesso linguaggio possiedono gli stessi linguaggi dei prefissi e dei periodi.

3. La parte di automa che riconosce i prefissi viene posta in forma canonica sfruttando gli algoritmi di minimizzazione per gli automi regolari su parole finite. Al termine di questo passo anche la parte di automa che non comprende gli stati finali è unica rispetto al linguaggio riconosciuto.

Nelle prossime sezioni mostriamo in dettaglio le operazioni necessarie per compiere i tre passi della procedura di costruzione di una forma canonica per gli automi fondamentalmente periodici.

5.2.2 Minimo linguaggio dei periodi

La procedura di minimizzazione del linguaggio dei periodi degli automi fondamentalmente periodici è basata sugli algoritmi di partizionamento stabile per singole funzioni, in particolare sulla soluzione lineare proposta da Paige, Tarjan e Bonic in [PTB85].

Analizzando la struttura di un automa fondamentalmente periodico in forma normale si può infatti osservare che gli stati finali formano un grafo diretto in cui ogni nodo ha grado uno, e che soddisfa le seguenti proprietà:

1. il grafo ha almeno un ciclo (se l'insieme degli stati finali non è vuoto);

2. non esistono cammini al di fuori dei cicli del grafo;

3. nessun cammino può uscire dal ciclo in cui si trova;

4. non esistono cammini che vanno da un ciclo ad un altro ciclo.

Il grafo formato dagli stati finali è infatti costituito da un insieme di cicli disgiunti tra loro, ognuno dei quali riconosce un solo periodo. L'approccio di Paige, Tarjan e Bonic al problema del partizionamento stabile per singole funzioni è quello di rappresentare la funzione da partizionare mediante un grafo e di fonderne quindi i nodi fino ad ottenere la partizione stabile più grossolana. Il grafo che rappresenta la funzione è un grafo diretto in cui tutti i nodi hanno

grado uno ed è costituito da cicli disgiunti e da alberi radicati nei cicli: il grafo degli stati finali di un automa fondamentalmente periodico in forma normale soddisfa quindi le stesse proprietà del grafo che rappresenta la funzione. L'algoritmo di Paige, Tarjan e Bonic può quindi essere utilizzato per minimizzare la parte di automa che comprende solamente gli stati finali, ottenendo una forma unica per tale porzione di automa.

Prima di presentare in dettaglio l'algoritmo, ricordiamo alcune proprietà dei periodi e delle rotazioni di stringhe che risulteranno utili.

Definizione 5.9. Se $s \in A^*$ è una stringa qualsiasi, il *minimo prefisso che si ripete in* s è il più piccolo prefisso p di s tale che $p^k = s$, con k opportuno.

Definizione 5.10. Se $C = c_1 \cdot \ldots \cdot c_k$ è un ciclo di stati finali dell'automa fondamentalmente periodico in forma normale $\mathcal{A}$, una *rotazione di* C è un ciclo di stati finali C' nella forma $C' = c_i \cdot \ldots \cdot c_k \cdot c_1 \cdot \ldots \cdot c_{i-1}$.

Definizione 5.11. Sia $<$ un ordine totale sull'alfabeto A. L'*ordine lessicografico* $<_{lex}$ è un ordine totale sulle stringhe finite di A tale che $x <_{lex} y$ se e solo se:

1. $\exists j | x(j) < y(j)$ e $x(i) = y(i), \forall i = 1, \ldots, j-1$, oppure
2. $|x| \leq |y|$ e $x(i) = y(i), \forall i = 1, \ldots, |x|$.

Definizione 5.12. Se C è un ciclo di stati finali, una rotazione C' di C è detta *minima rispetto all'ordine lessicografico* se, per ogni rotazione C'' di C, il periodo p' riconosciuto da C' è minore o uguale (rispetto all'ordine lessicografico) rispetto al periodo p'' riconosciuto da C''.

Algoritmo di minimizzazione degli stati finali

L'assenza (nel caso degli stati finali di una automa fondamentalmente periodico in forma normale) di porzioni di grafo al di fuori dei cicli consente di semplificare l'algoritmo rispetto a quello di Paige, Tarjan e Bonic eliminando i passi che si occupano dei nodi posti al di fuori dei cicli. L'algoritmo minimizza la parte di automa che comprende i soli stati finali nel modo seguente:

1. per ogni ciclo finale viene minimizzato il periodo riconosciuto, determinando il minimo prefisso che si ripete nel ciclo;

2. si determina la minima rotazione del ciclo rispetto all'ordine lessicografico: poiché il periodo riconosciuto dai cicli è minimo, tale rotazione è unica;

3. si ordinano lessicograficamente i cicli rispetto al periodo riconosciuto dalla loro minima rotazione;

4. tramite una singola scansione dei cicli ordinati precedentemente si determinano i cicli che riconoscono lo stesso periodo, che vengono quindi fusi.

La procedura appena descritta è effettiva: ricordiamo infatti che la determinazione del minimo prefisso che si ripete in un ciclo può essere risolta mediante una procedura di string-matching semplicemente determinando la prima occorrenza non banale del periodo p in $p \cdot p$, mente la determinazione della minima

rotazione lessicografica viene risolta in [Boo80] e [Shi81] con un algoritmo lineare rispetto alla lunghezza della stringa.

L'Algoritmo 5.1 mostra nel dettaglio come opera la procedura che minimizza la parte comprendente i soli stati finali di un automa fondamentalmente periodico: la procedura richiede in input un automa fondamentalmente periodico $\mathcal{A}$ in forma normale e ritorna in output un automa che riconosce lo stesso linguaggio di partenza ma è tale che ogni ciclo finale riconosce un periodo minimale (cioè che non possiede sottoperiodi propri) e tale che non esistono due stati finali a partire dai quali è possibile riconoscere lo stesso periodo. L'array *Cycles* è utilizzato nell'algoritmo per memorizzare ed ordinare i cicli finali.

Algoritmo 5.1 Minimizzazione degli stati finali

1: **for all** unmarked $q_f \in F$ **do**
2: $C_f \leftarrow$ ciclo finale che inizia in q_f
3: $v_f \leftarrow$ periodo riconosciuto da C_f
4: $i \leftarrow$ offset della prima occorrenza non banale di v in $v \cdot v$
5: **if** $i < |v|$ **then**
6: *delete transition* $(C_f[i], v[i], C_f[i+1])$
7: *insert transition* $(C_f[i], v[i], C_f[1])$
8: **for** $j \leftarrow i+1, \ldots, |v|$ **do**
9: **for all** $(q', a', C_f[j])$ tale che $q' \notin F$ **do**
10: *delete transition* $(q', a', C_f[j])$
11: *insert transition* $(q', a', C_f[((j-1) \ mod \ i)+1])$
12: **end for**
13: *delete state* $(C_f[j])$
14: **end for**
15: **end if**
16: $C'_f \leftarrow$ minima rotazione lessicografica di C_f
17: $v'_f \leftarrow$ periodo riconosciuto da C'_f
18: *insert* $\langle C'_f, v'_f \rangle$ *in Cycles*
19: *mark all states* (C_f)
20: **end for**
21: *sort* $(Cycles)$
22: $\langle C'_f, v'_f \rangle \leftarrow Cycles[1]$
23: **for** $i \leftarrow 2, \ldots, |Cycles|$ **do**
24: $\langle C_f, v_f \rangle \leftarrow Cycles[i]$
25: **if** $v_f = v'_f$ **then** {Fondi i cicli C_f e C'_f}
26: **for** $j \leftarrow 1, \ldots, |C'_f|$ **do**
27: **for all** $(q, a, C'_f[j]) \in \Delta$ tale che $q \notin F$ **do**
28: *delete transition* $(q, a, C'_f[j])$
29: *insert transition* $(q, a, C_f[j])$
30: **end for**
31: *delete state* $(C'_f[j])$
32: **end for**
33: **end if**
34: $\langle C'_f, v'_f \rangle \leftarrow \langle C_f, v_f \rangle$
35: **end for**

Il seguente lemma dimostra che l'algoritmo presentato è corretto, cioè che dopo la sua esecuzione l'automa riconosce lo stesso linguaggio di partenza ma

la struttura degli stati finali è unica e minimale.

Teorema 5.13. *Sia $\mathcal{A}$ un automa fondamentalmente periodico in forma normale. Allora l'automa $\mathcal{A}'$ ottenuto dopo l'esecuzione dell'Algoritmo 5.1 è tale che:*

1. *$\mathcal{A}'$ riconosce lo stesso linguaggio di $\mathcal{A}$;*
2. *la struttura degli stati finali di $\mathcal{A}'$ è in forma unica e minimale.*

Dimostrazione. (1) Per dimostrare che $L(\mathcal{A}) = L(\mathcal{A}')$ studiamo la struttura dei cicli finali dell'automa prima dell'esecuzione della riga 21 dell'algoritmo e al termine dell'esecuzione dell'algoritmo.

Se C_f è un ciclo finale dell'automa iniziale $\mathcal{A}$ e v_f il periodo riconosciuto dal ciclo, dopo l'esecuzione delle righe 1-20 dell'algoritmo si possono presentare due casi:

1. il periodo v_f è minimale, quindi il ciclo C_f non viene modificato dall'algoritmo;
2. se il periodo v_f non è minimale il ciclo C_f viene accorciato in modo da riconoscere un periodo minimale, e le transizioni entranti nella parte eliminata vengono spostate opportunamente nella parte iniziale del ciclo, per garantire che vengano riconosciute le stesse parole del ciclo iniziale.

Quindi, dopo l'esecuzione della prima parte dell'algoritmo, l'automa riconosce ancora lo stesso linguaggio di partenza.

Nella seconda parte dell'algoritmo si effettua la fusione dei cicli che riconoscono lo stesso periodo. Se C_f è un ciclo finale presente nell'automa prima dell'esecuzione delle righe 21-35 si possono presentare due casi:

1. il ciclo C_f è presente anche nell'automa finale $\mathcal{A}'$;
2. il ciclo C_f non è presente in $\mathcal{A}'$: allora esiste un opportuno ciclo C'_f di $\mathcal{A}'$ che riconosce lo stesso periodo di C_f, e su cui sono state spostate le transizioni entranti in C_f.

Quindi anche al termine dell'algoritmo il linguaggio riconosciuto dall'automa rimane invariato.

(2) Siano ora $\mathcal{A}_1$ e $\mathcal{A}_2$ due automi fondamentalmente periodici tali che $L(\mathcal{A}_1) = L(\mathcal{A}_2)$: dimostriamo che dopo aver eseguito l'Algoritmo 5.1 su entrambi la loro struttura degli stati finali è identica e minimale.

È sufficiente osservare che la prima parte dell'algoritmo (righe 1-20) fa sì che il periodo riconosciuto da ogni stato finale dell'automa sia minimale, quindi dopo tale passo i linguaggi dei periodi di $\mathcal{A}_1$ e $\mathcal{A}_2$ sono identici perché contengono solamente i periodi minimali delle parole fondamentalmente periodiche riconosciute.

Dopo l'esecuzione della seconda parte dell'algoritmo i cicli che riconoscono lo stesso periodo vengono fusi, ed è quindi garantito che ogni stato finale riconosce un periodo diverso dagli altri. Poiché ogni stato finale riconosce esattamente un periodo, al termine dell'algoritmo gli automi $\mathcal{A}_1$ e $\mathcal{A}_2$ possiedono lo stesso numero di stati finali (uno per ogni periodo minimale) e anche la struttura delle transizioni è identica.

Quindi la struttura degli stati finali dell'automa ottenuto dopo l'esecuzione dell'Algoritmo 5.1 è unica e minimale. □

5.2.3 Minimo linguaggio dei prefissi

Il principio su cui si basa la procedura di minimizzazione del linguaggio dei prefissi di un automa fondamentalmente periodico è quello secondo il quale, se $u \cdot v^\omega$ è una parola fondamentalmente periodica tale che $u(|u|) = v(|v|)$, cioè l'ultimo carattere del prefisso è uguale all'ultimo carattere del periodo, allora $u \cdot v^\omega = u[1, |u| - 1] \cdot (v(|v|) \cdot v[1, |v| - 1])^\omega$. Iterando tale procedimento è possibile, per ogni suddivisione prefisso-periodo di una parola fondamentalmente periodica, determinare una suddivisione minimale rispetto alla lunghezza del prefisso, ed il prefisso così determinato è unico.

Sia ora $\mathcal{A}$ un automa fondamentalmente periodico e sia w una parola riconosciuta. Se ρ è un run accettante di $\mathcal{A}$ su w, esso divide la parola fondamentalmente periodica in un prefisso u tale che $q_0 \xrightarrow[\mathcal{A}]{u} q_f$, con q_f primo stato finale che appare in ρ, ed un periodo v corrispondente al periodo riconosciuto dal ciclo finale che inizia in q_f. Se u è un prefisso non minimale di w, allora $u(|u|) = v(|v|)$ ed il run ρ è così strutturato:

$$\rho = q_0 \overset{u[1,|u|-1]}{\rightsquigarrow} q' \xrightarrow{u(|u|)} q_f \overset{v[1,|v|-1]}{\multimap} q'_f \xrightarrow{v(|v|)} q_f \to \dots$$

Il run può essere facilmente modificato in modo da riconoscere un prefisso di lunghezza minore sostituendo la transizione $q' \xrightarrow{u(|u|)} q_f$ con la transizione $q'_f \xrightarrow{v(|v|)} q_f$, che legge lo stesso carattere ma appartiene ad un ciclo finale. Il primo stato finale che occorre nel run è ora q'_f e quindi la parola w viene suddivisa nel prefisso $u[1, |u| - 1] < u$ e nel periodo $v(|v|) \cdot v[1, |v| - 1]$. La struttura dell'automa $\mathcal{A}$ deve essere modificata di conseguenza: la transizione $(q', u(|u|), q_f)$ viene eliminata da Δ e viene aggiunta per ogni transizione (q'', a, q') entrante in q', una transizione (q'', a, q'_f) che entra nel ciclo finale con un passo di anticipo.

Algoritmo 5.2 Minimizza il linguaggio dei prefissi di $\mathcal{A}$

```
1:  c ← true
2:  while c = true do
3:    c ← false
4:    for all q_f ∈ F do
5:      let q'_f, a_f tale che (q'_f, a_f, q_f) ∈ Δ e q'_f ∈ F
6:      for all (q, a, q_f) non marcate t.c. a = a_f e q ∉ F do
7:        c ← true
8:        mark(q, a, q_f)
9:        for all (q', a', q) ∈ Δ do
10:         insert transition(q', a', q'_f)
11:       end for
12:     end for
13:   end for
14: end while
15: clear(A)
```

L'Algoritmo 5.2 minimizza il linguaggio dei prefissi dell'automa $\mathcal{A} = (Q, \Delta, q_0, F)$ dato in input. Per assicurare la corretta terminazione si assume che $\mathcal{A}$ sia in forma normale ed inoltre che lo stato iniziale q_0 non possieda transizioni

entranti. La procedura *clear* richiamata alla riga 15 elimina le transizioni marcate e le parti di automa non più raggiungibili a partire dallo stato iniziale o dalle quali non è più possibile raggiungere uno stato finale.

La strategia utilizzata è quella di esaminare le transizioni entranti in ogni stato finale dell'automa: se esiste una transizione (q', a, q_f) proveniente da uno stato non finale etichettata con lo stesso carattere dell'unica transizione (q'_f, a, q_f) proveniente da uno stato finale, essa viene marcata (per essere rimossa in seguito) e vengono aggiunte le opportune transizioni entranti in q'_f. Al termine vengono eliminate le transizioni che sono state marcate e le parti di automa che non sono raggiungibili dallo stato iniziale o che non possono raggiungere uno stato finale. La marcatura delle transizioni e la loro eliminazione al termine della procedura servono per evitare le situazioni in cui, a causa di un ciclo di stati non finali dell'automa, si continua ad eliminare e successivamente inserire la stessa transizione infinite volte: se una transizione viene marcata essa non può più essere inserita in Δ e viene ignorata nei passi successivi.

Per dimostrare la correttezza dell'algoritmo dimostriamo per prima cosa che dopo ogni iterazione l'automa risultante riconosce lo stesso linguaggio dell'automa di partenza e successivamente che, quando l'algoritmo termina, il linguaggio dei prefissi dell'automa risultante è minimale.

Lemma 5.14. *Sia $\mathcal{A} = (Q, \Delta, q_0, F)$ un automa fondamentalmente periodico e sia $q_f \in F$ uno stato finale tale che $(q'_f, a, q_f) \in \Delta$ con $q'_f \in F$ e $(q, a, q_f) \in \Delta$ con $q \in Q \setminus F$, $q \neq q_0$. Allora l'automa $\mathcal{A}' = (Q, \Delta', q_0, F)$, dove*

$$\Delta' = (\Delta \setminus \{(q, a, q_f)\}) \cup \{(q', a', q'_f) | (q', a', q) \in \Delta\}$$

riconosce lo stesso linguaggio di $\mathcal{A}$.

Dimostrazione. Sia $\mathcal{A} = (Q, \Delta, q_0, F)$ un automa fondamentalmente periodico e siano $q_f \in F$ e $(q'_f, a, q_f), (q, a, q_f) \in \Delta$ uno stato finale e due transizioni che soddisfano le ipotesi del teorema. Sia $\mathcal{A}' = (Q, \Delta', q_0, F)$ un automa fondamentalmente periodico costruito secondo le ipotesi del teorema. Dimostriamo che $L(\mathcal{A}) = L(\mathcal{A}')$.

Sia $w \in L(\mathcal{A})$ e sia ρ un run accettante per w. Se ρ non contiene nessuna occorrenza della transizione (q, a, q_f) allora ρ è un run valido e accettante anche per l'automa $\mathcal{A}'$, quindi $w \in L(\mathcal{A}')$. In caso contrario ρ contiene esattamente una occorrenza della transizione (q, a, q_f) (q_f è uno stato finale: dopo aver raggiunto uno stato finale l'automa può raggiungere solamente stati finali) ed è possibile costruire un run ρ' dell'automa $\mathcal{A}'$ che accetta w nel modo seguente: se (q', a', q) è la transizione che precede (q, a, q_f) in ρ, il run ρ' è costruito sostituendo (q', a', q) con (q', a', q'_f) e (q, a, q_f) con (q'_f, a, q_f) in ρ. Il run ρ' è un run valido ed accettante di $\mathcal{A}'$ su w perché (q', a', q'_f) e (q'_f, a, q_f) appartengono a Δ'. Quindi $L(\mathcal{A}) \subseteq L(\mathcal{A}')$.

Sia ora $w \in L(\mathcal{A}')$ e sia ρ' un run accettante per w in $\mathcal{A}'$. Se ρ' non contiene nessuna occorrenza delle transizioni (q', a', q'_f) aggiunte all'automa $\mathcal{A}'$ allora è un run valido ed accettante di w anche per l'automa $\mathcal{A}$ e $w \in L(\mathcal{A})$. Nel caso opposto ρ' contiene esattamente una occorrenza di una opportuna transizione (q', a', q'_f) (dopo aver raggiungo q_f l'automa passa solo per stati finali) ed essa è necessariamente seguita dalla transizione (q'_f, a, q_f) perché q_f possiede una sola transizione uscente. Quindi è possibile costruire un run ρ di w in $\mathcal{A}$ sostituendo in ρ' la transizione (q'_f, a, q_f) con la transizione (q, a, q_f)

eliminata nella costruzione di $\mathcal{A}'$ e sostituendo (q', a', q'_f) con (q', a', q). Quindi $w \in L(\mathcal{A})$ e $L(\mathcal{A}') \subseteq L(\mathcal{A})$.

Abbiamo quindi dimostrato che $\mathcal{A}$ e $\mathcal{A}'$ riconoscono lo stesso linguaggio. □

Dimostriamo che l'algoritmo non modifica il linguaggio riconosciuto dall'automa costruendo una opportuna sequenza di automi $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n, \ldots$ tale che $\mathcal{A}_1$ è l'automa iniziale di input e, per ogni $n > 1$, $\mathcal{A}_{n+1}$ è l'automa costruito dall'algoritmo a partire da $\mathcal{A}_n$ dopo l'n-esima esecuzione delle righe 7-11 e dopo aver eliminato le transizioni marcate. Il lemma seguente stabilisce che tutti gli automi della sequenza riconoscono lo stesso linguaggio.

Lemma 5.15. *Sia $\mathcal{A}_n$ un generico elemento della sequenza di automi definita in precedenza. Allora l'automa $\mathcal{A}_{n+1}$, ottenuto dopo l'n-esima esecuzione delle righe 7-11 dell'Algoritmo 5.2 eliminando le transizioni marcate, è tale che $L(\mathcal{A}_n) = L(\mathcal{A}_{n+1})$.*

Dimostrazione. Sia $\mathcal{A}_n = (Q, \Delta, q_0, F)$ un automa come da ipotesi del Lemma. Se l'algoritmo esegue le righe 7-11 significa che esistono uno stato finale q_f, una transizione (q'_f, a_f, q_f), con $q'_f \in F$, ed una transizione (q, a_f, q_f) con $q \in Q \setminus F \setminus \{q_0\}$ non marcata (quindi presente nell'automa $\mathcal{A}_n$) che soddisfano le ipotesi del Lemma 5.14.

L'algoritmo procede quindi come segue:

1. marca la transizione (q, a_f, q_f) (che quindi non è più presente in $\mathcal{A}_{n+1}$);
2. inserisce, per ogni transizione (q', a, q) entrante in q, una transizione (q', a, q'_f), a meno che questa non sia già stata marcata in precedenza.

L'automa $\mathcal{A}_{n+1}$ è quindi costruito in maniera leggermente diversa dall'automa $\mathcal{A}'$ del Lemma 5.14 perché non è detto che tutte le transizioni che vengono aggiunte ad $\mathcal{A}'$ siano aggiunte anche ad $\mathcal{A}_{n+1}$. Tuttavia si vede immediatamente che il linguaggio di $\mathcal{A}_{n+1}$ è contenuto nel linguaggio di $\mathcal{A}'$, perché ogni run di $\mathcal{A}_{n+1}$ è anche un run di $\mathcal{A}'$.

Sia ora (q', a, q'_f) una transizione di $\mathcal{A}'$ ma non di $\mathcal{A}_{n+1}$. Ciò significa che essa è già stata marcata in precedenza dall'algoritmo e non può più essere inserita in $\mathcal{A}_{n+1}$. Quindi esiste uno stato finale $q''_f \in F$ tale che $(q''_f, a, q'_f) \in \Delta$ e che ha consentito di marcare la transizione perché essa può essere "assorbita" dal ciclo finale.

Fissiamo quindi un run ρ dallo stato iniziale q_0 allo stato finale q'_f di $\mathcal{A}'$ la cui ultima transizione sia proprio (q', a, q'_f) e costruiamo un run di $\mathcal{A}_{n+1}$ da q_0 a q'_f che riconosce la stessa parola.

Consideriamo la transizione (q'', a'', q') immediatamente precedente l'ultima transizione: poiché q'' è un predecessore immediato di q', nel momento in cui la transizione (q', a, q_f) viene marcata l'algoritmo inserisce anche una opportuna transizione (q'', a'', q''_f) in Δ. Si possono quindi presentare due casi:

1. la transizione (q'', a'', q''_f) è presente in $\mathcal{A}_{n+1}$, quindi il run ρ' ottenuto da ρ sostituendo (q'', a'', q') con (q'', a'', q''_f) e (q', a, q'_f) con (q''_f, a, q'_f) è un run valido di $\mathcal{A}_{n+1}$ e riconosce la stessa parola di ρ.
2. La transizione (q'', a'', q''_f) non è presente in $\mathcal{A}_{n+1}$ perché è stata a sua volta marcata. In tal caso si considera la transizione che precede (q'', a'', q')

in ρ e si ripete il ragionamento fatto: esiste una opportuna transizione (q''', a''', q_f''') che è stata inserita nell'automa e che (se non è stata marcata) consente di costruire un run ρ' di $\mathcal{A}_{n+1}$ che riconosce la stessa parola di ρ. Se anche questa transizione è stata marcata si procede ulteriormente a ritroso. Il fatto che la transizioni uscenti dallo stato iniziale q_0 non possano essere marcate garantisce che dopo al più $|\rho|$ passi si determina una transizione presente in $\mathcal{A}_{n+1}$ che consente di costruire un run ρ' da q_0 a q'_f che riconosce la stessa parola di ρ.

Quindi gli automi $\mathcal{A}_{n+1}$ e $\mathcal{A}'$ riconoscono esattamente lo stesso linguaggio e, per il Lemma 5.14, anche $\mathcal{A}_n$ e $\mathcal{A}_{n+1}$ riconoscono lo stesso linguaggio. □

Studiamo ora la struttura del linguaggio dei prefissi dell'automa ottenuto dopo l'esecuzione dell'Algoritmo 5.2, dimostrando che è minimale.

Teorema 5.16. *Sia $\mathcal{A} = (Q, \Delta, q_0, F)$ l'automa fondamentalmente periodico ottenuto dopo l'esecuzione dell'Algoritmo 5.2. Allora il linguaggio dei prefissi $L^{pre}(\mathcal{A})$ è tale che ogni prefisso $u \in L^{pre}(\mathcal{A})$ ha lunghezza unitaria, oppure è il prefisso minimo di una opportuna parola fondamentalmente periodica $w \in L(\mathcal{A})$.*

Dimostrazione. L'Algoritmo 5.2 termina quando la variabile booleana c rimane posta a false dopo un'iterazione del ciclo while (2-14). Ciò si può presentare in due casi:

1. tutte le transizioni entranti nei cicli finali a partire da stati non finali (e distinti da q_0) sono state marcate. In questo caso le transizioni che entrano nei cicli finali sono solamente quelle che partono dallo stato iniziale q_0.

2. Tutte le transizioni entranti da stati non finali in ogni stato finale $q_f \in F$ che non sono marcate sono tali che (se (q'_f, a_f, q_f) è l'unica transizione entrante in q_f da uno stato finale) il carattere che etichetta la transizione è diverso da a_f, oppure la transizione entra in q_f a partire dallo stato iniziale.

Sia quindi $w \in L(\mathcal{A})$ e sia ρ un run accettante di $\mathcal{A}$ su w. Per le proprietà degli automi fondamentalmente periodici il run ρ divide la parola w in un prefisso u riconosciuto dalla porzione iniziale di stati non finali di ρ e in un periodo v riconosciuto dal ciclo finale raggiunto da ρ. Se q_f è il primo stato finale incontrato nel run, ρ ha la seguente struttura:

$$\rho = q_0 \xrightarrow{u(1)} \dots \xrightarrow{u(|u|-1)} q' \xrightarrow{u(|u|)} q_f \xrightarrow{v(1)} \dots \xrightarrow{v(|v|-1)} q'_f \xrightarrow{v(|v|)} q_f \xrightarrow{v(1)} \dots$$

Il prefisso u gode quindi delle seguenti proprietà:

1. $|u| \geq 1$, perché q_0 è uno stato non finale;

2. se $|u| > 1$, allora $u(|u|) \neq v(|v|)$, perché in caso contrario la transizione $(q', u(|u|), q_f)$ sarebbe stata marcata ed eliminata dall'automa, e u è il prefisso minimo di w.

Abbiamo quindi dimostrato che ogni prefisso $u \in L^{pre}(\mathcal{A})$ è tale che u ha lunghezza unitaria, oppure è prefisso minimo di una opportuna parola fondamentalmente periodica riconosciuta da $\mathcal{A}$. □

Corollario 5.17. *Siano $\mathcal{A}_1$ e $\mathcal{A}_2$ due automi fondamentalmente periodici tali che $L(\mathcal{A}_1) = L(\mathcal{A}_2)$. Allora dopo l'esecuzione dell'Algoritmo 5.2 si ha che $L^{pre}(\mathcal{A}_1) = L^{pre}(\mathcal{A}_2)$.*

Dimostrazione. Per il Teorema 5.16 appena dimostrato, dopo l'esecuzione dell'Algoritmo 5.2 il linguaggio dei prefissi dei due automi è composto solamente da prefissi di lunghezza unitaria e da prefissi minimi. Poiché il prefisso minimo di ogni parola fondamentalmente periodica è unico, si ha quindi che $L^{pre}(\mathcal{A}_1) = L^{pre}(\mathcal{A}_2)$. □

Abbiamo quindi dimostrato che quando l'Algoritmo 5.2 termina, il linguaggio dei prefissi dell'automa è unico e minimale (cioè contiene solamente prefissi unitari o minimi). Per dimostrare inoltre che l'algoritmo termina sempre, è sufficiente osservare che:

1. l'algoritmo non aggiunge stati all'automa;
2. l'algoritmo si limita a marcare transizioni esistenti e ad aggiungerne delle altre;
3. le transizioni già marcate non vengono più aggiunte all'automa;
4. il numero di transizioni possibili tra stati finali e stati non finali è finito e pari a $|Q \setminus F| \cdot |F| \cdot |A|$.

Di conseguenza, dopo al più $2 \cdot |Q \setminus F| \cdot |F| \cdot |A|$ passi l'algoritmo ha aggiunto tutte le possibili transizioni, le ha marcate tutte per essere eliminate e quindi necessariamente termina.

La procedura di minimizzazione del linguaggio dei prefissi è quindi effettiva: ciò garantisce (assieme ai risultati del Teorema 5.13) che, dopo aver eseguito gli Algoritmi 5.1 e 5.2, un automa fondamentalmente periodico è tale che:

1. il linguaggio dei periodi è minimale e la struttura degli stati finali è minimale;
2. il linguaggio dei prefissi è minimale.

Nella prossima sezione mostreremo come ottenere una struttura unica anche per la parte di automa che comprende solamente gli stati non finali, risolvendo così il problema dell'equivalenza per gli automi fondamentalmente periodici.

5.3 Forma canonica per gli stati non finali

Abbiamo già osservato che è possibile costruire, a partire da un automa fondamentalmente periodico $\mathcal{A}$, l'automa regolare che riconosce il linguaggio dei prefissi di $\mathcal{A}$. Tale automa può essere utilizzato (con opportune modifiche) per ottenere una forma canonica per gli stati non finali di $\mathcal{A}$, rendendo così possibile (assieme ai risultati precedenti) la costruzione di una forma canonica per gli automi fondamentalmente periodici che consenta di risolvere il problema dell'equivalenza.

Prima di introdurre la costruzione della forma canonica per gli stati non finali, ricordiamo brevemente i risultati teorici che consentono la minimizzazione degli automi regolari.

5.3.1 Minimizzazione degli automi regolari

Gli automi regolari sono automi a stati finiti che operano su parole finite. Introduciamo quindi la definizione di automa regolare e di linguaggio riconosciuto dagli automi regolari. Per una trattazione più approfondita degli automi regolari e delle loro proprietà, si veda [HU80].

Definizione 5.18 (Automa regolare). Un *automa regolare* sull'alfabeto A è una quadrupla $\mathcal{A} = (Q, \Delta, q_0, F)$ dove

- Q è un insieme finito di stati;
- $\Delta \subseteq Q \times A \times Q$ è la relazione di transizione;
- $q_0 \in Q$ è lo stato iniziale;
- $F \subseteq Q$ è l'insieme degli stati finali.

Una parola finita $v \in A^*$ è *riconosciuta dall'automa* se esiste un run di $\mathcal{A}$ su v che parte dallo stato iniziale q_0 e porta ad uno stato finale. Il linguaggio di $\mathcal{A}$ è l'insieme delle parole riconosciute da $\mathcal{A}$, ed è detto *linguaggio regolare*.

Gli automi regolari si dividono in automi *deterministici*, per i quali la relazione Δ è tale che per ogni coppia $(q, a) \in Q \times A$ esiste al più una transizione $(q, a, q') \in \Delta$; e automi *non deterministici* per i quali non vale questa limitazione. Si dimostra inoltre che le due classi di automi regolari (deterministici e non deterministici) riconoscono la stessa classe di linguaggi e che è possibile costruire a partire da un automa non deterministico un automa deterministico che riconosce lo stesso linguaggio.

Definiamo ora due relazioni di equivalenza tra le parole riconosciute da un automa regolare che ci permetteranno di definire le proprietà dell'automa regolare minimo per un linguaggio. Ricordiamo che una relazione di equivalenza $\sim$ tra parole finite si dice *invariante destra rispetto alla concatenazione* se per ogni $x, y \in A^*$, $x \sim y \Rightarrow \forall z \in A^*.xz \sim yz$, e si dice di *ordine finito* se l'insieme delle sue classi di equivalenza è finito.

Definizione 5.19. Dato un linguaggio $L \subseteq A^*$ si definisce la relazione $\sim_L$ su A^* nel modo seguente:

$$\forall x, y \in A^*.x \sim_L y \Leftrightarrow (\forall z \in A^*.xz \in L \Leftrightarrow yx \in L).$$

La relazione $\sim_L$ è una relazione di equivalenza invariante destra.

Definizione 5.20. Dato un automa regolare deterministico $\mathcal{A} = (Q, \Delta, q_0, f)$ si definisce la relazione $\sim_{\mathcal{A}}$ su A^* nel modo seguente:

$$\forall x, y \in A^*.x \sim_{\mathcal{A}} y \Leftrightarrow (\exists q \in Q.q_0 \xrightarrow{x} q \text{ e } q_0 \xrightarrow{y} q).$$

La relazione $\sim_{\mathcal{A}}$ è una relazione di equivalenza invariante destra di indice finito.

Il seguente teorema pone le basi per la minimizzazione degli automi regolari deterministici.

Teorema 5.21 (Teorema di Myhill-Nerode). *Sia $L \subseteq A^*$ un linguaggio di parole finite. Allora le seguenti proposizioni sono equivalenti:*

- *L è un linguaggio regolare;*
- *L è esprimibile come unione di classi di equivalenza di una relazione di equivalenza invariante destra e di indice finito;*

- $\sim_L$ *ha indice finito.*

La dimostrazione del Teorema di Myhill-Nerode mostra alcune importanti proprietà delle relazioni $\sim_L$ e $\sim_{\mathcal{A}}$ definite in precedenza. In particolare si osserva che la relazione $\sim_{\mathcal{A}}$ definita dall'automa è un *raffinamento* della relazione $\sim_L$, e quindi che il numero di stati dell'automa $\mathcal{A}$ è sempre maggiore o uguale del numero di classi di equivalenza di $\sim_L$. Inoltre si mostra come sia possibile costruire un automa che riconosce L avente un numero di stati esattamente pari al numero di classi di equivalenza di $\sim_L$. Ciò porta alla definizione naturale di un automa minimo per gli automi regolari deterministici.

Teorema 5.22. *L'automa deterministico minimo (rispetto al numero degli stati) che accetta il linguaggio L è unico a meno di isomorfismi (ad esempio, rinomina degli stati).*

Questa costruzione è inoltre effettiva, ed esistono numerosi algoritmi che calcolano l'automa minimo a partire da un automa regolare dato. Hopcroft ed Ullman propongono in [HU80] un esempio di algoritmo per calcolare l'automa minimo, mentre Watson presenta in [Wat93] una classificazione dei principali algoritmi presenti in letteratura.

Osservazione 5.23. È importante osservare che vale l'unicità dell'automa minimo solamente per gli automi regolari deterministici, e che tutti gli algoritmi di minimizzazione forniscono come risultato un automa deterministico: nel caso in cui l'automa di input sia non deterministico viene quindi operato un passo preliminare di conversione dell'automa ad un automa deterministico che riconosce lo stesso linguaggio. Questa osservazione risulta importante nel contesto degli automi fondamentalmente periodici: in questo caso non vale l'equivalenza tra automi deterministici e non deterministici, e l'automa regolare che verrà minimizzato nel corso della procedura di costruzione della forma canonica sarà sempre (in generale) non deterministico.

5.3.2 Definizione dell'automa prefisso

Iniziamo ora la discussione sull'algoritmo di costruzione di una forma canonica per gli stati non finali degli automi fondamentalmente periodici presentando brevemente i passi principali della procedura:

1. a partire dall'automa fondamentalmente periodico $\mathcal{A}$ si definisce l'*automa prefisso*, che è un automa regolare su parole finite;

2. l'automa prefisso viene minimizzato, ottenendo un automa deterministico che è unico rispetto al linguaggio riconosciuto;

3. a partire dall'automa prefisso minimo si costruisce un automa fondamentalmente periodico che riconosce lo stesso linguaggio di $\mathcal{A}$, ma la cui forma è unica rispetto al linguaggio riconosciuto.

L'automa prefisso, introdotto al punto 1, deve avere le seguenti caratteristiche: deve ricalcare la struttura degli stati non finali di $\mathcal{A}$ (riconoscendone il linguaggio dei prefissi) e deve permettere di ricostruire l'automa fondamentalmente periodico dopo la sua minimizzazione (mantenendo informazione sull'accoppiamento prefisso-periodo delle parole riconosciute).

L'automa $\mathcal{A}^{pre}$ definito nella Sezione 5.2.1 per dimostrare la regolarità del linguaggio dei prefissi di $\mathcal{A}$ rispetta la prima condizione ma non la seconda: la sua struttura ricalca infatti quella dell'automa fondamentalmente periodico $\mathcal{A}$, ma essa non consente (dopo essere stata minimizzata) di ricostruire l'automa fondamentalmente periodico. Infatti, poiché i suoi stati finali sono stati "pozzo" (cioè stati senza transizioni uscenti) l'automa minimo da esso ricavato possiede un unico stato finale su cui confluiscono tutte le parole riconosciute, perdendo così memoria dell'accoppiamento tra i prefissi ed i periodi.

Ricordando che gli stati finali di $\mathcal{A}^{pre}$ sono gli stessi stati finali di $\mathcal{A}$, la struttura dell'automa viene modificata in modo da mantenere traccia del periodo associato ad ogni prefisso nel modo seguente:

- un nuovo stato viene aggiunto all'automa, e tale stato diventerà l'unico stato finale dell'automa prefisso;
- per ogni stato finale $f \in F$ di $\mathcal{A}$, viene aggiunta una transizione uscente etichettata con f che porta nel nuovo stato finale.

La definizione di automa prefisso è quindi la seguente.

Definizione 5.24 (Automa prefisso). Se $\mathcal{A} = (Q, \Delta, q_0, F)$ è un automa fondamentalmente periodico in forma normale, l'automa prefisso $pre(\mathcal{A}) = (Q^P, \Delta^P, q_0^P, F^P)$ è definito nel modo seguente:

- l'alfabeto su cui opera è $A \cup F$, con $F \cap A = \emptyset$;
- $Q^P = Q \cup \{f_0\}$, con $f_0 \notin Q$;
- $\Delta^P = (\Delta \setminus \{(f, a, f') \in \Delta | f, f' \in F\}) \cup \{(f, f, f_0) | f \in F\}$;
- $q_0^P = q_0$;
- $F^P = \{f_0\}$.

Nell'automa prefisso l'informazione sul periodo associato ad ogni prefisso è quindi incastonata nel linguaggio: ogni parola riconosciuta da $pre(\mathcal{A})$ ha come ultimo carattere lo stato finale da cui si inizia a riconoscere il periodo. Questa proprietà fa sì che sia possibile operare sull'automa $pre(\mathcal{A})$ e (se il linguaggio riconosciuto rimane invariato) ricostruire successivamente un automa fondamentalmente periodico che riconosce lo stesso linguaggio di partenza. Il seguente lemma presenta le caratteristiche del linguaggio riconosciuto dall'automa prefisso, e i suoi legami con il linguaggio riconosciuto da $\mathcal{A}$.

Proposizione 5.25. *Sia $\mathcal{A} = (Q, \Delta, q_0, F)$ un automa fondamentalmente periodico in forma normale e sia $pre(\mathcal{A})$ il suo automa prefisso. Allora il linguaggio riconosciuto da $pre(\mathcal{A})$ è tale che:*

1. *ogni parola $v \in L(pre(\mathcal{A}))$ è nella forma $u \cdot q_f$, con $u \in A^*$ e $q_f \in F$;*
2. *la parola $u \cdot q_f$ appartiene a $L(pre(\mathcal{A}))$ se e solo se $\mathcal{A}$ è tale che $q_0 \overset{u}{\rightsquigarrow} q_f$.*

Dimostrazione. Sia $\mathcal{A} = (Q, \Delta, q_0, F)$ un automa fondamentalmente periodico in forma normale e sia $pre(\mathcal{A}) = (Q^P, \Delta^P, q_0, \{f_0\})$ il suo automa prefisso.

Per la dimostrazione del punto *(1)* del Lemma è sufficiente osservare che le uniche transizioni di $pre(\mathcal{A})$ etichettate con gli stati finali di $\mathcal{A}$ sono quelle entranti nell'unico stato finale f_0, mentre tutte le altre transizioni sono etichettate con simboli di A. Quindi ogni run accettante di $pre(\mathcal{A})$ è composto da una parte iniziale di transizioni etichettate con simboli di A, ed un'ultima transizione

etichettata con uno stato di F. Le parole accettate dall'automa prefisso sono quindi necessariamente nella forma $u \cdot q_f$, con $u \in A^*$ e $q_f \in F$.

Dimostriamo ora il punto *(2)* del Lemma. Sia $u \cdot q_f \in L(pre(\mathcal{A}))$ e sia σ un run accettante di $pre(\mathcal{A})$ su $u \cdot q_f$. Poiché f_0 è l'unico stato finale dell'automa prefisso, e poiché l'unica transizione entrante in f_0 ed etichettata con q_f proviene dallo stesso stato q_f, la struttura del run è la seguente:

$$\sigma = q_0 \xrightarrow{u} q_f \xrightarrow{q_f} f_0;$$

inoltre il segmento di run da q_0 a q_f passa solamente per stati appartenenti a $Q \setminus F$, perché non esistono transizioni etichettate con simboli di A uscenti dagli stati di F. La struttura degli stati di $pre(\mathcal{A})$ ricalca quella degli stati di $\mathcal{A}$ (con l'eccezione dello stato finale f_0), quindi il run $q_0 \xrightarrow{u} q_f$ è un segmento di run valido anche per $\mathcal{A}$ e vale l'implicazione $u \cdot q_f \in L(pre(\mathcal{A})) \Rightarrow q_0 \underset{\mathcal{A}}{\overset{u}{\rightsquigarrow}} q_f$.

Sia ora $u \in A^*$ tale che $q_0 \underset{\mathcal{A}}{\overset{u}{\rightsquigarrow}} q_f$, con $q_f \in F$. Come abbiamo già osservato in precedenza, la struttura di $pre(\mathcal{A})$ ricalca quella di $\mathcal{A}$ e quindi anche per $pre(\mathcal{A})$ vale che $q_0 \xrightarrow{u} q_f$. Poiché la transizione (q_f, q_f, f_0) appartiene a Δ^P è possibile costruire il seguente run accettante di $pre(\mathcal{A})$:

$$\sigma = q_0 \xrightarrow{u} q_f \xrightarrow{q_f} f_0.$$

Quindi $u \cdot q_f \in L(pre(\mathcal{A}))$ ed abbiamo dimostrato che vale anche l'implicazione inversa $q_0 \underset{\mathcal{A}}{\overset{u}{\rightsquigarrow}} q_f \Rightarrow u \cdot q_f \in L(pre(\mathcal{A}))$. □

5.3.3 Ricostruzione dell'automa fondamentalmente periodico

In questa sezione mostriamo come sia possibile, dato un automa regolare che riconosce lo stesso linguaggio di $pre(\mathcal{A})$, costruire un automa fondamentalmente periodico che riconosce lo stesso linguaggio di $\mathcal{A}$, mediante il seguente algoritmo.

Algoritmo 5.3 Ricostruzione dell'automa fondamentalmente periodico

1: **let** $\mathcal{A} = (Q, \Delta, q_0, F)$
2: **let** $\mathcal{P} = (Q^P, \Delta^P, q_0^P, F^P)$
3: **let** $\mathcal{A}' = (Q', \Delta', q_0', F')$
4: $Q' \leftarrow Q^P \cup F$ {Supponendo che $Q^P \cap F = \emptyset$}
5: $\Delta' \leftarrow \Delta^P \cup \{(f, a, f') \in \Delta | f, f' \in F\}$
6: **for all** $f \in F^P$ **do**
7: **for all** $(q, q_f, f) \in \Delta'$ tali che $q_f \in F$ **do**
8: **for all** $(q', a', q) \in \Delta'$ tali che $a \in A$ **do**
9: *insert transition* (q', a', q_f) *in* Δ'
10: **end for**
11: *delete transition* (q, q_f, f)
12: **end for**
13: **end for**
14: $F' \leftarrow F$
15: $q_0' \leftarrow q_0^P$
16: $clear(\mathcal{A}')$

Sia quindi $\mathcal{A} = (Q, \Delta, q_0, F)$ un automa fondamentalmente periodico in forma normale, e sia $\mathcal{P} = (Q^P, \Delta^P, q_0^P, F^P)$ un automa regolare tale che $L(\mathcal{P}) = L(pre(\mathcal{A}))$. L'Algoritmo 5.3 costruisce un automa fondamentalmente periodico $\mathcal{A}' = (Q', \Delta', q_0', F')$ che riconosce lo stesso linguaggio di $\mathcal{A}$, come dimostrato dal seguente lemma.

Teorema 5.26. *Sia $\mathcal{A} = (Q, \Delta, q_0, F)$ un automa fondamentalmente periodico in forma normale, e sia $\mathcal{P} = (Q^P, \Delta^P, q_0^P, F^P)$ un automa regolare tale che $L(\mathcal{P}) = L(pre(\mathcal{A}))$. Allora, al termine dell'esecuzione dell'Algoritmo 5.3, l'automa $\mathcal{A}' = (Q', \Delta', q_0', F')$ è un automa fondamentalmente periodico in forma normale che riconosce lo stesso linguaggio di $\mathcal{A}$.*

Dimostrazione. L'Algoritmo 5.3 opera nel modo seguente:

1. pone come insieme degli stati di $\mathcal{A}'$ l'unione dell'insieme degli stati di $\mathcal{P}$ con l'insieme degli stati finali di $\mathcal{A}$ (riga 4);
2. ricostruisce la struttura dei cicli finali di $\mathcal{A}$ e la struttura di $\mathcal{P}$ in $\mathcal{A}'$ (riga 5);
3. per ogni stato finale f di $\mathcal{P}$ l'algoritmo elimina le transizioni entranti etichettate con uno stato finale di $\mathcal{A}$, ed introduce delle opportune transizioni che portano ai cicli finali (righe 6-13);
4. gli stati finali di $\mathcal{A}$ diventano stati finali anche per $\mathcal{A}'$ (riga 14);
5. lo stato iniziale di $\mathcal{P}$ è lo stato iniziale anche di $\mathcal{A}'$ (riga 15).

Al termine dell'algoritmo l'automa $\mathcal{A}'$ è quindi un automa fondamentalmente periodico in forma normale perché ogni stato finale appartiene ad un solo ciclo (di soli stati finali) senza transizioni uscenti, che riconosce un unico periodo; inoltre $q_0' \notin F'$.

Mostriamo ora che $L(\mathcal{A}) = L(\mathcal{A}')$. Sia quindi $w \in L(\mathcal{A})$ e sia ρ un run accettante di $\mathcal{A}$ su w. Se q_f è il primo stato finale che occorre in ρ, la struttura del run è la seguente:

$$\rho = q_0 \xrightarrow{u} q_f \xrightarrow{v} q_f \xrightarrow{v} \dots$$

Per la Proposizione 5.25 la parola $u \cdot q_f \in L(pre(\mathcal{A})) = L(\mathcal{P})$ e, se $u = u' \cdot a$, con $a \in A$, esiste un opportuno run σ di $\mathcal{P}$ così composto:

$$\sigma = q_0^P \xrightarrow{u'} q' \xrightarrow{a} q \xrightarrow{q_f} f, \text{ con } f \in F^P.$$

Tutte le transizioni di σ sono presenti in $\mathcal{A}'$, con l'eccezione di (q, q_f, f) che è stata eliminata dall'algoritmo per inserire la transizione (q', a, q_f), rendendo possibile la costruzione del seguente run accettante ρ' di $\mathcal{A}'$ su w:

$$\rho' = q_0' \xrightarrow{u'} q' \xrightarrow{a} q_f \xrightarrow{v} q_f \xrightarrow{v} \dots$$

Quindi $w \in L(\mathcal{A}')$ e $L(\mathcal{A}) \subseteq L(\mathcal{A}')$.

Sia ora $w \in L(\mathcal{A}')$, e sia ρ' un run accettante di $\mathcal{A}'$ su w. Se q_f è il primo stato finale che occorre in ρ', la struttura del run è la seguente:

$$\rho' = q_0' \xrightarrow{u} q' \xrightarrow{a} q_f \xrightarrow{v} q_f \xrightarrow{v} \dots$$

La transizione (q', a, q_f) è stata aggiunta dall'algoritmo in una delle iterazioni del ciclo 3-10, ed esistono quindi in $\mathcal{P}$ le transizioni (q', a, q) e (q, q_f, f), con $f \in F^P$. Di conseguenza la parola $u \cdot a \cdot q_f \in L(\mathcal{P}) = L(pre(\mathcal{A}))$ e, per la Proposizione 5.25, l'automa $\mathcal{A}$ è tale che $q_0 \xrightarrow{u \cdot a} q_f$. Quindi è possibile costruire il seguente run accettante ρ di $\mathcal{A}$ su w:

$$\rho = q_0 \xrightarrow{u \cdot a} q_f \xrightarrow{v} q_f \xrightarrow{v} \dots$$

Vale quindi anche l'inclusione $L(\mathcal{A}') \subseteq L(\mathcal{A})$ e perciò $L(\mathcal{A}) = L(\mathcal{A}')$. □

5.3.4 Algoritmo di costruzione della forma canonica

La definizione data di automa prefisso e gli Algoritmi 5.1, 5.2 e 5.3 consentono di costruire, a partire da un automa fondamentalmente periodico in forma normale $\mathcal{A}$, un automa fondamentalmente periodico $\mathcal{A}'$ in forma canonica attraverso la procedura presentata nell'Algoritmo 5.4.

Algoritmo 5.4 Costruzione della forma canonica

1: Esecuzione dell'Algoritmo 5.1 su $\mathcal{A}$ per minimizzare la struttura degli stati finali;
2: minimizzazione del linguaggio dei prefissi di $\mathcal{A}$ tramite l'Algoritmo 5.2;
3: costruzione dell'automa prefisso $pre(\mathcal{A})$;
4: minimizzazione di $pre(\mathcal{A})$;
5: costruzione dell'automa fondamentalmente periodico $\mathcal{A}'$ in forma canonica mediante l'Algoritmo 5.3.

Tutti i passi presentati sono effettivi e mantengono il linguaggio riconosciuto dall'automa (come dimostrato dai Teoremi 5.13 e 5.26 e dal Lemma 5.15). Dimostriamo quindi che l'automa costruito dall'algoritmo è in forma canonica, cioè che gli automi ottenuti dopo aver eseguito i cinque passi previsti su due automi fondamentalmente periodici che riconoscono lo stesso linguaggio sono identici (a meno di isomorfismi).

Il Teorema 5.13 ed il Corollario 5.17 stabiliscono che, dopo aver eseguito i primi due passi dell'algoritmo, due automi che riconoscono lo stesso linguaggio possiedono lo stesso numero e la stessa struttura degli stati finali e riconoscono gli stessi linguaggi dei prefissi e dei periodi. Poiché la struttura degli stati finali dei due automi è identica, anche i loro automi prefisso riconoscono lo stesso linguaggio (a meno di rinomina degli stati), come dimostrato dal seguente lemma.

Lemma 5.27. *Siano $\mathcal{A} = (Q, \Delta, q_0, F)$ e $\mathcal{A}' = (Q', \Delta', q_0', F')$ due automi fondamentalmente periodici in forma normale tali che $L(\mathcal{A}) = L(\mathcal{A}')$. Dopo aver eseguito i primi due passi dell'Algoritmo 5.4 su entrambi esiste una opportuna rinomina degli stati finali di $\mathcal{A}'$ che fa sì che gli automi prefisso $pre(\mathcal{A})$ e $pre(\mathcal{A}')$ riconoscano lo stesso linguaggio.*

Dimostrazione. Dopo aver eseguito gli Algoritmi 5.1 e 5.2 sui due automi fondamentalmente periodici $\mathcal{A}$ e $\mathcal{A}'$ valgono le seguenti proprietà:

1. $\mathcal{A}$ e $\mathcal{A}'$ riconoscono lo stesso linguaggio dei prefissi, ed esso è minimale;

2. $\mathcal{A}$ e $\mathcal{A}'$ riconoscono lo stesso linguaggio dei periodi, ed esso è minimale;
3. $\mathcal{A}$ e $\mathcal{A}'$ possiedono lo stesso numero di stati finali, ed ogni stato finale riconosce un periodo diverso da tutti gli altri stati finali.

Esiste quindi una biezione tra gli stati finali degli automi ed i periodi riconosciuti: ciò consente di rinominare gli stati finali di $\mathcal{A}'$ in modo che $\mathcal{A}'$ possieda gli stessi stati finali di $\mathcal{A}$, ed in modo che a partire dallo stesso stato finale il periodo riconosciuto in $\mathcal{A}$ e in $\mathcal{A}'$ sia lo stesso.

Dopo aver rinominato gli stati di $\mathcal{A}'$ gli automi $pre(\mathcal{A})$ e $pre(\mathcal{A}')$ operano sullo stesso alfabeto, ed i loro linguaggi possono essere confrontati.

Sia quindi $u \cdot q_f \in L(pre(\mathcal{A}))$: per la Proposizione 5.25 esiste una parola fondamentalmente periodica $w \in L(\mathcal{A})$ tale che $w = u \cdot v^\omega$ ed esiste un run accettante ρ di $\mathcal{A}$ su w che ha la seguente struttura:

$$\rho = q_0 \xrightarrow{u} q_f \xrightarrow{v} q_f \xrightarrow{v} \dots$$

per i Teoremi 5.13 e 5.16 il prefisso u ed il periodo v sono minimali, quindi in $\mathcal{A}'$ esiste un run ρ' tale che:

$$\rho' = q'_0 \xrightarrow{u} q_f \xrightarrow{v} q_f \dots$$

allora $u \cdot q_f \in L(pre(\mathcal{A}'))$ e $L(pre(\mathcal{A})) \subseteq L(pre(\mathcal{A}'))$. Ragionando in modo analogo si dimostra che vale anche l'inclusione $L(pre(\mathcal{A}')) \subseteq L(pre(\mathcal{A}))$, quindi $L(pre(\mathcal{A})) = L(pre(\mathcal{A}'))$. □

Questo risultato preliminare consente di dimostrare che dopo aver eseguito l'Algoritmo 5.4 su due automi fondamentalmente periodici che riconoscono lo stesso linguaggio la loro struttura è identica (a meno di isomorfismi).

Teorema 5.28. *Siano $\mathcal{A} = (Q, \Delta, q_0, F)$ e $\mathcal{A}' = (Q', \Delta', q'_0, F')$ due automi fondamentalmente periodici in forma normale tali che $L(\mathcal{A}) = L(\mathcal{A}')$. Dopo aver eseguito l'Algoritmo 5.4 su entrambi la struttura dei due automi è identica a meno di isomorfismi.*

Dimostrazione. Abbiamo già stabilito, per il Lemma 5.27, che dopo l'esecuzione dei primi due passi dell'algoritmo gli automi $\mathcal{A}$ e $\mathcal{A}'$ sono tali che:

- riconoscono gli stessi linguaggi dei prefissi e dei periodi (e tali linguaggi sono minimali);
- possiedono la stessa struttura degli stati finali;
- $pre(\mathcal{A})$ e $pre(\mathcal{A}')$ riconoscono lo stesso linguaggio (a meno di rinomina degli stati finali).

Quindi, dopo aver minimizzato $pre(\mathcal{A})$ e $pre(\mathcal{A}')$ (passo 4), si ottiene lo stesso automa deterministico, perché per gli automi regolari l'automa minimo è unico. Il passo 5, che ricollega la struttura degli stati finali all'automa prefisso minimo costruisce quindi lo stesso automa fondamentalmente periodico sia per $\mathcal{A}$ che per $\mathcal{A}'$.

Abbiamo dunque dimostrato che l'Algoritmo 5.4 costruisce una forma canonica per gli automi fondamentalmente periodici. □

Osservazione 5.29. L'Algoritmo 5.4 costruisce una forma canonica per gli automi fondamentalmente periodici che non è minimale perché la procedura di minimizzazione degli automi regolari genera un automa deterministico a partire da un automa che è (in generale) non deterministico, con un possibile aumento esponenziale del numero degli stati.

Il Teorema 5.28 appena dimostrato consente di risolvere il problema dell'equivalenza per gli automi fondamentalmente periodici.

Corollario 5.30. *Il problema dell'equivalenza per gli automi fondamentalmente periodici è decidibile.*

Dimostrazione. Siano $\mathcal{A}$ e $\mathcal{A}'$ due automi fondamentalmente periodici. Allora, per il Teorema 5.28, $L(\mathcal{A}) = L(\mathcal{A}')$ se e solo se, dopo aver eseguito l'Algoritmo 5.4 su entrambi, le strutture di $\mathcal{A}$ e $\mathcal{A}'$ sono identiche (a meno di isomorfismi). □

Conclusioni e sviluppi futuri

In questa tesi si è affrontato il problema della rappresentazione e della gestione delle rappresentazioni temporali. Inizialmente si è fornita una definizione formale del termine *granularità* e si è presentato il concetto di sistema di granularità, attraverso la definizione delle relazioni che possono intercorrere tra diverse granularità temporali. In seguito, sono stati studiati alcuni formalismi per rappresentare e manipolare le strutture temporali, a partire dalla Calendar Algebra, di cui si sono definiti i principali operatori e di cui si è studiata l'espressività.

Nel Capitolo 2 il formalismo delle Granspec proposto da Wijsen è stato ripreso ed adattato a rappresentare le granularità etichettate, confrontando la sua espressività con quella della Calendar Algebra. Dopo aver stabilito che questo formalismo, basato su stringhe, è espressivo almeno quanto la Calendar Algebra, si è proposta una soluzione al problema dell'equivalenza tra rappresentazioni, fornendo un algoritmo che determina la codifica minima di una parola fondamentalmente periodica. Successivamente, si sono studiati i formalismi basati su automi proposti da Dal Lago, Montanari e Puppis, stabilendone l'espressività e proponendo alcuni algoritmi per la soluzione del problema dell'equivalenza. I principali vantaggi degli approcci basati su automi sono:

- la possibilità di applicare i risultati della teoria degli automi per operare sulle granularità (come, per esempio, le *proprietà di chiusura rispetto alle operazioni sui linguaggi*, o le soluzioni ai problemi della *minimizzazione* o dell'*equivalenza*);
- la dimensione ridotta delle rappresentazioni (in particolare, nel caso degli automi estesi riducibili e degli automi etichettati ristretti).

Il principale limite dei formalismi proposti nel Capitolo 3 è la loro incapacità di rappresentare più di una granularità per automa. Nel Capitolo 4 la teoria degli automi di Büchi viene sfruttata per rappresentare *insiemi di granularità*. Dopo aver determinato e caratterizzato i linguaggi di parole fondamentalmente periodiche che possono essere riconosciuti dagli automi ω-regolari, si è definita la classe degli *automi di Büchi fondamentalmente periodici* che riconosce esattamente tali linguaggi e se ne sono studiate le proprietà effettive di chiusura rispetto alle principali operazioni sui linguaggi. Nell'ultimo capitolo si sono studiati alcuni problemi paradigmatici della teoria degli automi, mostrandone alcune possibili applicazioni nella gestione delle granularità temporali, ed uti-

lizzando alcuni risultati di decidibilità degli automi di Büchi per fornire una soluzione ad alcuni di essi. Nel caso del problema dell'equivalenza tra automi fondamentalmente periodici non è, però, possibile utilizzare l'approccio al problema dell'equivalenza utilizzato nel caso degli automi di Büchi e si è dovuta, quindi, sviluppare una strategia alternativa. Sono stati presentati alcuni algoritmi che permettono, mediante passaggi successivi, di costruire una *forma canonica* per gli automi fondamentalmente periodici, ossia di arrivare ad un automa la cui struttura è unica rispetto al linguaggio riconosciuto. Il problema dell'equivalenza tra due automi fondamentalmente periodici diventa, quindi, decidibile semplicemente confrontando le forme canoniche dei due automi di partenza.

I risultati ottenuti permettono di affermare che i formalismi basati su automi consentono di rappresentare, oltre a singole granularità temporali, anche *insiemi di granularità*, risultando quindi propriamente più espressivi della Calendar Algebra e del formalismo delle Granspec. Le proprietà di chiusura e di decidibilità degli automi fondamentalmente periodici mostrano, inoltre, come questo formalismo permetta di costruire effettivamente rappresentazioni di insiemi di granularità mediante la composizione di elementi più semplici e di studiare proprietà interessanti delle granularità rappresentate.

Gli automi di Büchi fondamentalmente periodici (o alcune loro eventuali estensioni, che potrebbero permettere di rappresentare insiemi di granularità più ampi) costituiscono quindi un formalismo semplice ed efficace per gestire insiemi di granularità temporali.

Operazioni su insiemi di granularità

Nel Capitolo 3 si è accennato alla possibilità di utilizzare gli automi single-string (e le loro estensioni) per implementare le operazioni di conversione tra granularità e, in particolare, gli operatori della Calendar Algebra, mediante gli algoritmi proposti da Dal Lago, Montanari e Puppis in [DMP03a]. Queste operazioni sono definite su singole granularità (o su coppie di granularità) e hanno pertanto una controparte naturale nel contesto degli automi single-string, che sono in grado di rappresentare solamente singole parole fondamentalmente periodiche. Operando, invece, con insiemi di granularità, le espressioni della Calendar Algebra e, più in generale, le operazioni di conversione risultano difficili da interpretare. Estendere la definizione di tali operazioni a insiemi di granularità è sicuramente un problema interessante, cui si accompagna il problema di codificare tali operazioni nel formalismo impiegato per rappresentare gli insiemi.

Si considerino, ad esempio, le funzioni di conversione $\lceil\ \rceil_G^H$ e $\lfloor\ \rfloor_G^H$. Se $\mathcal{G}$ è un insieme di coppie di granularità (G, H) tali che $G \trianglelefteq H$ (rappresentato, ad esempio, dal linguaggio di un opportuno automa fondamentalmente periodico), gli operatori di conversione possono essere reinterpretati in senso esistenziale o universale, qualora si presenti la necessità di stabilire se, per esempio:

- esiste una coppia di granularità $(G, H) \in \mathcal{G}$ tale che $\lceil i \rceil_G^H = z$, oppure
- per ogni coppia di granularità $(G, H) \in \mathcal{G}$ vale l'uguaglianza $\lfloor z \rfloor_G^H = (i, n)$;

con i, z, n parametri del problema. Una possibile soluzione potrebbe essere quella di costruire le rappresentazioni degli insiemi $\mathcal{G}_1 = \{(G, H) \in \mathcal{G} | \lceil i \rceil_G^H = z\}$

e $\mathcal{G}_2 = \{(G, H) \in \mathcal{G} | \lfloor z \rfloor_G^H = (i, n)\}$, e di verificare se $\mathcal{G}_1 \neq \emptyset$, nel primo caso, o $\mathcal{G}_2 = \mathcal{G}$, nel secondo.

Per quanto riguarda le espressioni della Calendar Algebra esiste, invece, una naturale estensione agli insiemi di granularità. Ad esempio, se $\mathcal{G}$ e $\mathcal{H}$ sono insiemi di granularità, gli operatori *Group*, *AlterTick*, *Combine* e *SelectDown* possono essere interpretati su di essi nel modo seguente:

- $Group_m(\mathcal{G}) = \{Group_m(G) | G \in \mathcal{G}\}$;
- $AlterTick_{l,k}^m(\mathcal{G}, \mathcal{H}) = \{AlterTick_{l,k}^m(G, H) | G \in \mathcal{G} \text{ e } H \in \mathcal{H}\}$;
- $Combine(\mathcal{G}, \mathcal{H}) = \{Combine(G, H) | G \in \mathcal{G} \text{ e } H \in \mathcal{H}\}$;
- $SelectDown_k^l(\mathcal{G}, \mathcal{H}) = \{SelectDown_k^l(G, H) | G \in \mathcal{G} \text{ e } H \in \mathcal{H}\}$.

In verità, poiché alcuni operatori della Calendar Algebra sono definiti solamente se valgono precise relazioni tra gli operandi, la loro estensione deve verificare se tutte le granularità degli insiemi coinvolti le soddisfano. Tale problema è, in generale, indecidibile. È necessario, quindi, imporre opportuni vincoli sugli insiemi di granularità utilizzabili, mediante opportune restrizioni sulle definizioni dei nuovi operatori (ad esempio, nel caso di operatori binari, si può imporre che un operando sia una singola granularità e non un insieme), oppure mediante restrizioni sintattiche sulla composizione dei termini (in modo analogo a quanto proposto da Ning, Jajodia e Wang per la Calendar Algebra).

Dopo aver definito in maniera coerente le estensioni degli operatori di conversione o dei termini della Calendar Algebra a insiemi di granularità, occorre verificare che il formalismo adottato per rappresentare tali insiemi consenta di implementare le nuove operazioni. In particolare, occorre verificare che il formalismo sia *chiuso* rispetto a tali operazioni e che esse siano *effettivamente (ed efficacemente) computabili* mediante opportuni algoritmi.

La definizione di nuove operazioni su insiemi di granularità e la loro implementazione mediante appositi algoritmi è quindi un campo aperto a sviluppi interessanti, che potrebbero portare sia all'allargamento dell'ambito di utilizzo dei formalismi esistenti (quali gli automi fondamentalmente periodici), sia alla definizione di nuovi formalismi in grado di superare le limitazioni di quelli attuali.

Estensione della classe di automi

In questa tesi si è fornita una caratterizzazione dei linguaggi di parole fondamentalmente periodiche riconoscibili dagli automi di Büchi, osservando come essi possano essere composti da un numero infinito di prefissi, ma solamente da un numero finito di periodi distinti. La classe degli automi di Büchi fondamentalmente periodici riconosce esattamente tali linguaggi e soffre quindi di tale limitazione sul numero di periodi distinti rappresentabili. Di conseguenza, qualora si presenti la necessità di rappresentare insiemi di parole fondamentalmente periodiche con un *numero infinito di periodi distinti*, i formalismi basati sugli automi di Büchi risultano non essere sufficientemente espressivi. Si pone quindi il problema di determinare un formalismo, strettamente più espressivo degli automi fondamentalmente periodici, che consenta di rappresentare insiemi di granularità non rappresentabili tramite automi di Büchi, preservando la semplicità e l'efficenza dell'attuale soluzione.

Un possibile approfondimento degli argomenti della tesi potrebbe dunque essere l'estensione dello studio delle proprietà dei linguaggi ω-regolari, descritto nel Capitolo 4, a classi più espressive degli automi di Büchi (ad esempio, gli automi Push-down) per determinare quali insiemi di parole fondamentalmente periodiche possano rappresentare, stabilendo così quale sia la minima classe (nella gerarchia delle classi di automi) che consenta di riconoscere insiemi di parole fondamentalmente periodiche con un numero infinito di periodi distinti.

Determinata tale classe, risulterebbe interessante confrontarla con gli automi di Büchi e gli automi fondamentalmente periodici rispetto alla decidibilità dei problemi paradigmatici (quale, ad esempio, il problema dell'equivalenza) e rispetto alla complessità degli algoritmi che operano su di essa.

La ricerca di un formalismo più espressivo degli automi fondamentalmente periodici potrebbe portare alla definizione di una nuova classe di automi, che consenta di riconoscere linguaggi più ampi, mantenendo tuttavia i risultati di decidibilità e di efficenza degli automi fondamentalmente periodici. Si potrebbe, per esempio, estendere la definizione di automa etichettato ristretto (inserendo, per esempio, transizioni non deterministiche) per permetterle di riconoscere insiemi di parole fondamentalmente periodiche, mantenendo tuttavia le proprietà di compattezza date dalla struttura basata sui contatori. Un'altra possibilità è quella di definire una classe di automi che "memorizzi" in una opportuna struttura dati (ad esempio, una coda) il periodo della parola da riconoscere, per poter poi ripeterlo infinite volte.

Altri approcci al problema

Presentiamo, infine, altri due possibili formalismi per la rappresentazione di insiemi di granularità, che sfruttano strategie diverse, ma che sono comunque correlati con i risultati presentati in questa tesi.

Rappresentazione mediante formule logiche

In [CFP03], Combi, Franceschet e Peron presentano un approccio alla rappresentazione e manipolazione di granularità temporali basato sulla logica. In particolare, gli autori identificano le granularità con sequenze infinite di istanti temporali discreti, etichettati con opportuni simboli proposizionali che stabiliscono i punti iniziali e finali dei singoli granuli, e modellano gli insiemi di granularità e le loro relazioni tramite formule della logica temporale lineare. Questo formalismo permette di risolvere algoritmicamente i problemi della consistenza, equivalenza e classificazione degli insiemi rappresentati riducendoli al problema della validità delle formule logiche considerate, fornendo un modo semplice ed uniforme per manipolare le granularità.

Si noti come questo formalismo presenti un collegamento diretto con i formalismi basati su automi: gli automi di Büchi possono infatti venire utilizzati per rappresentare i modelli delle formule logiche considerate (cfr. [Tho90]), fornendo un metodo per rappresentare mediante automi gli insiemi di granularità coinvolti. Tale prospettiva presenta, però, alcune complicazioni:

- *la complessità degli algoritmi è elevata*, in quanto il problema della validità delle formule della logica considerata ha complessità temporale esponenziale;

- *gli insiemi rappresentati sono troppo ampi*, in quanto contengono anche parole non fondamentalmente periodiche.

Queste osservazioni suggeriscono di cercare formalismi alternativi per rappresentare insiemi di granularità mediante formule logiche, che consentano soluzioni algoritmiche più efficienti. Alla luce dei risultati presentati in questa tesi, la definizione di un collegamento diretto tra il formalismo logico e gli automi fondamentalmente periodici potrebbe portare alla costruzione di un sistema per la manipolazione e la rappresentazione di (insiemi di) granularità che presenti i vantaggi di entrambi gli approcci. Un tale sistema permetterebbe infatti di:

- *definire gli insiemi e le loro proprietà mediante formule*, in maniera più intuitiva e naturale rispetto alla costruzione diretta dell'automa;
- *manipolare gli insiemi mediante automi*, sfruttando gli algoritmi esistenti per tali automi.

La definizione del legame tra i due formalismi sicuramente non è banale, in quanto, in generale, si possono adottare due diverse strategie per arrivare ad un formalismo che integri la logica temporale e le rappresentazioni basate su automi:

- la definizione di una *nuova classe di automi*, che consenta di rappresentare esattamente gli insiemi di granularità catturati dal formalismo basato sulla logica, qualora gli automi fondamentalmente periodici si rivelino non sufficientemente espressivi;
- la determinazione di un opportuno *frammento della logica temporale* che catturi esattamente gli insiemi di granularità rappresentabili dagli automi fondamentalmente periodici.

Entrambe le strategie potrebbero condurre ad interessanti sviluppi degli argomenti presentati in questa tesi.

Rappresentazione mediante linguaggi di parole finite

In [CNP94a], Calbrix, Nivat e Podelsky introducono un metodo per rappresentare i linguaggi ω-regolari riconosciuti dagli automi di Büchi mediante linguaggi regolari di parole finite, riducendo così gli algoritmi sugli automi di Büchi ad algoritmi più semplici, che operano su automi regolari. Questo metodo può essere applicato, per esempio, alla logica S1S, ottenendo algoritmi più efficienti rispetto al metodo tradizionale, come mostrato in [CNP94b].

Gli autori partono dall'osservazione che l'insieme $\mathrm{UP}(L)$ delle parole fondamentalmente periodiche di L caratterizza il linguaggio ω-regolare L (cfr. Proposizione 4.4) e dall'osservazione che una parola fondamentalmente periodica $u \cdot v^\omega$ può essere rappresentata dalla parola finita $u \cdot \$ \cdot v$ (con \$ simbolo separatore non presente nell'alfabeto). Un linguaggio ω-regolare può, quindi, essere rappresentato dal linguaggio di parole finite $L_\$ = \{u \cdot \$ \cdot v | u \cdot v^\omega \in L\}$, e tale linguaggio risulta essere *regolare*, cioè riconoscibile da un automa a stati finiti che opera su parole finite. Vengono, inoltre, stabilite le condizioni per le quali un generico linguaggio regolare $K \subseteq A^* \cdot \$ \cdot A^+$ rappresenta un linguaggio ω-regolare.

Questa costruzione può essere facilmente adattata alla rappresentazione di insiemi di granularità temporali, osservando semplicemente che la parola $u \cdot \$ \cdot v$

rappresenta la codifica (u, v) della parola fondamentalmente periodica $u \cdot v^\omega$, e quindi che è possibile utilizzare gli automi regolari per rappresentare insiemi di (codifiche di) parole fondamentalmente periodiche. Tuttavia, tale approccio, basato su una teoria che nasce in un contesto diverso da quello delle granularità temporali, presenta alcuni problemi.

- Ai linguaggi regolari (di parole finite) che rappresentano insiemi di parole fondamentalmente periodiche non corrisponde una classe di automi. Gli algoritmi della teoria degli automi regolari vanno quindi adattati a questo formalismo, per garantire (dopo ogni operazione) che l'automa riconosca ancora un linguaggio di parole fondamentalmente periodiche.
- I linguaggi regolari che rappresentano linguaggi ω-regolari contengono sempre tutte le codifiche possibili di ogni parola fondamentalmente periodica: il formalismo è quindi ridondante e potrebbe generare problemi legati alla complessità degli algoritmi e alla dimensione delle rappresentazioni.

Il secondo problema potrebbe essere risolto mediante un approccio simile a quello usato nel Capitolo 3 per minimizzare le codifiche di parole fondamentalmente periodiche, passando così da insiemi che contengono tutte le codifiche possibili a insiemi che contengono solamente le *codifiche minime* delle parole fondamentalmente periodiche rappresentate. In questo caso il formalismo risulterebbe più compatto rispetto alla dimensione delle rappresentazioni e permetterebbe di sfruttare alcuni risultati interessanti della teoria degli automi regolari (come l'esistenza e l'unicità dell'automa minimo) per ottimizzare l'efficenza degli algoritmi che operano su di esso.

Bibliografia

[BDE+98] C. Bettini, C. E. Dyreson, W. S. Evans, R. T. Snodgrass, e X. S. Wang. A glossary of time granularity concepts. In O. Etzion, S. Jajodia, e S. Sripada, (A cura di), *Temporal Databases: Research and Practice*, volume 1399 di *Lecture Notes in Computer Science*, pp. 406–413. Springer-Verlag, 1998.

[BDS99] C. Bettini e R. De Sibi. Symbolic representation of user-defined time granularities. In *Proceedings of TIME99 6th International Workshop on Temporal Representation and Reasoning*, pp. 17–28. IEEE Computer Society, 1999.

[BJW00] C. Bettini, S. Jajodia, e S. X. Wang. *Time Granularities in Databases, Data Mining and Temporal Reasoning*. Springer-Verlag, Luglio 2000.

[BN96] M. Bielikova e P. Navrat. A declarative approach to calendrical calculations. Relazione tecnica, Department of Computer Science and Engineering, Slovak Technical University, Bratislava, Febbraio 1996.

[Boo80] K. S. Booth. Lexicographically least circular substrings. *Information Processing Letters*, 10(4,5):240–242, 1980.

[CFP03] C. Combi, M. Franceschet, e A. Peron. Representing and reasoning about temporal granularities. *Journal of Logic and Computation*, 2003. Sottomesso per la pubblicazione.

[CM00] L. Chittaro e A. Montanari. Temporal representation and reasoning in artificial intelligence: Issues and approaches. *Annals of Mathematics and Artificial Intelligence*, 28:47–106, 2000.

[CNP94a] H. Calbrix, M. Nivat, e A. Podelski. Ultimately periodic words of rational ω-languages. In *Mathematical Foundations of Programming Semantics, 9th International Conference*, volume 802 di *Lecture Notes in Computer Science*, pp. 554–566. Springer, 1994.

[CNP94b] H. Calbrix, M. Nivat, e A. Podelski. Une méthode de décision de la logique monadique du second ordre d'une fonction successeur. In *Comptes Rendus de l'Académie des Sciences, Série I: Mathematique*, volume 318. Gauthier-Villars, 1994.

[DM01] U. Dal Lago e A. Montanari. Calendars, time granularities, and automata. In *Proceedings of SSTD 2001 7th International Symposium on Spatial and Temporal Databases*, volume 2121 di *Lecture Notes in Computer Science*, pp. 279–298, Berlino, Germania, Luglio 2001. Springer.

[DMP03a] U. Dal Lago, A. Montanari, e G. Puppis. Time granularities, calendar algebra, and automata. Relazione Tecnica 4, Università degli Studi di Udine, Italia, Febbraio 2003.

[DMP03b] U. Dal Lago, A. Montanari, e G. Puppis. Towards compact and tractable automaton-based representations of time granularities. In *Proceedings of ICTCS 2003 Eight Italian Conference on Theoretical Computer Science*. Springer Verlag, 2003.

[FM02] M. Franceschet e A. Montanari. Branching within time: an expressively complete and elementarily decidable temporal logic for time granularity. *Research on Language and Computation*, 1(4), 2002. In corso di pubblicazione.

[Fra01] M. Franceschet. *Dividing and Conquering the Layered Land*. Tesi di Dottorato, Dipartimento di Matematica e Informatica, Università degli Studi di Udine, Italia, 2001.

[FW65] N. J. Fine e H. S. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16:109–114, 1965.

[HU80] J. E. Hopcroft e J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1980.

[KMP77] D. E. Knuth, J. H. Morris, e V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, Giugno 1977.

[LMF86] B. Leban, D. McDonald, e D. Forster. A representation for collections of temporal intervals. In T. Kehler e S. Rosenschein, (A cura di), *Proceedings of the 5th National Conference on Artificial Intelligence*, volume 1, pp. 367–371. Morgan Kaufmann, 1986.

[MPP99] A. Montanari, A. Peron, e A. Policriti. Theories of omega-layered metric temporal structures: Expressiveness and decidability. *Logic Journal of the IGPL*, 7(1):79–102, Gennaio 1999.

[NJW02] P. Ning, S. Jajodia, e X. S. Wang. An algebraic representation of calendars. *Annals of Mathematics and Artificial Intelligence*, 36(1-2):5–38, Settembre 2002.

[NS92] M. Niezette e J. Stevenne. An efficent symbolic representation of periodic time. In *Proceedings of the CIKM-92 International Conference on Information and Knowledge Management*, pp. 161–168. ACM Press, 1992.

[PTB85] R. Paige, R. E. Tarjan, e R. Bonic. A linear time solution to the single function coarsest partition problem. *Theoretical Computer Science*, 40(1):67–84, 1985.

[Shi81] Y. Shiloach. Fast canonization of circular strings. *Journal of algorithms*, 2:107–121, 1981.

[Tho90] W. Thomas. Automata on infinite objects. In J. van Leeuwen, (A cura di), *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pp. 133–192. Elsevier MIT Press, 1990.

[Wat93] B. W. Watson. A taxonomy of finite automata minimization algorithms. Computing Science Report 93/44, Department of Mathematics and Computing Science, Eindhoven University of Technology, Olanda, 1993.

[Wij00] J. Wijsen. A string-based model for infinite granularities. In C. Bettini e A. Montanari, (A cura di), *Proceedings of the AAAI Workshop on Spatial and Temporal Granularities*, pp. 9–16. AAAI Press, 2000.

[ZU96] R. J. Zhang e E. Unger. Calendar algebra. Technical Report 96-1, Department of Computing and Information Sciences, Kansas State University, Manhattan, 1996.

Indice delle figure

Indice degli algoritmi