Espressioni Regolari e Equivalenze con Automi

Consigli e Conversioni

Tutorato 2: Espressioni Regolari, equivalenze con Automi e conversioni

Gabriel Rovesti
Corso di Laurea in Informatica - Università degli Studi di Padova

Anno Accademico 2024-2025

Contents

1 Introduzione

Questo documento raccoglie metodologie, consigli pratici e approfondimenti teorici relativi alle espressioni regolari e alla loro equivalenza con gli automi a stati finiti. È un complemento alle lezioni e ai tutorati, pensato per aiutare gli studenti ad affrontare gli esercizi tipici di questa parte del corso.

Concetto chiave

Le espressioni regolari sono un modo dichiarativo per descrivere linguaggi regolari, equivalenti in potere espressivo agli automi a stati finiti (DFA e NFA). Comprendere le conversioni tra questi formalismi è fondamentale per lo studio della teoria dei linguaggi formali.

2 Espressioni Regolari: Fondamenti

2.1 Definizione e Sintassi

Le espressioni regolari sono costruite utilizzando:

- **Costanti di base**:
  - ε per la stringa vuota
  - $\emptyset$ per il linguaggio vuoto
  - $a, b, \ldots$ per i simboli $a, b, \ldots \in \Sigma$
- **Operatori**:
  - $+$ per l'unione
  - $\cdot$ per la concatenazione
  - $*$ per la chiusura di Kleene
- **Parentesi** per il raggruppamento: ( )

Suggerimento

Le regole di precedenza per le espressioni regolari sono:

1. La chiusura di Kleene ($*$) ha la precedenza più alta
2. La concatenazione ($\cdot$) ha precedenza intermedia
3. L'unione ($+$) ha la precedenza più bassa

Usa le parentesi quando hai dubbi sulla precedenza degli operatori.

2.2 Semantica

Se E è un'espressione regolare, allora $\mathcal{L}(E)$ è il linguaggio rappresentato da E. La definizione è induttiva:

Caso Base:

- $\mathcal{L}(\varepsilon) = \{\varepsilon\}$
- $\mathcal{L}(\emptyset) = \emptyset$
- $\mathcal{L}(a) = \{a\}$ per $a \in \Sigma$

Caso Induttivo:

- $\mathcal{L}(E + F) = \mathcal{L}(E) \cup \mathcal{L}(F)$
- $\mathcal{L}(E \cdot F) = \mathcal{L}(E) \cdot \mathcal{L}(F)$
- $\mathcal{L}(E^*) = \mathcal{L}(E)^*$
- $\mathcal{L}((E)) = \mathcal{L}(E)$

Errore comune

Un errore comune è confondere l'espressione $01^* + 10^*$ (che rappresenta stringhe che iniziano con 0 seguite da un numero arbitrario di 1, o stringhe che iniziano con 1 seguite da un numero arbitrario di 0) con $(01)^* + (10)^*$ (che rappresenta stringhe formate da ripetizioni di 01 o ripetizioni di 10).

3 Equivalenza tra Automi e Espressioni Regolari

Concetto chiave

I linguaggi regolari possono essere rappresentati equivalentemente da:

- Automi a stati finiti deterministici (DFA)
- Automi a stati finiti non deterministici (NFA)
- Espressioni regolari (RE)

Esiste una corrispondenza biunivoca tra questi formalismi, ovvero ogni linguaggio regolare può essere rappresentato in ciascuno di questi modi.

3.1 Da Espressioni Regolari a ε-NFA

Procedimento di risoluzione

Per convertire un'espressione regolare R in un ε-NFA A tale che $\mathcal{L}(A) = \mathcal{L}(R)$:

1. **Casi base**:
   - Per ε: un singolo stato iniziale e finale
   - Per $\emptyset$: un automa che non accetta alcuna stringa
   - Per un simbolo $a \in \Sigma$: due stati collegati da una transizione etichettata a
2. **Casi induttivi**:
   - Per $R_1 + R_2$: costruisci un nuovo stato iniziale con ε-transizioni verso gli stati iniziali degli automi per R_1 e R_2
   - Per $R_1 \cdot R_2$: collega gli stati finali dell'automa per R_1 agli stati iniziali dell'automa per R_2 con ε-transizioni
   - Per R^*: aggiungi un nuovo stato iniziale/finale e ε-transizioni appropriate per implementare il ciclo

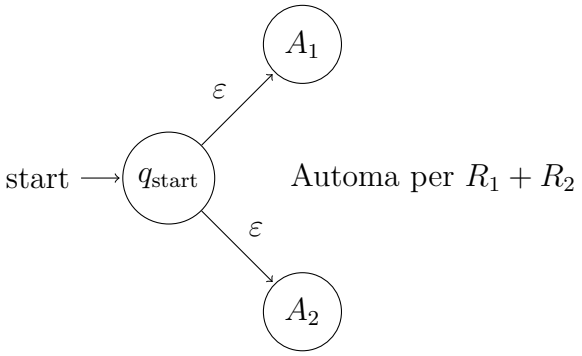


Figure 1: Schema di costruzione dell'automa per l'unione

Suggerimento

Quando costruisci l'automa per R^*, ricorda di aggiungere ε-transizioni che permettano di:

- Saltare direttamente all'accettazione (per rappresentare ε)
- Ripetere la costruzione per R zero o più volte

3.2 Da NFA a Espressioni Regolari

La conversione da NFA a espressioni regolari è più complessa e si basa sul metodo di eliminazione degli stati.

Procedimento di risoluzione

Per convertire un NFA in un'espressione regolare equivalente:

1. **Trasformazione iniziale**: Converti l'NFA in un GNFA (Automa a Stati Finiti Non-deterministico Generalizzato) in forma speciale:
   - Aggiungi un nuovo stato iniziale q_{start} che ha transizioni verso tutti gli altri stati, ma nessuna transizione entrante
   - Aggiungi un nuovo stato finale q_{accept} che ha transizioni da tutti gli altri stati, ma nessuna transizione uscente
   - Assicurati che ci sia una transizione (possibilmente etichettata con $\emptyset$) per ogni coppia di stati
2. **Eliminazione iterativa degli stati**: Elimina uno ad uno tutti gli stati diversi da q_{start} e q_{accept}:
   - Per ogni stato q_{rip} da eliminare, aggiorna le etichette delle transizioni dirette tra gli altri stati
   - Se abbiamo transizioni $q_i \xrightarrow{R_1} q_{\text{rip}}$, $q_{\text{rip}} \xrightarrow{R_2} q_{\text{rip}}$ (ciclo), e $q_{\text{rip}} \xrightarrow{R_3} q_j$
   - Più una transizione esistente $q_i \xrightarrow{R_4} q_j$
   - Allora la nuova etichetta diventa: $R_1(R_2)^*R_3 + R_4$
3. **Risultato finale**: Quando rimangono solo q_{start} e q_{accept}, l'etichetta della transizione da q_{start} a q_{accept} è l'espressione regolare equivalente all'NFA originale.

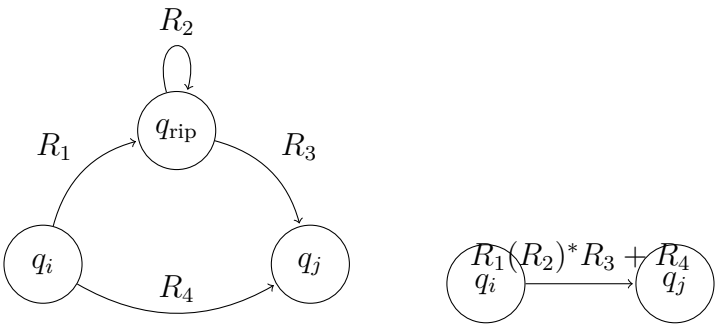


Figure 2: Schema di eliminazione di uno stato

Errore comune

Quando si applica il metodo di eliminazione degli stati, un errore comune è non considerare correttamente i cicli (self-loop) sullo stato da eliminare. Il termine $(R_2)^*$ è essenziale e rappresenta la possibilità di rimanere nello stato q_{rip} per zero o più ripetizioni.

4 Automi Nondeterministici Generalizzati (GNFA)

Concetto chiave

Un GNFA è un NFA dove le transizioni sono etichettate con espressioni regolari invece che con singoli simboli o ε. Questa struttura intermedia semplifica notevolmente la conversione da NFA a espressioni regolari.

4.1 Definizione Formale

Un Automa a Stati Finiti Non Deterministico Generalizzato (GNFA) è una quintupla $A = (Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$ dove:

- Q è un insieme finito di stati
- Σ è un alfabeto finito
- $\delta : Q \setminus \{q_{\text{accept}}\} \times Q \setminus \{q_{\text{start}}\} \mapsto R$ è una funzione di transizione che associa a ogni coppia di stati un'espressione regolare
- $q_{\text{start}} \in Q$ è lo stato iniziale
- $q_{\text{accept}} \in Q$ è lo stato finale

Suggerimento

Per convertire un NFA in un GNFA in forma speciale, segui questi passaggi:

1. Aggiungi un nuovo stato iniziale q_{start} con transizione ε verso il vecchio stato iniziale
2. Aggiungi un nuovo stato finale q_{accept} con transizioni ε da tutti i vecchi stati finali
3. Sostituisci le transizioni multiple tra due stati con l'unione delle rispettive etichette
4. Aggiungi transizioni etichettate con $\emptyset$ tra stati non collegati da alcuna transizione

5 Esercizi Guidati

5.1 Conversione da Espressione Regolare a ε-NFA

Procedimento di risoluzione

Convertiamo l'espressione regolare $(0+1)^*1(0+1)$ in un ε-NFA.

1. Scomponiamo l'espressione nelle sue componenti:
   - Partiamo con $(0+1)$: un automa per l'unione di 0 e 1
   - Applichiamo la chiusura di Kleene: $(0+1)^*$
   - Concateniamo con l'automa per il simbolo 1
   - Concateniamo con l'automa per $(0+1)$
2. Per (0+1), creiamo un NFA con uno stato iniziale che ha transizioni etichettate 0 e 1 verso uno stato finale.
3. Per $(0+1)^*$, aggiungiamo un nuovo stato iniziale/finale e le appropriate ε-transizioni per implementare il ciclo.
4. Per la concatenazione con 1, colleghiamo lo stato finale del precedente automa con un nuovo stato attraverso una transizione etichettata 1.
5. Infine, per la concatenazione con $(0+1)$, aggiungiamo transizioni opportune.

Il risultato finale sarà un ε-NFA che accetta esattamente il linguaggio rappresentato dall'espressione regolare $(0+1)^*1(0+1)$.

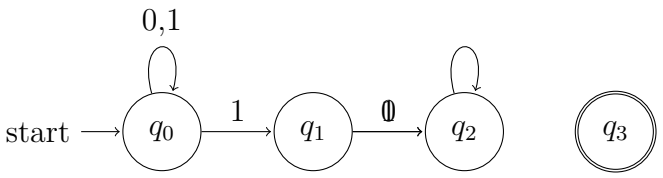
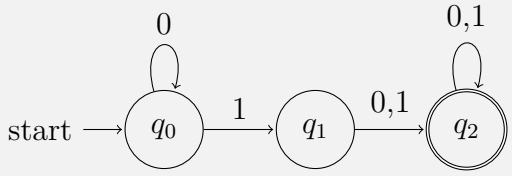


Figure 3: ε-NFA per l'espressione $(0+1)^*1(0+1)$ (semplificato)

5.2 Conversione da NFA a Espressione Regolare

Procedimento di risoluzione

Convertiamo l'NFA seguente in un'espressione regolare:



1. Trasformiamo l'NFA in un GNFA in forma speciale:
   - Aggiungiamo un nuovo stato iniziale q_{start} con transizione ε verso q_0
   - Aggiungiamo un nuovo stato finale q_{accept} con transizione ε da q_2
   - Aggiungiamo transizioni mancanti etichettate con $\emptyset$
2. Eliminiamo lo stato q_0:
   - Abbiamo $q_{\text{start}} \xrightarrow{\varepsilon} q_0$, $q_0 \xrightarrow{0} q_0$ (ciclo), e $q_0 \xrightarrow{1} q_1$
   - La nuova transizione da q_{start} a q_1 diventa $\varepsilon \cdot 0^* \cdot 1 = 0^*1$
   - Aggiorniamo anche le altre transizioni
3. Eliminiamo lo stato q_1:
   - Aggiorniamo le transizioni rimanenti
4. Leggiamo l'espressione regolare risultante dalla transizione da q_{start} a q_{accept}

Il risultato finale è l'espressione regolare $0^*1(0{+}1)(0{+}1)^*$, che può essere semplificata in $0^*1(0+1)^+$.

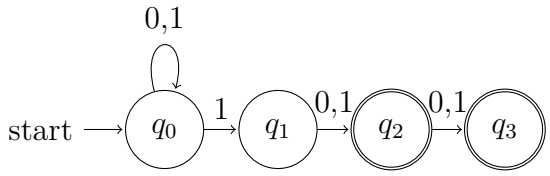
6 Esercizi Proposti

6.1 Espressioni Regolari

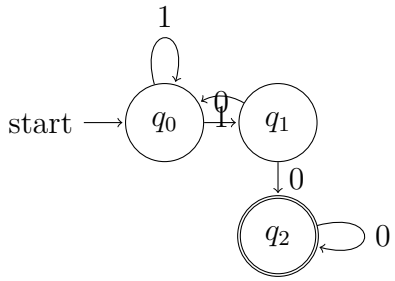
1. Scrivere un'espressione regolare per il linguaggio sull'alfabeto $\{a, b, c\}$ che contiene tutte le stringhe con un numero pari di a.
2. Scrivere un'espressione regolare per tutte le stringhe binarie che cominciano e finiscono con 1.
3. Scrivere un'espressione regolare per le stringhe binarie che contengono almeno tre 1 consecutivi.
4. Scrivere un'espressione regolare per le stringhe binarie che contengono almeno tre 1 (anche non consecutivi).
5. Scrivere un'espressione regolare per le stringhe che rappresentano date in formato GG/MM/AAAA.

6.2 Conversioni

1. Trasformare l'espressione regolare $(a + b)^*abb$ in un ε-NFA.
2. Convertire il seguente NFA in un'espressione regolare:



3. Convertire il seguente NFA in un'espressione regolare:



7 Risorse Aggiuntive

- **JFLAP**: strumento interattivo per la creazione e simulazione di automi, disponibile gratuitamente all'indirizzo http://www.jflap.org/.
- **Simulatori online**:
  - https://automata.cs.ru.nl/
  - https://ivanzuzak.info/noam/webapps/fsm_simulator/
  - https://cyberzhg.github.io/toolbox/nfa2re
- **Libri consigliati**:
  - Hopcroft, Motwani, Ullman. "Introduction to Automata Theory, Languages, and Computation"
  - Sipser. "Introduction to the Theory of Computation"