Espressioni Regolari

Fondamenti di Informatica I
Corso di laurea in Ingegneria Informatica e Automatica
Sapienza Università di Roma

Domenico Lembo, Paolo Liberatore,
Alberto Marchetti Spaccamela, Marco Schaerf

teoria delle espressioni regolari

modello matematico che include:

- definizione di stringa
- sintassi delle espressioni regolari
- Semantica

Si definisce l'insieme delle stringhe che soddisfa i requisiti specificati

notazione

- alfabeto: un insieme di simboli
- stringa: una sequenza (anche vuota) di elementi dell'alfabeto

se l'alfabeto è A:

A^n è l'insieme di tutte le stringhe di lunghezza n fatte di simboli di A

A* è l'insieme di tutte le stringhe (di lunghezza qualsiasi, anche zero) fatte di simboli di A

Stringhe

operazioni su stringhe:

- concatenazione

  $s \cdot r$ è una stringa contenente i simboli di s e poi i simboli di r
  anche scritta a volte senza il $\cdot$, cioè come sr

insieme vuoto

$\emptyset$

unione

$A \cup B$ tutte le stringhe di A e tutte quelle di B

intersezione

$A \cap B$ solo le stringhe che sono sia in A che in B

espressione regolare

una espressione regolare è un modo per specificare un insieme di stringhe

dato un alfabeto A, una espressione regolare lo divide in:

- stringhe che **collimano** con l'espressione regolare
- stringhe che **non collimano** con l'espressione regolare

sintassi delle espressioni regolari

dato un alfabeto:

- **ε** è una espressione regolare
- un simbolo dell'alfabeto è una espressione regolare
- se **e** e **f** sono due espressioni regolari, lo è anche **ef**
- se **e** è una espressione regolare allora lo è anche **e***
- se **e** e **f** sono due espressioni regolari, lo è anche **e|f**
- se **e** è una espressione regolare, lo è anche **e?**
- se **e** è una espressione regolare, lo è anche **(e)**

Significato (intuitivamente)

dato un alfabeto:

- **ε** indica una stringa vuota
- **ef** è una abbreviazione per **e·f**
- **e*** indica 0 o più occorrenze di **e.** In altri termini, indica una stringa vuota oppure una concatenazione di **e** con se stessa, un numero qualunque di volte
- **e|f** indica **e** oppure **f**
- **e?** indica che **e** è opzionale
- Le parentesi servono a specificare un ordine nell'uso degli operatori

stringa vuota

ε indica la stringa vuota

quindi: **ε|e** è uguale a **e?**

? si può definire in termini di **ε** e **|**

Semantica (formale)

espressione regolare → insieme di stringhe che collimano

la semantica è una funzione matematica *collimano()*

data una espressione regolare e:

collimano(e) = insieme di stringhe che collimano con l'espressione regolare

semantica: esempi

- *collimano(abc?)* = {ab, abc}
- *collimano(a*b)* = {b, ab, aab, aaab, … }
- *collimano(ab?|c*)* = {a, ab, ∅, c, cc, ccc, … }
- …

definizione generale di collimano?

semantica, in generale

- definire collimano per ogni espressione regolare
- modo "automatico" per dire quanto vale collimano(e)
- sistema: seguire la definizione sintattica

la sintassi era:

una espressione regolare è: un carattere, oppure ε, oppure …
vari casi

definire collimano per ciascun caso

sintassi ↔ semantica

Possibile espressione regolare	semantica
ε	*collima(ε)*=?
x ∈ alfabeto	*collima(x)*=?
ef	*collima(ef)*=?
e*	*collima(e*)*=?
e|f	*collima(e|f)*=?
e?	*collima(e?)*=?

in ognuno dei casi, definire collima

deve produrre le stringhe che collimano

semantica: casi facili

l'unica stringa che collima con ε è la stringa vuota

collima(ε)={∅}

l'unica stringa che collima con l'espressione regolare x è la stringa composta solo dal carattere x

collima(x)={x}

concatenazione

se è una concatenazione di caratteri:

collima(xy)={xy}

in generale?

es: a*b? è la concatenazione di a* e b?
cosa c'è in *collima(a*b?)*?

definizione ricorsiva della semantica

problema: definire *collima(a*b?)*

ma a* è una espressione regolare

anche b?

quindi *collima* sarà definito anche per loro

semantica delle composizioni

collima(ef)

insieme delle stringhe ottenute concatenando una stringa di *collima(e)* con una di *collima(f)*

collima(e)*

insieme delle stringhe ottenute concatenando un numero qualsiasi di stringhe di *collima(e)*

collima(e|f)

collima(e) ∪ *collima(f)* (unione)

collima(e?)

{∅} ∪ *collima(e)*

definizioni ricorsive

collima è definito in termini di se stesso?

non esattamente:

collima(e) è definito in termini di *collima(f)*, *collima(g)*, ecc.
dove f e g sono più semplici di e

esempi:

ef definito in termini di e ed f

e* definito in termini di e

ecc.

sviluppo della definizione

collima(e) è definito in termini di *collima(f)* ecc.,

più semplici

collima(f) è definito in termini di *collima(g)*, ecc.

ancora più semplici

ecc.

da *collima(e)* prima o poi si arriva a *collima(ε)* e *collima(x)* (che sono i casi base della definizione induttiva della semantica)

Alcune utili abbreviazioni

L'espressione $x_1|x_2|..|x_n$ con x_i appartenente all'alfabeto A si può anche scrivere come $[x_1x_2..x_n]$, con lo stesso significato

Inoltre, con l'espressione `[x-y]` indichiamo un qualsiasi carattere compreso fra il carattere denotato da x e quello denotato da y (si noti che l'alfabeto di simboli A è ordinato).

Ad esempio, l'espressione `[a-z]` indica qualunque carattere compreso fra a e b.

Se l'alfabeto A corrisponde alle lettere non accentate dell'alfabeto inglese, `[a-z]` è equivalente all'espressione regolare

`a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z`

Alcune utili abbreviazioni

Con l'espressione `[^x-y]` indichiamo un qualsiasi carattere dell'alfabeto eccetto i caratteri nella sequenza `x-y`

Se l'alfabeto A corrisponde alle lettere dell'alfabeto inglese, `[^b-f]` è equivalente all'espressione regolare

```
a|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
```

La variante `[^x]` indica un qualsiasi carattere dell'alfabeto eccetto il carattere `x`

Alcune utili abbreviazioni

e+ è l'espressione regolare che indica 1 o più occorrenze di **e**.

In altri termini, indica una concatenazione di **e** con se stessa, un numero qualunque di volte.

e+ è equivalente a **ee***

Alcune utili abbreviazioni

- **e{n}** è l'espressione regolare che collima con una sequenza di n stringhe che collimano con l'espressione **e**.

  e{n} è equivalente a **ee..e** (n volte)

  Esempio: **ba{2}b** collima con la stringa 'baab' ma non collima, ad esempio, con le stringhe 'bab' e 'baaab'

- **e{n,m}**, con n<=m, è l'espressione regolare che collima con una sequenza di *almeno* n stringhe ed al *massimo* m stringhe che collimano con l'espressione **e**.

  e{n,m} è equivalente a

ee....e	**ee..... e**	…	**ee....e**
n volte	n+1 volte	…	m volte

Esempio 1

- Si consideri come alfabeto quello costituito da soli caratteri alfabetici minuscoli. Scrivere l'espressione regolare che collima con le stringhe di lettere minuscole che contengono almeno una a e almeno una b, e in cui la prima a precede la prima b.

Soluzione Esempio 1

- Si consideri come alfabeto quello costituito da soli caratteri alfabetici minuscoli. Scrivere l'espressione regolare che collima con le stringhe di lettere minuscole che contengono almeno una a e almeno una b, e in cui la prima a precede la prima b.

Poiché la prima a deve venire prima di qualunque b, la parte iniziale dell'espressione regolare dovrà essere [c-z]*a, cioè una sequenza di lettere che non sono né a né b seguita da una a. poi ci saranno delle lettere qualunque ma non b e poi una b: [^b]*b. L'ultima parte sarà composta da lettere qualunque [a-z]*. Complessivamente l'espressione regolare sarà:

[c-z]*a[^b]*b[a-z]*

Esempio 2

- Si consideri come alfabeto quello costituito dalle sole cifre (i.e., 0 1 2…9). Scrivere l'espressione regolare che collima con le stringhe composte solo di cifre e in cui ogni volta che compare la cifra 1 le cifre immediatamente seguenti sono 234. Per esempio, collimano 21234236, 12345678, 281234789123490023 4, 233884 (nell'ultima non compare 1, quindi collima), mentre non collimano 134, 21567.

Soluzione Esempio 2

- Si consideri come alfabeto quello costituito dalle sole cifre (i.e., 0 1 2…9) . Scrivere l'espressione regolare che collima con le stringhe composte solo di cifre e in cui ogni volta che compare la cifra 1 le cifre immediatamente seguenti sono 234. Per esempio, collimano 21234236, 12345678, 2812347891234900234, 233884 (nell'ultima non compare 1, quindi collima),), mentre non collimano 134, 21567.

La parte iniziale della stringa può quindi contenere un numero qualunque di cifre diverse da 1 [^1]*, sarà poi seguita da una parte centrale composta 1 e poi dalla sequenza 234. La parte finale nuovamente sarà composta da cifre diverse da 1. La prima versione quindi è [^1]*1234[^1]*, ma questa soluzione non tiene in considerazione che la sequenza 1234 potrebbe comparire anche 0 o più di 1 volta e le varie occorrenza possono essere separate da altri caratteri. Per fare ciò ripetiamo 0 o più volte la seconda parte, ottenendo:

[^1]*(1234[^1]*)*

Esercizi

1. Scrivere l'espressione regolare che collima con le stringhe di lettere minuscole che iniziano con a e contengono almeno una b, e con quelle che iniziano con b e terminano con a.
2. L'espressione regolare a{0,5}|a?b+|abcd collima con la stringa vuota? Con qualche stringa di un solo carattere? Di qualche stringa di più di dieci caratteri? Motivare le risposte.
3. Scrivere l'espressione regolare che collima con le sequenze di cifre che rappresentano numeri interi da 1 a 16.
4. L'espressione regolare [0-9]+ [+] ([0-9]+)+ dovrebbe collimare con le somme di interi, come per esempio 2+21+4+5, ma è sbagliata. Dire perché lo è (fornire un esempio) e come andrebbe corretta per farla funzionare. (NOTA BENE: nell'espressione precedente la prima occorrenza di + ha un significato diverso dalla seconda occorrenza).

Espressioni regolari in Python

- Python, come quasi tutti i linguaggi di programmazione e molti altri programmi, permette di usare le espressioni regolari.
- Per utilizzarle bisogna importare un modulo ad-hoc chiamato **re**
- L'implementazione delle espressioni regolari in Python è basata sullo standard IEEE POSIX

Uso in Python

- Le espressioni regolari vengono principalmente usate per cercare una sottostringa con particolari proprietà all'interno di un testo oppure per effettuare sostituzioni complicate.
- Per questo motivo lo standard IEEE POSIX (e quindi Python) definiscono un insieme di funzioni e caratteri speciali più esteso di quelle definite nella teoria che abbiamo appena visto

Altri usi delle espressioni regolari

- nella funzione di ricerca di editor di testo
- come controllo dei campi di un modulo (form)
- …

all'interno di una pagina html5 si può, ad esempio, specificare:

```
<input type="text" title="una data"
 pattern="[0-9]{1,2}/[0-9]{1,2}/[0-9]{4}" />
```

```
Dove {1,2} vuol dire ripetere almeno 1 massimo
2 volte, mentre {4} vuol dire esattamente 4
volte
```