

Espressioni Regolari

Gianni Ceccarelli <dakkar@thenutilus.net>

18 aprile 2008

Riferimenti

- Mastering Regular Expressions, O'Reilly
- <http://thenautilus.dyndns.org/~dakkar/regex/>

Argomenti

- 1 Alfabeti e linguaggi
 - Definizioni
 - Linguaggi Regolari e NFA
- 2 Espressioni Regolari
- 3 Implementazioni

Argomenti

- 1 Alfabeti e linguaggi
 - Definizioni
 - Linguaggi Regolari e NFA

Cos'è un alfabeto

Un alfabeto è un insieme di *simboli*.

- A B C

Cos'è un alfabeto

Un alfabeto è un insieme di *simboli*.

- A B C
- 1 2 3

Cos'è un alfabeto

Un alfabeto è un insieme di *simboli*.

- A B C
- 1 2 3
- $\alpha \beta \gamma$

Cos'è un alfabeto

Un alfabeto è un insieme di *simboli*.

- A B C
- 1 2 3
- $\alpha \beta \gamma$
- $\uparrow \leftarrow \downarrow \rightarrow \diamond \square \bigcirc \times$

Cos'è un linguaggio

Un linguaggio è un insieme di *stringhe*

Cos'è un linguaggio

Un linguaggio è un insieme di *stringhe*
Una stringa è una sequenza di *simboli*

Cos'è un linguaggio

Un linguaggio è un insieme di *stringhe*

Una stringa è una sequenza di *simboli*

Per cui un linguaggio è definito solo rispetto a un alfabeto

Qualche esempio

Alfabeto: A B C

Linguaggio:

- A
- AA
- B
- CA

Qualche esempio

Alfabeto: A B C

Linguaggio:

- A
- AA
- AAA
- AAAA

Qualche esempio

Alfabeto: A B C

Linguaggio:

- A
- AA
- AAA
- AAAA
- ...

Qualche esempio

Alfabeto: A B C

Linguaggio:

tutte le parole italiane scrivibili con quelle sole 3 lettere

Qualche esempio

Alfabeto: 0 1 2 3 4 5 6 7 8 9

Linguaggio: le rappresentazioni in base 10 di tutti i numeri primi

Qualche esempio

Alfabeto: $\uparrow \leftarrow \downarrow \rightarrow \diamond \square \bigcirc \times$

Linguaggio: le sequenze di tasti per le mosse di un personaggio di Tekken

Argomenti

- 1 Alfabeti e linguaggi
 - Definizioni
 - Linguaggi Regolari e NFA

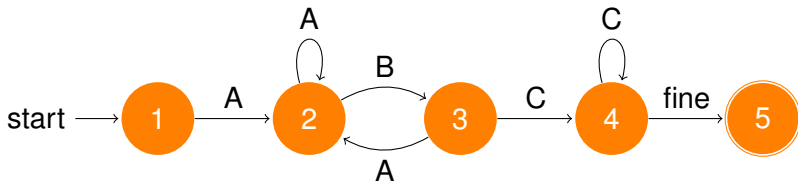
Definizioni

- Classe interessante di linguaggi: *linguaggi regolari*

Definizioni

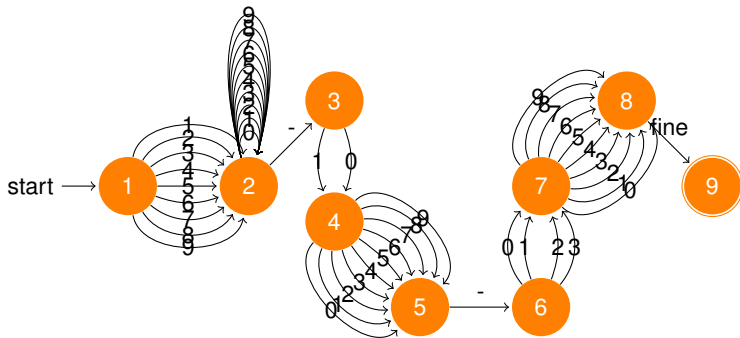
- Classe interessante di linguaggi: *linguaggi regolari*
- Tutti e soli i linguaggi accettati da *automi a stati finiti*

Esempio: AAABCC



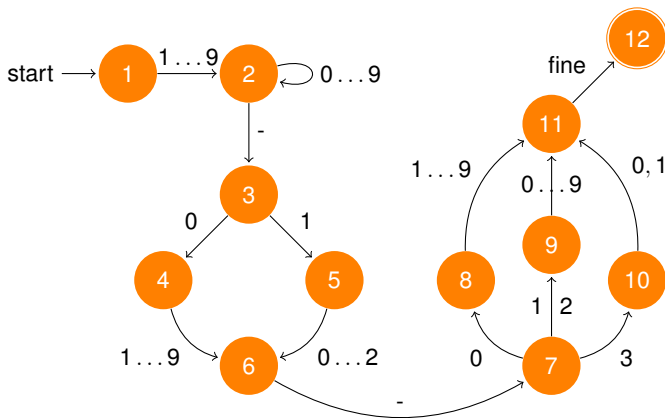
ABC AABC ABCC ABABABABCCCCC ...

Esempio: Data (semplice)



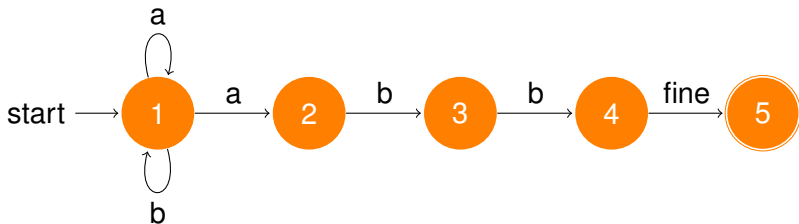
2008-04-19 2008-02-31 123-19-39

Esempio: Data (meno semplice)

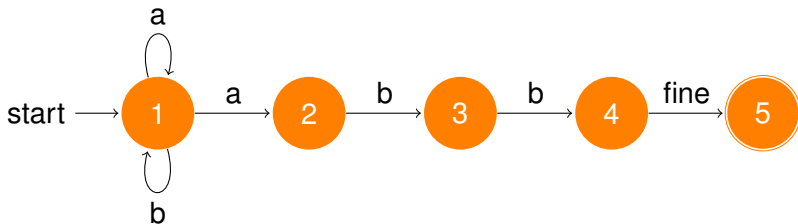


2008-04-19 2008-02-31 ~~123-19-39~~

Automati *non deterministici*



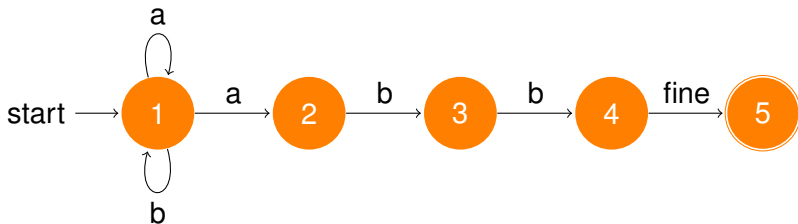
Automati *non deterministici*



Input: abb

Stati: 1 2 3 4 5

Automi *non deterministici*



Input: abbabb

Stati: 1 1 1 2 3 4 5

Argomenti

- 1 Alfabeti e linguaggi
- 2 **Espressioni Regolari**
 - Scrivere NFA
 - Corrispondenza, ricerca, cattura
 - Scrivere NFA, davvero
- 3 Implementazioni

Argomenti

- 2 **Espressioni Regolari**
 - **Scrivere NFA**
 - Corrispondenza, ricerca, cattura
 - Scrivere NFA, davvero

Disegnare un NFA è scomodo

Disegnare un NFA è scomodo, specie se lo dobbiamo spiegare a una macchina

Disegnare un NFA è scomodo, specie se lo dobbiamo spiegare
a una macchina
Ci serve una *sintassi* per descriverli

Disegnare un NFA è scomodo, specie se lo dobbiamo spiegare a una macchina

Ci serve una *sintassi* per descriverli

Questa sintassi definisce le *espressioni regolari*

Sintassi

- ciascun simbolo dell'alfabeto rappresenta un automa che accetta quel simbolo

Sintassi

- ciascun simbolo dell'alfabeto rappresenta un automa che accetta quel simbolo
- `.` rappresenta un automa che accetta un simbolo qualsiasi

Sintassi

- ciascun simbolo dell'alfabeto rappresenta un automa che accetta quel simbolo
- $.$ rappresenta un automa che accetta un simbolo qualsiasi
- xy rappresenta un automa che accetta ciò che viene accettato dall'automa rappresentato da x , *seguito da* ciò che viene accettato dall'automa rappresentato da y

Sintassi

- ciascun simbolo dell'alfabeto rappresenta un automa che accetta quel simbolo
- $.$ rappresenta un automa che accetta un simbolo qualsiasi
- xy rappresenta un automa che accetta ciò che viene accettato dall'automato rappresentato da x , *seguito da* ciò che viene accettato dall'automato rappresentato da y
- $x|y$ rappresenta un automa che accetta ciò che viene accettato dall'automato rappresentato da x , *oppure* ciò che viene accettato dall'automato rappresentato da y

Sintassi

- ciascun simbolo dell'alfabeto rappresenta un automa che accetta quel simbolo
- $.$ rappresenta un automa che accetta un simbolo qualsiasi
- $x\bar{y}$ rappresenta un automa che accetta ciò che viene accettato dall'automa rappresentato da x , *seguito da* ciò che viene accettato dall'automa rappresentato da y
- $x|y$ rappresenta un automa che accetta ciò che viene accettato dall'automa rappresentato da x , *oppure* ciò che viene accettato dall'automa rappresentato da y
- x^* rappresenta un automa che accetta un numero illimitato di volte (incluso 0) ciò che viene accettato dall'automa rappresentato da x

Sintassi

- ciascun simbolo dell'alfabeto rappresenta un automa che accetta quel simbolo
- $.$ rappresenta un automa che accetta un simbolo qualsiasi
- $x\bar{y}$ rappresenta un automa che accetta ciò che viene accettato dall'automa rappresentato da x , *seguito da* ciò che viene accettato dall'automa rappresentato da y
- $x|y$ rappresenta un automa che accetta ciò che viene accettato dall'automa rappresentato da x , *oppure* ciò che viene accettato dall'automa rappresentato da y
- x^* rappresenta un automa che accetta un numero illimitato di volte (incluso 0) ciò che viene accettato dall'automa rappresentato da x

Non proprio chiarissimo?

Sintassi, più chiara

α	sé stesso
\cdot	un simbolo qualsiasi
xy	x seguito da y
$x y$	x oppure y
x^*	0 o più x

Sintassi estesa

α	sé stesso
\cdot	un simbolo qualsiasi
xy	x seguito da y
$x y$	x oppure y
$[\alpha\beta]$	uno qualunque dei simboli
x^*	0 o più x
x^+	1 o più x
$x^?$	0 o 1 x
(x)	raggruppamento

Gli esempi

AAABCC (A+B) +C+

Gli esempi

AAABCC (A+B)+C+

Data semplice [123456789][0123456789]-
[01][0123456789]-[0123][0123456789]

Gli esempi

AAABCC $(A+B)^+C^+$

Data semplice $[123456789][0123456789]^-[01][0123456789]^-[0123][0123456789]$

Data meno semplice $[123456789][0123456789]^-[0][123456789]|1[012])^-[0][123456789]|1[0123456789]|3[01])$

Gli esempi

AAABCC $(A+B)+C+$

Data semplice $[123456789][0123456789]-$
 $[01][0123456789]-[0123][0123456789]$

Data meno semplice $[123456789][0123456789]-$
 $(0[123456789]|1[012])-$
 $(0[123456789]|1[0123456789]|3[01])$

Non-deterministico $(a|b)*abb$

Argomenti

- 2 **Espressioni Regolari**
 - Scrivere NFA
 - **Corrispondenza, ricerca, cattura**
 - Scrivere NFA, davvero

Corrispondenza

- la stringa appartiene al linguaggio

Corrispondenza

- la stringa appartiene al linguaggio
- l'automa accetta la stringa

Corrispondenza

- la stringa appartiene al linguaggio
- l'automa accetta la stringa
- la stringa corrisponde all'espressione regolare

Corrispondenza

- la stringa appartiene al linguaggio
- l'automa accetta la stringa
- la stringa corrisponde all'espressione regolare
- detto anche *match*

Ricerca

- forse ci basta che una *sotto-stringa* appartenga al linguaggio

Ricerca

- forse ci basta che una *sotto-stringa* appartenga al linguaggio
- vogliamo “cercare” dentro la nostra stringa

Ricerca

- forse ci basta che una *sotto-stringa* appartenga al linguaggio
- vogliamo “cercare” dentro la nostra stringa
- se la stringa “contiene” l’espressione x , la stringa *corrisponde* a $. *x . *$

Nuova sintassi

α	sé stesso
.	un simbolo qualsiasi
xy	x seguito da y
$x y$	x oppure y
$[\alpha\beta]$	uno qualunque dei simboli
x^*	0 o più x
x^+	1 o più x
$x?$	0 o 1 x
(x)	raggruppamento
x	x a <i>inizio stringa</i>
$x\$$	x a <i>fine stringa</i>

“Teste” e “code” della stringa sono ignorate, a meno di ancoraggi

Cattura

Spesso non ci basta sapere che una stringa è fatta in un certo modo: vogliamo pure estrarne dei pezzi.

Cattura

Spesso non ci basta sapere che una stringa è fatta in un certo modo: vogliamo pure estrarne dei pezzi.
Nell'esempio della data, vorremmo avere anno, mese, giorno

Cattura

Spesso non ci basta sapere che una stringa è fatta in un certo modo: vogliamo pure estrarne dei pezzi.

Nell'esempio della data, vorremmo avere anno, mese, giorno

Quasi tutte le implementazioni permettono di accedere ai pezzi di stringa corrispondenti ai sotto-automati *tra parentesi*

Espressione $^([1-9][0-9])-(0[1-9]|1[012])-(0[1-9]|1[0-9]|3[01])\$$

Stringa 2008-04-19

Catture

- 1 2008
- 2 04
- 3 19

Argomenti

- 2 **Espressioni Regolari**
 - Scrivere NFA
 - Corrispondenza, ricerca, cattura
 - **Scrivere NFA, davvero**

Riguardiamo l'ultimo esempio

$^ ([1-9] [0-9]) - (0 [1-9] | 1 [012]) -$
 $(0 [1-9] | 1 [0-9] | 3 [01]) \$$

Cosa sono quei trattini tra le quadre?

Riguardiamo l'ultimo esempio

$^([1-9][0-9])-(0[1-9]|1[012])-(0[1-9]|1[0-9]|3[01])\$$

Cosa sono quei trattini tra le quadre?

Che facciamo se vogliamo riconoscere un carattere tipo (| [\$?

Riguardiamo l'ultimo esempio

$^ ([1-9] [0-9]) - (0 [1-9] | 1 [012]) -$
 $(0 [1-9] | 1 [0-9] | 3 [01]) \$$

Cosa sono quei trattini tra le quadre?

Che facciamo se vogliamo riconoscere un carattere tipo (| [\$?

Fin'ora abbiamo un po' truccato

Collisioni

- in teoria, i simboli usati per scrivere le espressioni e i simboli dell'alfabeto sono *diversi*

Collisioni

- in teoria, i simboli usati per scrivere le espressioni e i simboli dell'alfabeto sono *diversi*
- in pratica, si usano sempre gli stessi caratteri

Collisioni

- in teoria, i simboli usati per scrivere le espressioni e i simboli dell'alfabeto sono *diversi*
- in pratica, si usano sempre gli stessi caratteri
- per cui serve un modo per distinguerli

Trucchi sintattici

- i caratteri alfanumerici corrispondono a sé stessi
- alcuni caratteri sono speciali
- si cambia la “specialità” di un carattere precedendolo con \

Esempi

- `\ (. \)` accetta (g)

Esempi

- $\backslash (. \backslash)$ accetta (g)
- $(\backslash .)$ accetta . (e cattura)

Esempi

- $\backslash (. \backslash)$ accetta (g)
- $(\backslash .)$ accetta . (e cattura)
- $[0-9]$ accetta 7

Esempi

- `\(.\\)` accetta `(g)`
- `(\\.)` accetta `.` (e cattura)
- `[0-9]` accetta `7`
- `\n` accetta un “a capo”

Argomenti

1 Alfabeti e linguaggi

2 Espressioni Regolari

3 Implementazioni

- Perl
- Python
- Javascript
- Java

Argomenti

3 Implementazioni

- Perl
- Python
- Javascript
- Java

Sintassi

<code>\x{..}</code>	carattere Unicode, per codepoint
<code>\N{..}</code>	carattere Unicode, per nome
<code>\s</code>	uno spazio
<code>\S</code>	un non-spazio
<code>\w</code>	un alfanumerico
<code>\d</code>	una cifra
<code>\A</code>	inizio stringa
<code>\z</code>	fine stringa
<code>^</code>	inizio stringa o riga
<code>\$</code>	fino stringa o riga

Operatori

```
$stringa =~ m/x/smx;
```

```
# ricerca
```

Operatori

```
$stringa =~ m/x/smx;           # ricerca  
$stringa =~ s/x/y/smxg;       # sostituisci
```

Operatori

```
$stringa =~ m/x/smx;           # ricerca  
$stringa =~ s/x/y/smxg;       # sostituisci  
$stringa =~ m/(x)(y)/smx;     # catture  
$prima=$1;$seconda=$2;
```

Operatori

```
$stringa =~ m{x}smx;           # ricerca  
$stringa =~ s{x}{y}smxg;      # sostituisci  
$stringa =~ m{(x)(y)}smx;     # catture  
$prima=$1;$seconda=$2;
```

Operatori

```
$stringa =~ m{x}smx;           # ricerca  
$stringa =~ s{x}{y}smxg;      # sostituisci  
($prima, $seconda) =  
    ($stringa =~ m{(x)(y)}smx); # catture
```

Argomenti

3 Implementazioni

- Perl
- Python
- Javascript
- Java

Esempio

```
import re
```

Esempio

```
import re
```

```
x=re.compile(r' . (.) .',re.M|re.S|re.X)
```

Esempio

```
import re

x=re.compile(r' . (.) . ',re.M|re.S|re.X)

m=x.search(s)    # ricerca
```

Esempio

```
import re

x=re.compile(r'(.).',re.M|re.S|re.X)

m=x.search(s)    # ricerca
m=x.match(s)     # corrispondenza
```

Esempio

```
import re

x=re.compile(r'(.).',re.M|re.S|re.X)

m=x.search(s)      # ricerca
m=x.match(s)       # corrispondenza
x.sub('y',s)       # sostituisci
```

Esempio

```
import re

x=re.compile(r'(.).',re.M|re.S|re.X)

m=x.search(s)      # ricerca
m=x.match(s)       # corrispondenza
x.sub('y',s)       # sostituisci

g1=m.group(1)      # catture
```

Argomenti

3 Implementazioni

- Perl
- Python
- **Javascript**
- Java

Esempio

```
stringa.search(/x/);
```

```
# ricerca
```

Esempio

```
stringa.search(/x/);           # ricerca  
stringa.replace(/x/g, 'y');   # sostituisci
```

Esempio

```
stringa.search(/x/);           # ricerca  
stringa.replace(/x/g, 'y');    # sostituisci  
result=stringa.match(/(x)(y)/); # corrispondenza
```

Esempio

```
stringa.search(/x/);           # ricerca
stringa.replace(/x/g, 'y');    # sostituisci
result=stringa.match(/(x) (y) /); # corrispondenza
prima=result[1];               # catture
seconda=result[2];
```

Argomenti

3 Implementazioni

- Perl
- Python
- Javascript
- **Java**

Esempio

```
import java.util.regex.*;

public class Test {
    public static void main(String argv[]) {
        Pattern p = Pattern.compile("(.)");
        Matcher m = p.matcher(argv[0]);
        m.matches();           // corrispondenza
        m.find();              // ricerca
        m.replaceAll("y");    // sostituisci
        m.group(1);           // catture
    }
}
```