

Linguaggi regolari

Altri semplici riconoscitori in
Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di
LinuxAwk e programmazione
data-driven

Qualche esercizio sulle e.r.

Linguaggi formali e compilazione

Corso di Laurea in Informatica

A.A. 2015/2016

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Perché sono importanti

- ▶ Molto diffusi in ambito informatico.
- ▶ Tipici pattern di ricerca all'interno di documenti definiscono linguaggi regolari.
- ▶ Rivestono poi un ruolo cruciale nei linguaggi di programmazione.
- ▶ Sono infatti regolari, ad esempio:
 - ▶ l'insieme degli identificatori di funzione e di variabile;
 - ▶ l'insieme di tutte le costanti numeriche (integer o float).
- ▶ Sono inoltre regolari tutti i linguaggi *finiti*, cioè costituiti da un numero finito di stringhe.

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Definizione di linguaggio regolare

- ▶ Dato un alfabeto Σ cominciamo col definire *unitario su Σ* ogni linguaggio costituito da un singolo carattere di Σ
- ▶ Ad esempio, se $\Sigma = \{a, b, c\}$, i linguaggi unitari su Σ sono: $\{a\}$, $\{b\}$ e $\{c\}$
- ▶ Un linguaggio L su un alfabeto $\Sigma = \{a_1, \dots, a_n\}$ si dice *regolare* se può essere espresso usando un numero finito di operazioni di concatenazione, unione e chiusura riflessiva a partire dai suoi linguaggi unitari $\{a_1\}, \dots, \{a_n\}$
- ▶ Più precisamente:
 - ▶ $\{a_1\}, \dots, \{a_n\}$ sono linguaggi regolari
 - ▶ se R_1 ed R_2 sono linguaggi regolari, allora $R_1 \cup R_2$ e $R_1 R_2$ sono linguaggi regolari
 - ▶ se R è un linguaggio regolare allora R^* è un linguaggio regolare

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Esempi di linguaggi regolari

- ▶ Sia Σ l'alfabeto ASCII e sia $X = X_1X_2 \dots X_n$ una generica stringa di Σ^* . Il linguaggio $\{X\}$ è regolare in quanto esprimibile come concatenazione dei linguaggi unitari $\{X_1\}, \{X_2\}, \dots, \{X_n\}$
- ▶ Ad esempio $\{C++\}$ è concatenazione dei linguaggi unitari $\{C\}, \{+\}$ e $\{+\}$, mentre $\{Python\}$ è concatenazione dei linguaggi unitari $\{P\}, \{Y\}, \{t\}, \{h\}, \{o\}$ e $\{n\}$
- ▶ Il linguaggio $\{X, Y, Z\}$, dove X, Y e Z sono stringhe generiche sull'alfabeto ASCII è regolare perché esprimibile come unione dei linguaggi regolari $\{X\}, \{Y\}$, e $\{Z\}$

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Esempi di linguaggi regolari (continua)

- ▶ Il linguaggio $\{C++, Python\}$ è regolare perché unione di due linguaggi che sappiamo essere regolari
- ▶ Generalizzando gli esempi precedenti si dimostra facilmente come ogni linguaggio finito sia esprimibile come unione di concatenazioni di linguaggi unitari

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Altri esempi

- ▶ $\{ab, c\}^2$ è (un linguaggio) regolare perché:

$$\{ab, c\}^2 = (\{a\}\{b\} \cup \{c\})(\{a\}\{b\} \cup \{c\})$$

- ▶ $L = \{a^n | n \geq 0\}$ è regolare perché $L = \{a\}^*$
- ▶ Anche il linguaggio $L_7 = \{a^n b^m | n, m \geq 0\}$ è regolare poiché $L_7 = \{a\}^* \{b\}^*$, cioè è la concatenazione di due linguaggi regolari
- ▶ Il linguaggio $\{a\}^+ = \{a^n | n \geq 1\}$ è regolare perché $\{a\}^+ = \{a\}\{a\}^*$
- ▶ $(\{ab, c\}^2)^R$ è regolare poiché $(\{ab, c\}^2)^R = \{ba, c\}^2$
- ▶ In generale L^R è regolare se (e solo se) L è regolare.
- ▶ Il linguaggio $L_8 = \{a^n b^n | n \geq 0\}$ non è regolare
- ▶ Il linguaggio $L_9 = \{a^n | n \text{ primo}\}$ non è regolare

I linguaggi L_7 e L_8

- ▶ Programma Python per riconoscere stringhe del tipo $a^n b^m$
- ▶ Due programmi Python, con soluzioni leggermente diverse, per riconoscere stringhe del tipo $a^n b^n$.

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

- ▶ È regolare il linguaggio di tutti i possibili *url* (indirizzi web)
- ▶ Perché è regolare?
- ▶ Dobbiamo preliminarmente chiederci: come è fatto un url?
 - ▶ un nome di dominio, cioè una serie di stringhe separate da punti, ...
 - ▶ ... seguito da un simbolo “/” e da una serie opzionale di nomi di cartelle ...
 - ▶ ... seguita dal nome completo (nome e tipo) di un documento
- ▶ Possiamo specificare ogni singola parte, e poi un intero url, come linguaggio regolare?

Un esempio relativo al Web (continua)

- ▶ Sia $\Sigma = \{a, \dots, z, 0, 1, \dots, 9\}$ e poniamo $S = \Sigma^*$.
- ▶ Per identificare nomi di dominio composti da due sole stringhe, ad esempio `stackoverflow.com`, possiamo usare il linguaggio:

$$S\{.\}S$$

- ▶ Per identificare nomi di dominio composti da due, tre, o quattro stringhe possiamo invece usare il linguaggio:

$$L_{DN} = S\{.\}S \cup S\{.\}S\{.\}S \cup S\{.\}S\{.\}S\{.\}S$$

- ▶ Se volessimo identificare nomi di dominio composti da un numero arbitrario di stringhe (a patto che possano esistere) potremmo definire S_{DN} nel modo seguente:

$$L_{DN} = S\{.\}S(\{.\}S)^*$$

Un esempio relativo al Web (continua)

- ▶ Per identificare una sequenza (eventualmente vuota) di cartelle possiamo definire, analogamente, il linguaggio L_C :

$$L_C = (S\{/})^*$$

- ▶ Definiamo ancora il linguaggio I di tutti i possibili identificatori di file:

$$I = (\Sigma \cup \{.\})^*$$

- ▶ Infine il linguaggio L_U di tutti i possibili `url` può essere definito come:

$$L_U = L_{DN}\{/}L_C I$$

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Un esempio relativo al Web (continua)

- ▶ Qualcuno potrebbe obiettare che L_U contiene stringhe, come ad esempio, `www.unimore.pippo/fasullo.html`, che non rappresentano url validi.
- ▶ Vero! Però va osservato come l'obiezione non riguardi un "difetto strutturale" della specifica del linguaggio.
- ▶ Possiamo comunque risolvere agevolmente il problema, definendo dapprima il linguaggio L_{DPL} di tutti i *domini di primo livello* validi (o meglio, "attualmente validi"):

$$L_{DPL} = \{com, org, edu, net, eu, it, fr, \dots\}$$

- ▶ ... e quindi ridefinire L_{DN} come:

$$L_{DN} = S(\{.\}S)^* \{.\}L_{DPL}$$

- ▶ Le *espressioni regolari* su un alfabeto Σ sono un formalismo (cioè a loro volta sono linguaggi) per definire linguaggi regolari
- ▶ Definiremo dapprima le espressioni regolari (e.r.) nella forma matematicamente più pulita
- ▶ In seguito presenteremo “abbreviazioni” linguistiche comunemente riconosciute da molti strumenti/ambienti (da MS Word[®] a `grep`)
- ▶ Negli esercizi useremo quasi esclusivamente le espressioni nella forma base

Espressioni regolari di base

- ▶ Le e.r. su un alfabeto Σ riflettono le costruzioni usate nella definizione dei linguaggi regolari su Σ

Base

- ▶ Φ è un'espressione regolare che denota il linguaggio vuoto
- ▶ per ogni $a \in \Sigma$, \mathbf{a} è un'e.r. che denota il linguaggio unitario $\{a\}$

Ricorsione Se \mathcal{E} ed \mathcal{F} sono e.r. che denotano, rispettivamente, i linguaggi E ed F , allora la scrittura:

- ▶ $\mathcal{E}\mathcal{F}$ è un'e.r. che denota il linguaggio EF (*concatenazione*)
- ▶ $\mathcal{E} + \mathcal{F}$ (o $\mathcal{E}|\mathcal{F}$) è un'e.r. che denota il linguaggio $E \cup F$ (*unione*)
- ▶ \mathcal{E}^* è un'e.r. che denota il linguaggio E^* (*chiusura riflessiva*)

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Un'ulteriore regola

Parentesi. Se \mathcal{E} è un'e.r., la scrittura (\mathcal{E}) è un'e.r.
equivalente alla prima, cioè che denota lo
stesso insieme di stringhe

serve a forzare un ordine di composizione delle
espressioni diverso da quello standard (in base al quale
chiusura precede concatenazione che precede unione)

Esempi

- ▶ L'espressione regolare $0 + 1^*10$ su \mathcal{B} (interpretabile come $0 + ((1^*)10)$, in base alle regole di precedenza) denota il linguaggio $R_1 = \{0, 10, 110, 1110, \dots\}$
- ▶ Il linguaggio R_1 è chiaramente differente dal linguaggio R_2 su \mathcal{B} definito dall'espressione regolare $(0 + 1)^*10$, che consiste di tutte le stringhe binarie che terminano con 10
- ▶ Posto $\Sigma = \{a, b, c\}$, l'espressione regolare $\mathbf{a(b + c)^*a}$ denota il linguaggio R_3 su Σ costituito dalle stringhe che iniziano e terminano con il carattere a e che non contengono altri caratteri a
- ▶ La scrittura $(1 + 01)^*(0 + 1 + 01)$ denota il linguaggio delle stringhe su \mathcal{B} di lunghezza almeno 1 che non contengono due caratteri 0 consecutivi

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Abbreviazioni di uso pratico

- ▶ Il simbolo ϵ si usa per indicare l'insieme $\{\epsilon\}$ (Attenzione! Non è l'insieme vuoto)
- ▶ La scrittura $[\mathcal{E}]$ si può utilizzare al posto della e.r. $\mathcal{E}+\epsilon$ e l'operatore $[\]$ prende il nome di *opzione*
- ▶ Se è definito un ordinamento fra i caratteri di Σ , allora si possono utilizzare convenzioni specifiche per denotare intervalli di caratteri. Ad esempio, la scrittura $[a - f]$ denota i caratteri compresi fra a ed f (opzione su un intervallo)
- ▶ Le scritture $[xyz]$ e $[\^xyz]$ indicano rispettivamente un qualunque carattere appartenente o non appartenente all'insieme $\{x, y, z\}$

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Abbreviazioni di uso pratico

- ▶ Poiché $L^+ = LL^*$, l'operatore di chiusura (non riflessiva) è ammesso nelle e.r. dove si intende che $\mathcal{E}^+ = \mathcal{E}\mathcal{E}^*$ (in tal caso l'operatore di unione o alternativa viene sostituito da |)

- ▶ Poiché $L^n = \overbrace{LL \dots L}^n$, l'operatore di elevamento a potenza è ammesso nelle e.r. e si intende che

$$\mathcal{E}^n = \overbrace{\mathcal{E}\mathcal{E} \dots \mathcal{E}}^n$$

- ▶ La scrittura $[\mathcal{E}]_i^j$ si può utilizzare al posto della e.r. $\mathcal{E}^i + \mathcal{E}^{i+1} + \dots + \mathcal{E}^j$

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Definizione di numeri e di identificatori

- ▶ Per alcuni insiemi di caratteri di particolare importanza (cifre, lettere, caratteri alfanumerici, caratteri di spaziatura, ...) si possono usare espressioni specifiche, come ad esempio (prendendo a prestito la notazione dalle espressioni riconosciute dal comando `grep` di Linux):
[: *digit* :], [: *alpha* :], [: *alnum* :], [: *space* :], ...
- ▶ L'espressione regolare `[1 - 9][: digit :]*` denota l'insieme delle stringhe che rappresentano (nella consueta rappresentazione in base 10) i numeri interi positivi
- ▶ L'espressione regolare `[: alpha :]([: alpha :][: digit :])*` denota l'insieme degli identificatori legali in alcuni linguaggi di programmazione (soprattutto fra i più vecchi)

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Applicazioni diffuse che usano le espressioni regolari

- ▶ Innanzitutto gli editor di testo, più o meno sofisticati (MS Word[®], Libre Office Writer, Emacs, ...)
- ▶ Molte applicazioni che manipolano file eseguibili da linea di comando in ambiente Unix/Linux (ad esempio, `grep`, `find` e `sed`)
- ▶ Utility per la costruzione di analizzatori lessicali (come `Lex`)
- ▶ In tutti questi casi, la sintassi per le espressioni regolari è molto più ampia, e (pur non aumentando il potere espressivo) rende la definizione dei pattern molto più semplice

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

`Sed`, lo stream editor di Linux

`Awk` e programmazione data-driven

Qualche esercizio sulle e.r.

Espressioni regolari in applicazioni di Linux

- ▶ `.` ha un match con qualsiasi singolo carattere, ad eccezione di `'\n'`
- ▶ `*` ha un match con 0 o più copie della precedente espressione
- ▶ `+` ha un match con 1 o più copie della precedente espressione
- ▶ `?` ha un match con 0 o 1 copia della precedente espressione
- ▶ `{n}` e `{n,m}`, dove n ed m sono numeri, hanno un match con la precedente espressione rispettivamente n volte (prima forma) oppure fra n ed m volte (seconda forma)
- ▶ `[]` ha un match con qualunque carattere incluso tra le parentesi; se il primo carattere è `^`, allora c'è un match con qualunque carattere non incluso fra le parentesi

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Espressioni regolari in applicazioni di Linux

(2)

- ▶ Un simbolo - entro le parentesi quadre serve per indicare un intervallo di caratteri, come nel caso di `[0 - 9]`
- ▶ `^` come primo carattere di una e.r. ha un match con l'inizio di una linea
- ▶ `$` come ultimo carattere di una e.r. ha un match con la fine di una linea
- ▶ `\` è il classico carattere di escape
- ▶ `|` è il simbolo di alternativa
- ▶ `()`, o `\(\)`, servono per il raggruppamento di e.r. e per il loro eventuale riferimento

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Esempi di uso di e.r. in ambiente Libre Office

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Esempi di uso di e.r. In Libreoffice

1) Sostituire sequenze costituite da più spazi e/o tabulazioni con un singolo carattere spazio. Ad esempio:

La vispa Teresa fra l'erbetta rincorre la farfallotta!

Soluzione: `([:space:])\t{2,} → " "`

2) Inserire in ogni url l'indicazione dello schema (protocollo) `http://`. Ad esempio `www.unimore.it` deve diventare `http://www.unimore.it` e `stackoverflow.com` deve diventare `http://stackoverflow.com`

Soluzione: `([:alnum:]+\.)+(it|com|org|net|edu) → http://$0`

3) Attenzione però, perché `http://www.unimore.it` non deve diventare `http://http://www.unimore.it`.

Soluzione: `([:space:])([:alnum:]+\.)+(it|com|org|net|edu) → $1http://$2`

4) Eliminare ogni zero non significativo dai numeri, ad esempio, trasformare 001, 0.10 e 2.84000 in 1, 0.1 e 2.84 evitando invece di modificare 10 e 0.101. Idem per 05 e 0.0810700. Tuttavia, vorremmo che 1.0000 diventasse 1.0 e non 1, perché vorremmo che rimanesse un numero float. 1.000001

Soluzione (da applicare nell'ordine indicato):

`([:space:])0+([0-9])([^\0-9]) → $1$2$3`

`(\.[0-9]*[1-9])0+([^\0-9]) → $1$2`

`\.[0]+([^\0-9]) → .0$1`

Man pages di grep

Anchoring

The caret `^` and the dollar sign `$` are meta-characters that respectively match the empty string at the beginning and end of a line.

The Backslash Character and Special Expressions

The symbols `\<` and `\>` respectively match the empty string at the beginning and end of a word. The symbol `\b` matches the empty string at the edge of a word, and `\B` matches the empty string provided it's *not* at the edge of a word. The symbol `\w` is a synonym for `[[:alnum:]]` and `\W` is a synonym for `[^[:alnum:]]`.

Repetition

A regular expression may be followed by one of several repetition operators:

- ? The preceding item is optional and matched at most once.
 - *
 - +
 - {n}
 - {n,}
 - {,m}
 - {n,m}
- The preceding item is matched exactly n times.
 The preceding item is matched n or more times.
 The preceding item is matched at most m times.
 The preceding item is matched at least n times, but not more than m times.

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Esempi di uso di grep

- ▶ `grep -e 404 algogroup.log`
stampa tutte le richieste che contengono la sottostringa 404 (quindi, presumibilmente, che hanno originato status code 404 (file not found))
- ▶ `grep -e "algogroup.unimo\(re\)\"?.it"`
`algogroup.log`
stampa tutte le richieste il cui *referer* è una pagina del sito *algogroup.unimore.it* (viene accettato anche l'*alias* `algogroup.unimo.it`)
- ▶ `grep -e "http:[^;)\"]*" -o`
`algogroup.log`
`| sort | uniq -c | sort -n -r`
stampa i differenti referer, ordinati per numerosità decrescente

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

- ▶ `sed` è uno `stream editor`, cioè un editor utilizzabile per eseguire trasformazioni su uno stream di input (un file, un input da terminale o come risultato di un comando precedente)
- ▶ `sed` opera con un singolo passo (linea dopo linea) sull'input
- ▶ Non daremo la sintassi precisa del comando `sed`; vedremo piuttosto alcuni esempi

Esempi di uso di sed

- ▶ `sed 'd' test.txt` Legge tutte le linee del file `test.txt` e produce un output in cui esse sono tutte cancellate
- ▶ `sed '2d' test.txt` Cancella da `test.txt` la sola riga 2
- ▶ `sed '1,3d' test.txt` Cancella da `test.txt` le righe da 1 a 3
- ▶ `sed '/^1/d' test.txt` Cancella le righe che iniziano con 1
- ▶ `sed '/1$/d' test.txt` Cancella le righe che terminano con 1

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Esempi di uso di sed

- ▶ `sed '/\(1\)\1/d' test.txt` Cancella le righe che presentano almeno due ripetizioni consecutive della sequenza “ 1”
- ▶ `sed 's/abc/xyz/' test.txt` Sostituisce in ogni riga le prime occorrenze di `abc` con `xyz`
- ▶ `sed 's/abc/xyz/g' test.txt` Sostituisce tutte le occorrenze di `abc` con `xyz`
- ▶ `sed -n 's/abc/xyz/p' test.txt` Sostituisce in ogni riga le prime occorrenze di `abc` con `xyz`; produce in output solo le righe modificate
- ▶ `sed -n 's/abc/xyz/gp' test.txt` Sostituisce tutte le occorrenze di `abc` con `xyz`; produce in output solo le righe modificate

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Esempi di uso di sed

- ▶ `sed 's/a/b/g;s/b/c/g' test.txt`
 Sostituisce prima tutte le **a** con **b** e poi tutte le **b** (incluse quelle derivanti dalla precedente trasformazione) in **c**
- ▶ `sed 's/.$//' test.txt` Utile se si legge sotto Linux un file DOS/Windows
- ▶ `sed "s/E'/È/g;s/a' /à/g;s/i' /ì/g; \
 s/u' /ù/g;s/o' /ò/g;s/é/è/g; \
 s/e' /è/g;s/chè/ché/g" test.txt`
 Corregge le accentazioni e cambia apostrofi “impropri” in accenti in un documento scritto in italiano

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

Esempi di uso di sed

- ▶ `sed 's/^[\t]*//;s/[\t]*$//'` test.txt
Toglie spazi e tabulazioni ad inizio e fine riga
- ▶ `sed -n 's/.*href="\([^"]*\)".*/\1/p'` test.txt
Restituisce solo la parte riconosciuta dalla (sotto)espressione regolare inclusa fra `\(` e `\)`, in questo caso un url
- ▶ `sed 's/<[^>]*>//g'` test.html
Rimuove i tag in un file html
- ▶ `sed 's/[[:cntrl:]]//g'` test.txt
Elimina tutti i caratteri di controllo
- ▶ `sed 's:/usr/local:/usr:g; \s:/usr:/usr/local:g'` test.txt
Sostituisce tutte le occorrenze di `/usr` ma non di `/usr/local` con `/usr/local`
- ▶ `'s/<[^>]*>//g;s/^[\t]*//;s/[\t]*$//; \s/[[:cntrl:]]//g;/^$/d'` test.txt
Quale effetto ha?

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

- ▶ Un programma `Awk` elabora l'input linea per linea (o, più in generale, *record per record*)
- ▶ Per ogni linea che soddisfa precise caratteristiche, `Awk` esegue una corrispondente azione
- ▶ Tipicamente, la condizione che fa “scattare” l'azione è la presenza, nella linea corrente, di una porzione di testo che soddisfa un determinato modello (*pattern*) descritto mediante un'espressione regolare
- ▶ Un programma `Awk` è dunque definito da un insieme di regole:

```
pattern { action }  
pattern { action }
```

...

Struttura dell'input per Awk

- ▶ Awk “vede” il file di input come sequenza di *record*
- ▶ La separazione dell'input in record è determinata da un carattere speciale, detto appunto *record separator*
- ▶ Il separatore default è il carattere di *newline*, per cui normalmente un record coincide con una linea
- ▶ A sua volta, un record è diviso in *campi (field)* per la presenza di *caratteri spazio* (spazi e tabulazioni)
- ▶ La separazione in campi può essere determinata da separatori diversi, specificati mediante espressioni regolari
- ▶ Le variabili speciali \$1, \$2, ... possono essere usate per fare riferimento ai campi del record corrente
- ▶ L'intero record corrente è riferibile con \$0

Esecuzione di programmi Awk

► Ci sono essenzialmente due possibilità:

- ▶ > awk '<programma>' <if1> <if2> ...
- ▶ > awk -f <file-programma> <if1> <if2>
- ...

► Primi semplici esempi:

```
ls -l | awk '{print $0}'
```

```
ls -l | awk '{print $5, $9}'
```

► Uso di pattern

```
ls -l | awk '/root/ {print $0}'
```

```
ls -l |
```

```
awk '$3 ~ /root/ {print $5, $9}'
```

Espressioni e linguaggi regolari

Linguaggi regolari

Altri semplici riconoscitori in Python

Il linguaggio delle url

Espressioni regolari

Sed, lo stream editor di Linux

Awk e programmazione data-driven

Qualche esercizio sulle e.r.

- ▶ **Uso di BEGIN e END: calcolo occupazione disco**

```
ls -l | awk 'BEGIN {CNT=0}
            {CNT = CNT + $5}
            END {print int(CNT/1024), "KB"}'
```

- ▶ **Owner e filename dei file scrivibili da tutti:**

```
ls -l |
awk '$1 ~ /.{7}rwx/ print $3,$9'
```

Qualche esercizio di prova

- ▶ Scrivere un'e.r. per il seguente linguaggio sull'alfabeto $\{a, b, c\}$:

$$E_1 = \{a^n b^m c^k \mid m = 0 \Rightarrow k = 3\}$$

- ▶ Scrivere un'e.r. per il linguaggio E_2 , sull'alfabeto $\{a, b\}$, delle stringhe contenenti al più due a
- ▶ Scrivere un'e.r. per il linguaggio E_3 , sull'alfabeto $\{a, b\}$, delle stringhe contenenti un numero dispari di b
- ▶ Scrivere un'e.r. per il linguaggio E_4 , sull'alfabeto $\{a, b\}$, definito ricorsivamente nel modo seguente:
 1. $\epsilon \in E_4$;
 2. Se $x \in E_4$ allora anche $abax \in E_4$ e $xaa \in E_4$
 Inoltre, solo stringhe ottenibili in questo modo appartengono a E_4 .

- ▶ Scrivere un'e.r. per il linguaggio E_5 , sull'alfabeto $\{a, b, c\}$, costituito dalle stringhe in cui ogni occorrenza di b è seguita da almeno un'occorrenza di c
- ▶ Descrivere nel modo più semplice possibile, in Italiano, il linguaggio corrispondente alla seguente espressione regolare: $((a|b)^3)^*(a|b)$
- ▶ Si dica qual è la stringa più corta che non appartiene al linguaggio descritto dall'espressione regolare $a^*(ab)^*b^*$

Alcuni esercizi dati all'esame

- ▶ Si scriva un'espressione regolare per definire il linguaggio delle stringhe sull'alfabeto $\{0, 1\}$ che non contengono tre 1 di fila
- ▶ Si consideri l'espressione regolare

$$\mathbf{b^* aa (ba | b)^* b}$$

e si descriva “a parole” il linguaggio da essa rappresentato

- ▶ Si scriva un'espressione regolare per il linguaggio su $\{0, 1\}$ costituito dalle stringhe che iniziano con 00 oppure terminano con 01