Espressioni regolari

Alfabeto

- Un'espressione regolare è una sequenza di caratteri che identifica un insieme di stringhe
- Se viene utilizzata per vincolare uno spazio lessicale solo le stringhe appartenenti all'insieme identificato rappresentano valori validi
- Un'espressione regolare può essere suddivisa in:
  - caratteri ordinari: trovano una corrispondenza diretta nelle stringe dell'insieme denotato
  - metacaratteri: caratteri speciali che assumono caratteristiche di controllo sui caratteri ordinari
- Lista dei metacaratteri:

[ ] \ { } | ( ) ^ ? + * .

Elementi di base

- **Atomo**: un carattere ordinario, un gruppo di caratteri (delimitato da []) o un'espressione regolare fra parentesi tonde
  - Sono atomi:
    `a, [abc], (abc), (a*b+c?) ...`

➔ Una qualunque espressione racchiusa da () diventa un atomo

- **Parte**: un atomo eventualmente seguito da un quantificatore
  - Sono parti:
    `a, a*, a{5,6}, (a*b+), (a*b+)?`

Quantificatori

- Un **quantificatore** (quantifier) vincola l'atomo che lo precede a comparire tante volte quanto indicato (m e n sono valori numerici generici)
  - `{n}`: esattamente n volte
  - `{m,n}`: almeno m e al più n volte
  - `{0,n}`: al più n volte
  - `{m,}`: almeno m volte
- Quindi: `{3,7}` significa da 3 a 7 volte
- Sono previste anche forme abbreviate (sono quelle usate anche dai DTD:
  - `*` equivale a `{0,}`
  - `+` equivale a `{1,}`
  - `?` equivale a `{0,1}`
- **Attenzione:** il quantificatore {1} si può omettere

Espressione banale

- **Espressione banale**: è la forma più semplice (e più banale) di espressione regolare costituita solo da caratteri ordinari (senza metacaratteri)
- Identifica un insieme composto da un solo elemento: la stringa uguale all'espressione stessa
- Esempio: l'espressione banana identifica solo la stringa "banana"

Esempi di espressioni con quantificatori

- `a*b+c?`
  - Interpretazione: 0 o più occorrenze di a, seguite da 1 o più b seguite da 0 o 1 occorrenza di c.
  - Stringhe ammissibili: `a`, `ab`, `abc`, `bb`, `abb`, `abbc`, `aa`
- `Ar(har){1,2}ha`
  - Interpretazione: "Ar" seguito da 1 o due occorrenze di "har" seguita da "ha"
  - Stringhe ammissibili: `Arharha`, `Arharharha`
- `Ar*gh`
  - Interpretazione: "A" seguita da un numero qualunque di "r" seguita da "gh"
  - Stringhe ammissibili: `Argh`, `Arrrgh`, `Arrrrgh...`

💣 Il quantificatore si applica all'atomo che lo precede:
  - Se scriviamo ab+ il quantificatore agisce solo su 'b'
  - Se vogliamo applicarlo ad "ab" scriveremo (ab)+

Gruppi di caratteri - 1

- L'atomo `[elenco-caratteri]` definisce un insieme da cui possono essere scelti tanti caratteri quanti indicati dal quantificatore applicato
- Esempio: `[ciao]{4}`
  - Interpretazione: tutte le stringhe lunghe esattamente 4 composte con i 4 caratteri: c, i, a, o
  - Stringhe ammissibili: `ciao`, `oaic`, `iaoc`, …
- `[`c_1`-`c_2`]` indica un insieme composto da un intervallo di caratteri, compreso fra c_1 e c_2
- Esempio: `[A-Z]{1,10}` tutte le "parole" da 1 a 10 lettere componibili con le lettere maiuscole
- È possibile concatenare intervalli:
  `[A-Za-z]`= tutte le lettere dell'alfabeto

Gruppi di caratteri - 2

- Con ^ è possibile negare un gruppo di caratteri ovvero richiedere che il carattere non sia fra quelli indicati.
  `[^elenco-caratteri]` oppure `[^intervallo]`
- Esempio: `[^A-Z]{5}` = tutte le "parole" di 5 caratteri che non contengono le lettere maiuscole
- Esistono abbreviazioni che rappresentano gruppi di caratteri predefiniti (tra parentesi le equivalenze):
  - `\d`: qualsiasi cifra decimale (`[0-9]`)
  - `\D`: qualsiasi carattere escluse le cifre (`[^0-9]`)
  - `\w`: qualsiasi carattere alfanumerico, compresi '_' e spazio (`[a-zA-Z0-9_ ]`)
  - `\W`: qualsiasi carattere non alfanumerico (`[^a-zA-Z0-9_ ]`)

Altri metacaratteri

- . corrisponde a un carattere qualsiasi
  - Esempio: [.] {10} rappresenta tutte le parole lunghe 10 caratteri
- | separa espressioni regolari in OR (branch)
- Esempi:
  - Se la regola è abc|def|xyz vanno bene: abc, def e xyz
  - Se la regola è abc(def|xyz) vanno bene: abcdef e abcxyz
- \ consente di inserire un metacarattere come carattere ordinario (\ si dice carattere di escape)
- Esempi:
  - \. indica il carattere punto
  - \(indica la parentesi aperta

Esempio: codice fiscale

- Un codice fiscale ha questa struttura:

  RSS MRA 80B21 H501V
- Quindi: 6 lettere maiuscole, 2 cifre, 1 lettera maiuscola, 2 cifre, 1 lettera maiuscola, 3 cifre e 1 lettera maiuscola
- Esprimendo questa regola sotto forma di espressione regolare possiamo definire un pattern di validazione per i codici fiscali:

```
<xs:simpleType name="CodFiscType">
 <xs:restriction base="xs:string">
  <xs:pattern
   value="[A-Z]{6}\d{2}[A-Z]\d{2}[A-Z]\d{3}[A-Z]"/>
 </xs:restriction>
</xs:simpleType>
```

Esempio

- Definiamo un pattern che valida i numeri che terminano per ‘0’ o ‘5’
- Utile quando c’era la lira e il taglio minimo era 5 lire

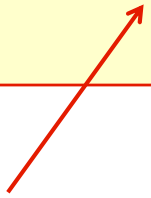
```
<xs:simpleType name="LireType">
 <xs:restriction base="xs:positiveInteger">
  <xs:pattern value="\d*[50]"/>
 </xs:restriction>
</xs:simpleType>
```

Tutte le cifre che vogliamo
basta che l’ultima cifra sia 0 o 5

Esempio

- Euro – tipo derivato che accetta numeri con **esattamente** due cifre decimali

```
<xs:simpleType name="StrictEuroType">
 <xs:restriction base="EuroType">
  <xs:pattern value=“\d*\.\d{2}" />
 </xs:restriction>
</xs:simpleType>
```



Tutte le cifre che vogliamo basta che ci sia
il carattere “.” seguito da 2 cifre.
Per inserire “.” abbiamo dovuto usare
un carattere di escape \.