UNIVERSITÀ
DEGLI STUDI
DI TORINO

Definizioni regolari e Analizzatori lessicali (cenni)

a.a. 2016-2017

Riferimenti bibliografici

Compilatori: principi, tecniche e strumenti
A.V. Aho, M. S. Lam, R. Sethi, J.D. Ullman

Analisi lessicale [cap.3]

3.8 Progettazione di un generatore di analizzatori lessicali

3.8.1 Struttura dell'analizzatore generato

3.8.2 Riconoscimento dei pattern basato su NFA

3.8.3 DFA per analizzatori lessicali

Estensioni delle espressioni regolari

Possiamo introdurre dei nomi simbolici per individuare specifiche espressioni regolari.

letter → A | B |. . . | Z | a | b | . . . | z |
digit → 0 | 1 | . . . | 9
id → *letter_*(*letter_* | *digit*)*

Notazioni alternative

letter → [A-Za-z_]
digit → [0-9]
id → *letter*(*letter* | *digit*)*

Estensioni delle espressioni regolari

digits	→	*digit digit**
opfraction	→	*. digits* | ε
opexp	→	(E(+ | - | ε) *digits*) | ε
number	→	*digits opfraction opexp*

Abbreviazioni: **e?** sta per **(e + ε)** (dove e è una r.e.)

[0-9] sta per 0 | 1 | . . . | 9

digit	→	[0-9]
digits	→	*digit* $^+$
number	→	*digits (. digits)?* (E [+-]? *digits*)?

Estensioni delle espressioni regolari

Esempio: specifica di pattern in Unix:

- **Espressioni regolari in Unix**:
  Esempio:

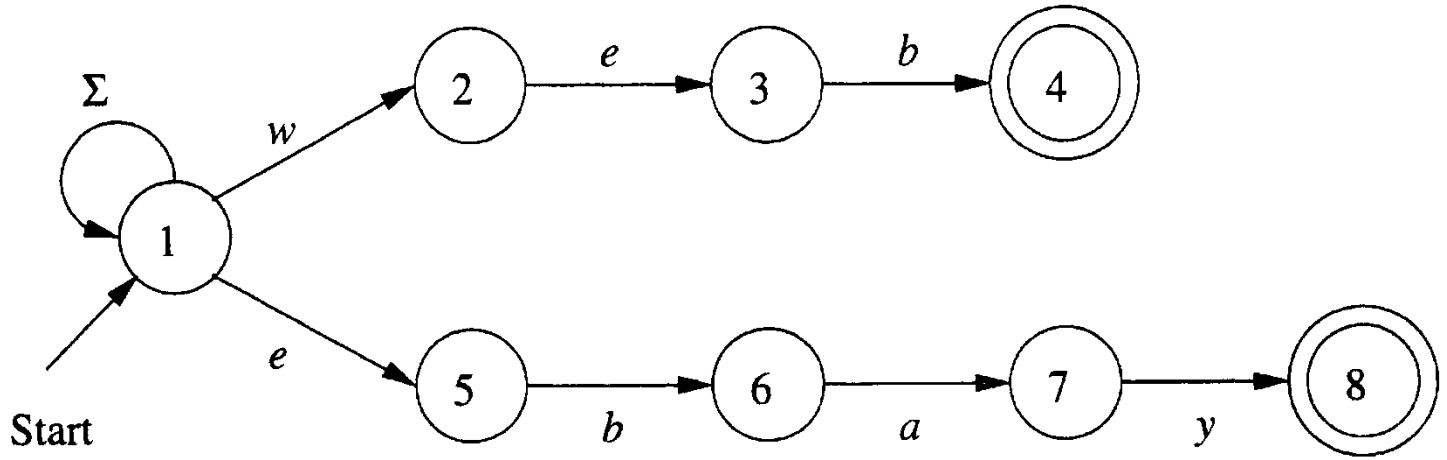
  `'[A-Z][a-z]*[][A-Z][A-Z]'`

  e' compatibile con (matches) `Ithaca NY`
  non e' compatibile con `Palo Alto CA`

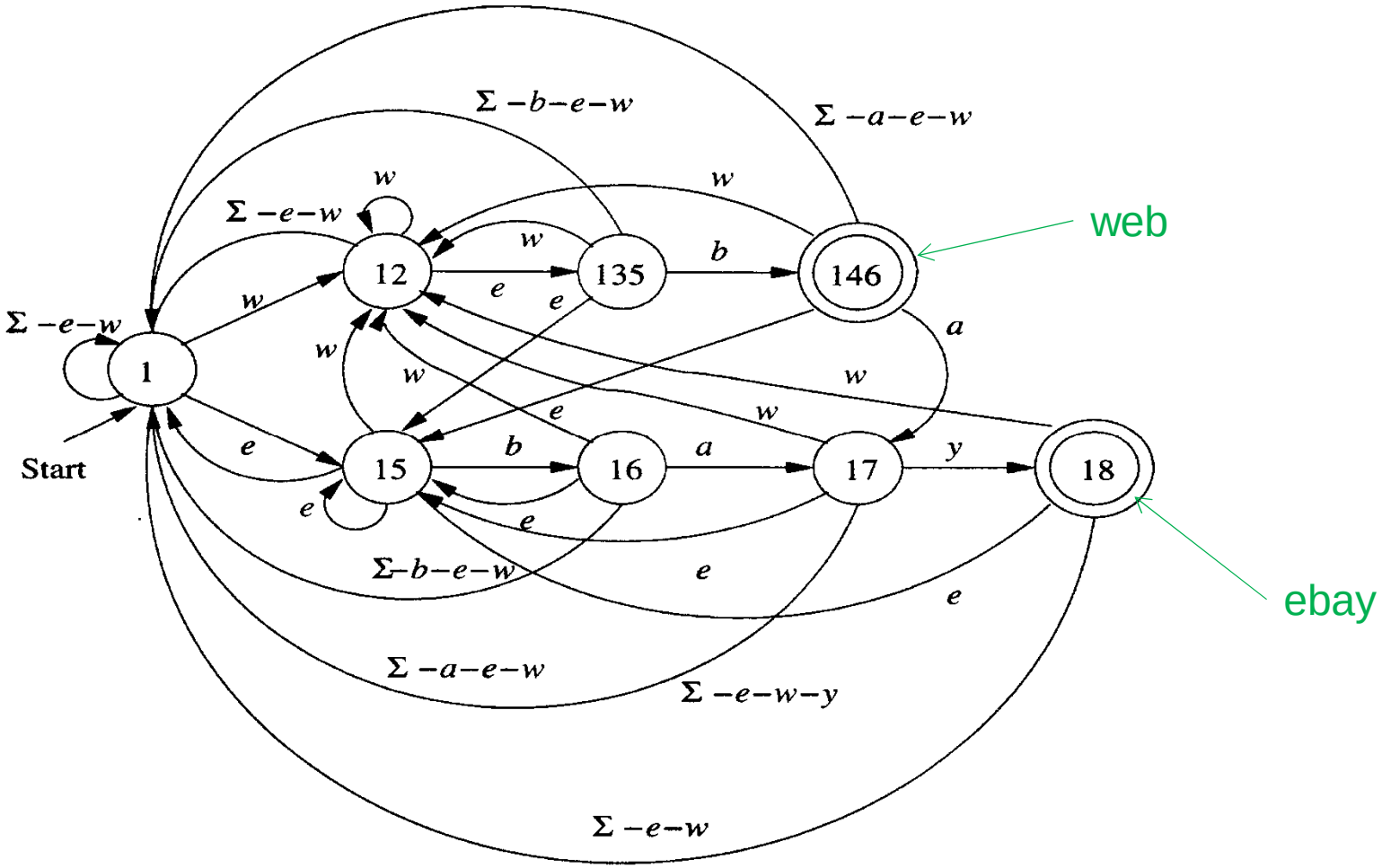
- Domanda: Quale espressione e' compatibile con
  `Palo Alto CA`?

Un’applicazione: riconoscimento di stringhe in un testo

Riconoscimento in un testo, su un alfabeto Σ, delle parole ‘’web’’ e ‘’ebay’’



Trasformazione in un automa deterministico



Generatori di analizzatori lessicali: Lex

1. Trasforma ogni espressione regolare in un NFA
2. Combina gli NFA in un unico automa aggiungendo un nuovo stato iniziale con transizioni ε verso ognuno degli stati iniziali degli automi N_i relativi ai pattern p_i.
3. Riconosce il *prefisso piu` lungo che soddisfa* un pattern.
4. Se una stringa soddisfa piu` pattern, viene considerata un lessema del primo che compare nel programma Lex.
5. Alcuni analizzatori convertono l'NFA relativo a tutti i pattern in un DFA equivalente mediante la costruzione per sottinsiemi.

Analizzatore lessicale

Legge la stringa di caratteri del programma sorgente e raggruppa i caratteri in sequenze chiamate *lessemi*

Per ogni lessema, crea un *token*

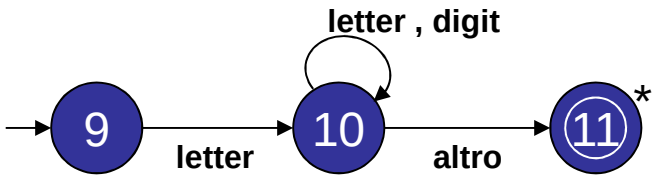
<nome-token, valore-attributo>

- *nome-token*: simbolo associato al token
- *valore-attributo*: che identifica il simbolo

Generatori di analizzatori lessicali

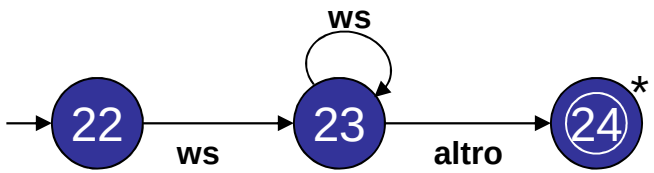
1. Trasforma ogni espressione regolare in un NFA
2. Combina gli NFA in un unico automa aggiungendo un nuovo stato iniziale con transizioni ε verso ognuno degli stati iniziali degli automi N_i relativi ai pattern p_i.
3. Riconosce il *prefisso piu` lungo che soddisfa* un pattern.
4. Se una stringa soddisfa piu` pattern, viene considerata un lessema del primo che compare nel programma Lex.
5. Alcuni analizzatori convertono l'NFA relativo a tutti i pattern in un DFA equivalente mediante la costruzione per sottinsiemi.

Automi per i token: esempi



return (**getToken**( ), **installID**( ))

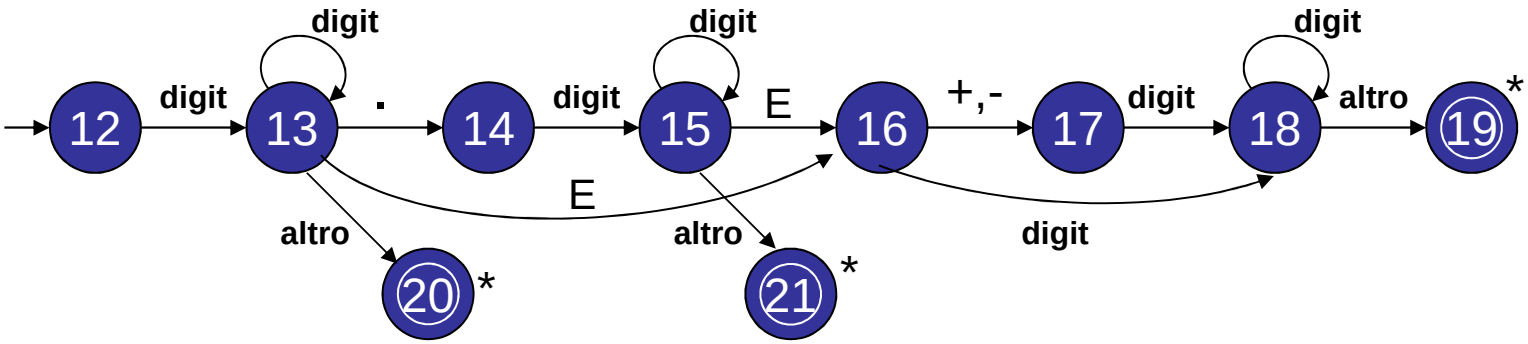
Automa per "**id**" e parole chiave: *letter_*(*letter_* | *digit*)*



Automa per i delimitatori: *ws**

$$ws \rightarrow (\ \text{blank} \mid \text{tab} \mid \text{newline}\ )^{+}$$

Automi per i token: esempi



return (**getToken**( ), **InstallNum**( ))

Automa per i numeri senza segno: *digits (. digits)?* (E [+-]? *digits*)