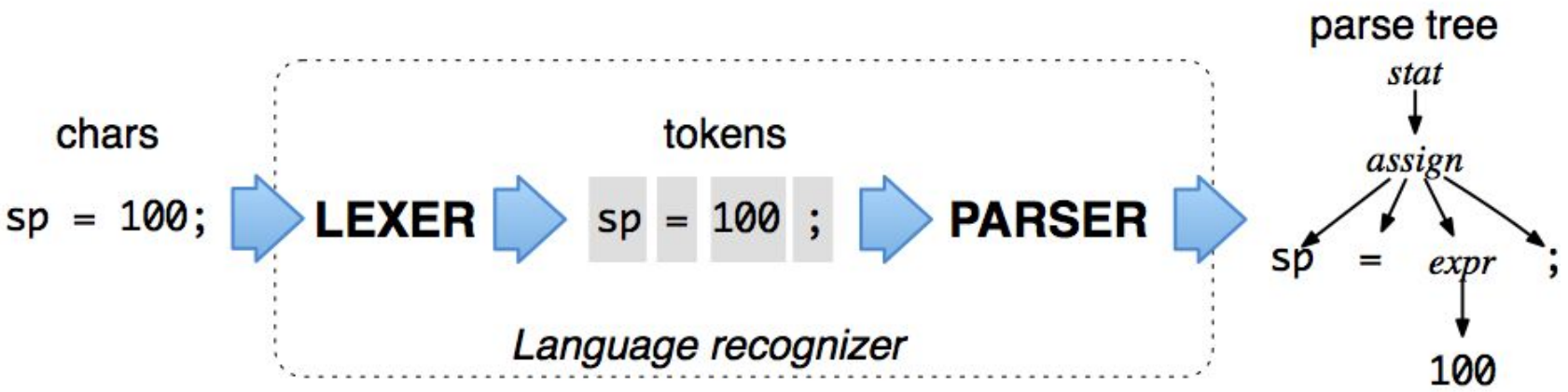
Automi e Linguaggi Formali

a.a. 2017/2018

LT in Informatica
3 Maggio 2018

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Analisi sintattica



- Il lexer prepara i token da passare al parser
- Il parser accetta uno stream dei token preparati dal lexer e cerca di riconoscere la struttura del testo
- La struttura del testo viene rappresentata con un albero sintattico

Lexer

- Legge la stringa di caratteri del testo e raggruppa i caratteri in sequenze chiamate token
- Le regole usano espressioni regolari per definire i token
- ANTLR costruisce un DFA per riconoscere i token

Sintassi delle regole del Lexer

- Il nome della regola deve iniziare con una lettera maiuscola
- `'letterale'` carattere o sequenza di caratteri come `'while'` o `'='`
- `[char set]` uno tra i caratteri specificati. `x-y` identifica l'insieme di caratteri compresi tra `x` e `y`. Esempi: `[abc]` o `[a-zA-Z]`
- `.` un carattere qualsiasi
- operatori di composizione: `|` (unione), `*` (chiusura di Kleene), `+` (una o più ripetizioni)
- parentesi per raggruppare le operazioni

Parser

- Il parser crea una rappresentazione ad albero della lista di tokens (struttura grammaticale)
- Albero: insieme di nodi e archi diretti (da padre a figlio), tale che un nodo (radice) non ha padri, e tutti gli altri hanno esattamente un padre. Foglie: nodi senza figli.
- Albero sintattico: nodo interno = regola, figli = simboli terminali
- Mostra l'ordine in cui eseguire le istruzioni del programma
- Il parser usa: grammatiche libere da contesto per definire le regole
- ANTLR usa un algoritmo chiamato LL(*) per costruire l'albero sintattico

Regole del Parser

- Il nome della regola deve iniziare con una lettera minuscola
- `'letterale'` carattere o sequenza di caratteri
- `nonterminale` simbolo non terminale della grammatica
- `TOKEN` token riconosciuto dal Lexer
- Le alternative sono separate da `|` :

```
superClass
  : 'extends' ID
  | // parola vuota
  ;
```

- la regola si può scrivere su più righe
- `//` inizia un commento
- si possono avere alternative vuote
- `;` termina la regola

- le parentesi raggruppano sottoregole alternative:
  - `(x | y | x)` fa il match con una sola tra `x, y, z`

Il ciclo while

- La guardia del while è un espressione booleana:
  - si devono definire gli operatori di confronto (minore, maggiore, uguale, ...)
- Il corpo del while deve essere identificabile in modo non ambiguo:

```
while a > b do
    sub b to a
    print a
end while
print b
```

NON AMBIGUO

```
while a > b :
    sub b to a
    print a
print b
```

AMBIGUO