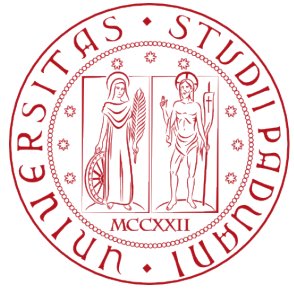
Automi e Linguaggi Formali

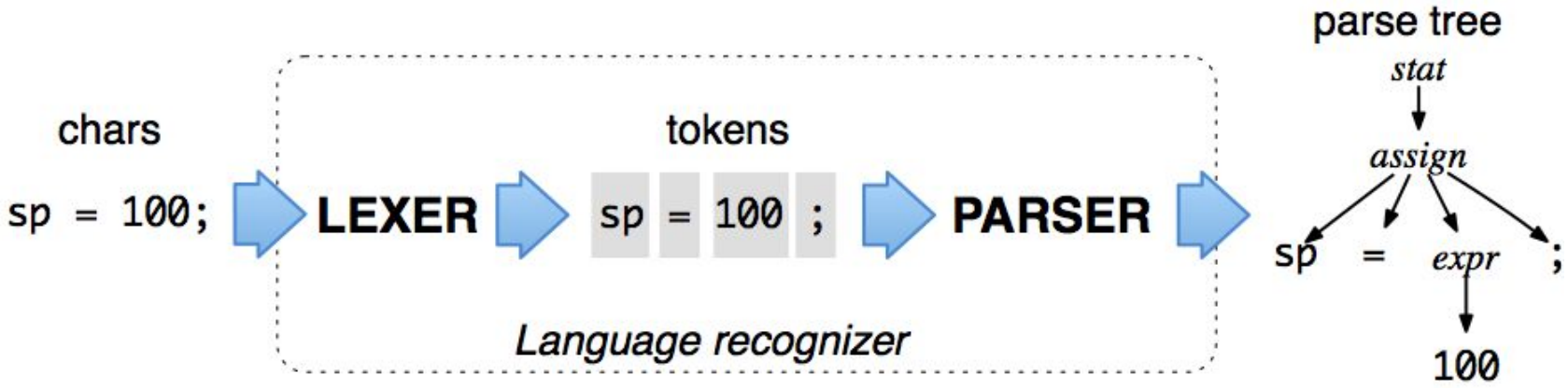
Seconda Esercitazione di Laboratorio

LT in Informatica
a.a. 2020/2021



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
- L'interprete usa l'albero sintattico per eseguire il programma

ANTLR v4

- Strumento automatico che permette di generare analizzatori sintattici di testi (parser)
- Sviluppato in Java permette di generare codice in C++, Java, C#, Python, Swift, Go, ...
- Supporta espressioni regolari, grammatiche EBNF e la generazione di alberi sintattici

Questo strumento può produrre in automatico il codice relativo a:

- Lexer: parte lessicale dove si definiscono i token e i separatori da ignorare
- Parser: parte grammaticale contenente le regole di produzione
- Visitor: parte semantica che definisce le azioni da eseguire sul testo

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), `~` (complementazione)
- parentesi per raggruppare le operazioni

Parser

- Il parser crea un albero sintattico dalla lista di tokens (struttura grammaticale)
- Mostra l'ordine in cui eseguire le istruzioni del programma
- Il parser usa una grammatica context-free 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`