# 2 approcci complementari al parsing

## top-down

- analisi a discesa ricorsiva
- analisi predittiva (LL(k))

## bottom-up

- lineare
- predittiva
- esistono altri metodi:

# PARSING SHIFT-REDUCE

L'idea generale è, data una stringa, trovare la sequenza di produzioni che possono portare ad essa.

2 op. possibili:

- **Shift**: (continua a leggere input, sposta in pila il prossimo token
- **Reduce** applica produzione e riduci pila. Individua stringa $\alpha$ tale che esiste $X \to \alpha$, esegui pop di tutti i caratteri di $\alpha$ ed esegui push di X (reduce), $\alpha$ è l'handle

Se dopo una reduce la pila contiene l'assioma di input allora è completato il parsing

in caso di scelte multiple come fare?
CONFLITTI

- reduce - reduce è possibile scegliere 2 riduzione
- shift - reduce

Es. dangling else, shift o reduce

grammatica contiene sviluppo

$S \rightarrow$ if E then S | if E then S else S'

Se esiste sequenza

if E then if E then S else S'

Si avrà nella pila

if E then if E then S

2 possibilità dato token else

1. Reduce: si sviluppa "if E then S", pila diventa "if E then S"
2. Shift le successive: pila cambia in "if E then if E then S else S"

Soluzioni:

- Regola aggiuntiva nella grammatica
- Obbligo di uso parentesi

per risolvere conflitti
- riconoscere l'handle
- decidere la produzione da usare nella riduzione (reduce-reduce)
- utilizzare strumenti per scegliere tra shift e reduce

Costruzione tavole LR (scan input da destra a sinistra)
- più numerose rispetto a parser LL (predittivo top-down)
- si possono costruire tasselle di produzione!

Per catturare meglio il contesto si mettono in pila STATI non token. Lo stato che affiora dalla pila permette di scegliere correttamente.

Utilizzo tavole action e goto

Tavole Action e Goto

Action [S, a]
descrive quale azione eseguire quando lo stato affiorante in pila è S e il prossimo token in input è il terminale a

Goto [S, X]
indica il nuovo stato da piazzare in cima alla pila (push) dopo la riduzione del non-terminale X, mentre lo stato affiorante è S, e serve per completare

- Shift
- reduce
- Accept
- report error

dal mantenimento dei simboli terminali e non nelle pila si usano STATI

ES

| state | Action | | Goto | | |
|---|---|---|---|---|---|
| | E | T | id | + | $ |
| 0 | $G_1$ | $G_2$ | $S_4$ | | |
| 1 | | | | S2 | ACC |
| 2 | | $G_3$ | $S_4$ | | |
| 3 | | | | $R_2$ | $R_2$ |

State: stato in cui ci si trova

GOTO: prossimo stato possibile

Action: prossima azione da fare shift / reduce

possibili azioni:

- Shift: inscrisce stato in pila (push)
- reduce: uno handle associato ad uno o più stati va in cima alla pila
- accept: termina con successo
- report error: termina con errore

Sy indica shift, nuovo stato y, mentre rx è reduce su produzione x

# LR parsing, esempio (secondo tabella slide 17)

input id + (id) $

| PILA | INPUT | Azione |
|---|---|---|
| $S_0$ | id + (id) $ | shift input, assumi $S_4$ in pila |
| $S_4 S_0$ | + (id) $ | reduce($S_4$, +) -> riduci, fai i pop di $S_4$ dalla pila applicando produzione 4) T -> id Goto($S_0$, T) dice prox stato = $S_2$ |
| $S_2 S_0$ | + (id) $ | reduce() = 2 2) E -> T Goto($S_0$, E) = 1 assumo $S_1$ |
| $S_1 S_0$ | + (id) $ | Shift($S_1$, '+') = 5, assumo S5 in pila |
| $S_5 S_1 S_0$ | (id) $ | shift, push 83 |
| ⋮ | | |
| $S_8 S_5 S_1 S_0$ | $ | Reduce 1, |

Se produzioni contengo >1 simbolo si eliminano >1 stati dalla pila

# Come costruire tavole action-goto?

- bisogna sapere riconoscere gli **handle** (sequenze di tale per cui si può applicare una produzione "inversa")

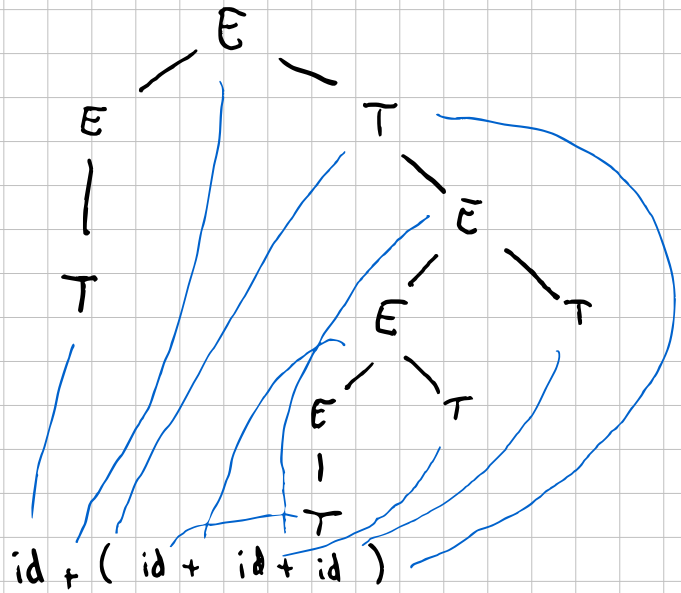
↓

data grammatica si costruisce automa a stati finiti, da lì si derivano tavole.

3 punti di vista per capire derivazioni destre

| Grammatica | Input |
|---|---|
| $E \to E+T \mid T$<br>$T \to id$<br>$T \to (E)$ | $id+(id+id+id)$ |

1) Albero sintattico



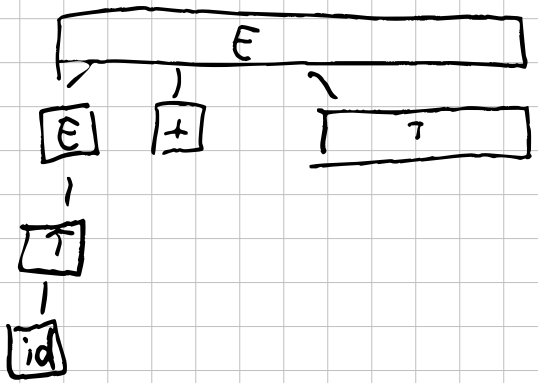
## 2) Derivazione destra

$\Rightarrow T + (id + id + id)$
$\rightarrow E + (id + id + id)$
$\rightarrow E + (T + id + id)$
$\rightarrow E + (E + id + id)$
$\rightarrow E + (E + T + id)$
$\rightarrow E + (E + id)$
$\rightarrow E + (E + T)$
$\rightarrow E + (E)$
$\rightarrow E + T$
$\rightarrow E$

L'analisi bottom up è derivazione destra in ordine inverso

## 3) Bottom up

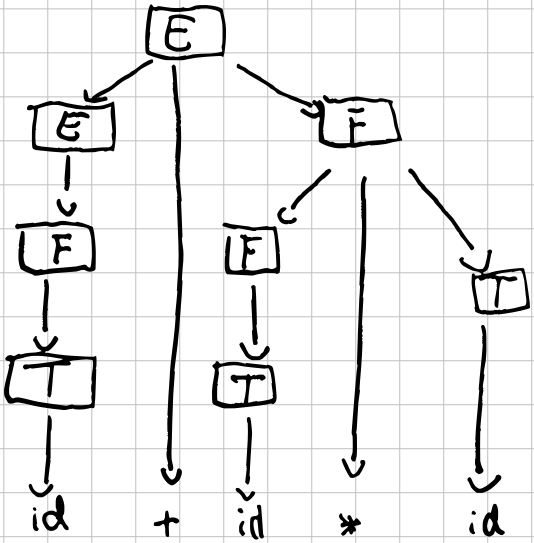
vengono eliminate con Shift e Reduce fino all'assioma (E)



# Come trovare handle?

- come sapere se è poi corretto?

E → F
E → E + F
F → F * T
F → T
T → id
T → ( E )




nel caso di produzioni semplici (es T → id) è l'unica produzione con "id" a destra, e da T a F

Quando bisognava applicare produzione F * T ?

=

# Costruzione Tavole Action e Goto

## • Introduzione punto "•"

Si introduce un punto per separare la produzione destra in due sotto-sequenze.
a sinistra vi sono elementi letti e impilati (shifted)
a destra elementi da analizzare

es. $E \to E \bullet + T$
Sono già stati analizzati elementi derivati da E e occorre accettare il input simbolo T

## • Grammatica aumentata

S' è il nuovo assioma, si aggiunge produzione $S' \to S$
È un simbolo che non compare in alcuna parte destra.
Si può sempre effettuare reduce conclusiva.

Per ogni produzione $A \to \alpha$ si considerano gli item ottenuti introducendo in tutte le possibili produzioni di $\alpha$ il punto.

Es $E \to E+T$

$$E \to \bullet E+T \to E \bullet +T \to E+ \bullet T \to E+T \bullet$$

# Chiusura di Set di item

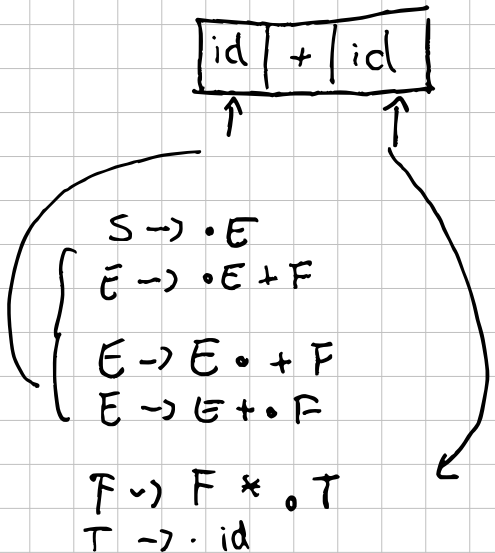
in generale non è possibile descrivere lo stato di un parser con un solo item perché in presenza di $\cdot X$ possono esistere vari elementi FIRST(X), o FOLLOW(X) in caso di annullamento.

Lo stato è caratterizzato da più elementi item, attraverso procedura di chiusura

Se il set contiene un item $A \to \alpha \cdot B\beta$ $B \in V_n$, allora aggiungere al set $B \to \cdot\gamma$ per ciascuna produzione $B \to \gamma$, $\gamma \in V^*$

Continuare su tutte le possibili produzioni del set finché possibile, aggiungendo produzioni aventi come parte sinistra NON terminali che appaiono in produzioni destre precedute dal punto.

$S \to E$ (s nuovo assioma)
$E \to F \mid E+F$
$F \to T \mid F*T$
$T \to id \mid (E)$



Ora quindi si costruiscono gli stati attraverso le grammatiche con il punto, così da poter analizzare il prossimo carattere

es. data la grammatica

$S' \to S$
$S \to aSc$
$S \to ac$
$S \to T$

$T \to bc$
$T \to bTc$

$S_0$ parte dallo stato 0
$S_0$ • inizio assioma $S' \to S$
• essendo S

$S_0$

$S' \to \cdot S$
$(+) S \to \cdot aSc \mid \cdot ac \mid \cdot T$
$(+) T \to \cdot bc \mid \cdot bTc$

$S_1$ costruiscono gli altri stati partendo da tutti i caratteri (terminali e non) che può accettare

$S_1$

$S \to a \cdot Sc \mid a \cdot c$
$S \to \cdot aSc \mid \cdot ac \mid \cdot T$
$T \to \cdot bc \mid \cdot bTc$

'a'

'T'

$S_2$

$S \to T \cdot$ (reduce)

$S_0$

$S_1$

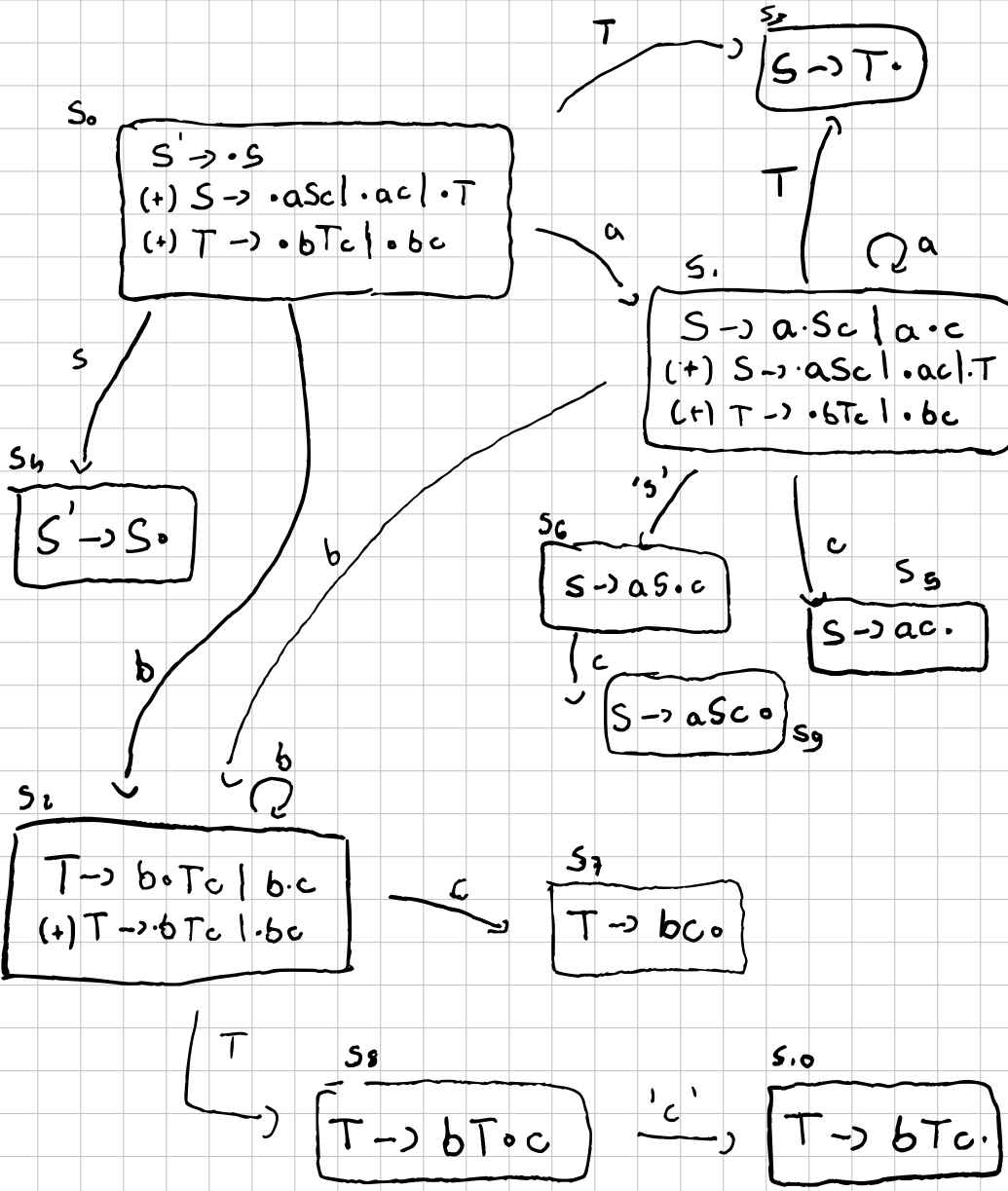
$T \to b \cdot c \mid b \cdot Tc$
$(+) T \to \cdot bc \mid \cdot bTc$

'b'

S

$S' \to S \cdot$

fare la stessa cosa con tutti gli altri stati


S0
S' → ·S
(+) S → ·aSc | ·ac | ·T
(+) T → ·bTc | ·bc
S
S4
S' → S·
a
S1
S → a·Sc | a·c
(+) S → ·aSc | ·ac | ·T
(+) T → ·bTc | ·bc
T
S2
S → T·
a
S5
S → ac·
'S'
S6
S → aS·c
c
S9
S → aSc·
b
b
S3
T → b·Tc | b·c
(+) T → ·bTc | ·bc
c
S7
T → bc·
T
S8
T → bT·c
'c'
S10
T → bTc·

# Costruzione tavole action e goto

- Un arco tra 2 stati $S_a$ e $S_b$ etichettato con un simbolo terminale x equivale a passare da $S_a$ a $S_b$ quando input è x (SHIFT), si inserisce $S_b$ in pila
- goto [s, U] dice quale stato inserire in pila quando lo stato affiorante è s e il simbolo a sinistra della produzione oggetto di reduce è U
- Quando si giunge in uno stato con . alla fine bisogna eseguire reduce: si tolgono tanti stati dalla pila quanti sono i simboli a destra della produzione e si inserisce un nuovo stato in pila.

| Stato | Action | | | | goto | |
|---|---|---|---|---|---|---|
| | a | b | c | $ | S | T |
| $S_0$ | $S_1$ | $S_2$ | | | 4 | 3 |
| $S_1$ | $S_1$ | $S_2$ | $S_5$ | | 6 | 3 |
| $S_2$ | | $S_2$ | $S_7$ | | | 8 |
| $S_3$ | reduce | $S \to T$ | | | | |
| $S_4$ | | | | ACC | | |
| $S_5$ | reduce | $S \to aC$ | | | | |
| $S_6$ | | $S_9$ | | | | |
| $S_7$ | reduce | $T \to bC$ | | | | |
| $S_8$ | | $S_{10}$ | | | | |
| $S_9$ | reduce | $S \to aSc$ | | | | |
| $S_{10}$ | reduce | $T \to bTc$ | | | | |