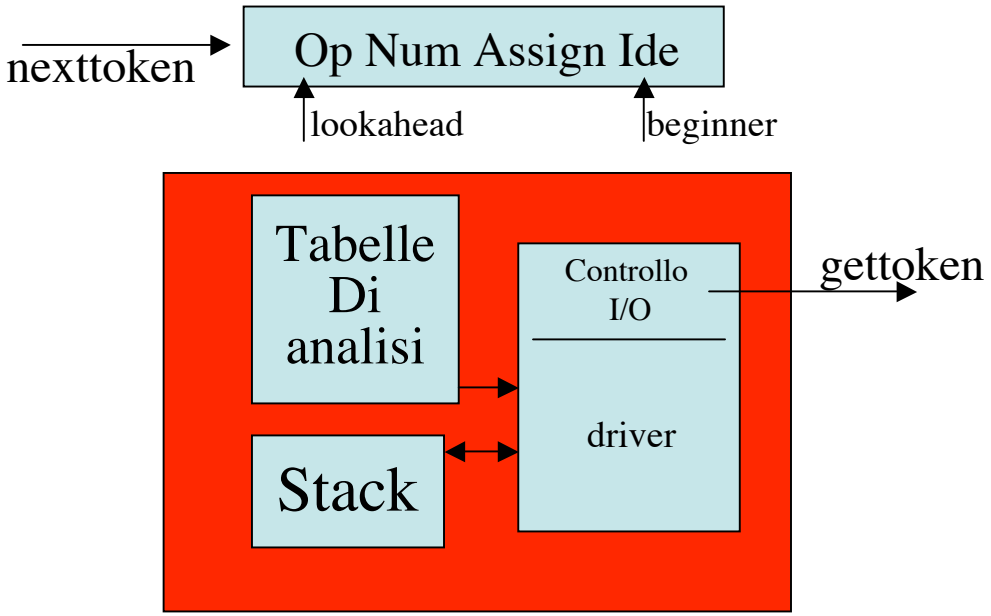
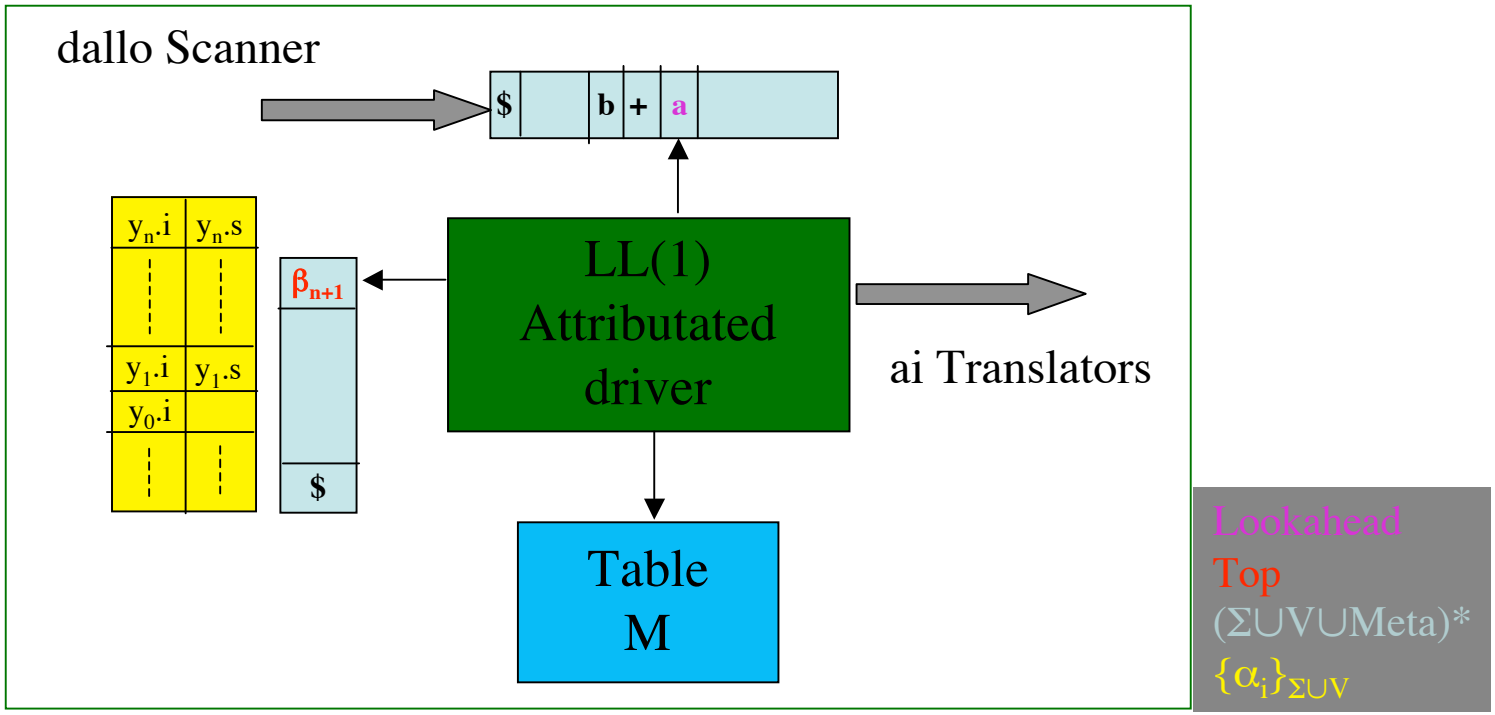
LL(1)-Attributated: View



Source: Marco Bellia - Dip. Informatica, Univ. Pisa

LL(1)-Attributated: Structure



Applicazione della produzione (in schema di traduzione): $y_0 ::= \beta_1 y_1 \ldots \beta_n y_n \beta_{n+1}$

dove $\beta_j \equiv y_j.i = E_j(y_0.i, y_1.i, y_1.s, \ldots, y_{j-1}.i, y_{j-1}.s)$ per arbitrarie epressioni $E \in Meta$

**** Stack dati mostra:

- risultato derivazioni dei simboli y_j,
- calcolo delle azioni β_j ($j \leq n$), e
- degli attributi (ereditato e sintetizzato) $y_j.i$ e $y_j.s$ dei simboli attraversati,

**** Stack di controllo mostra:

- azione β_{n+1} da valutare relativa al calcolo dell'alltributo x.s

Source: Marco Bellia - Dip. Informatica, Univ. Pisa

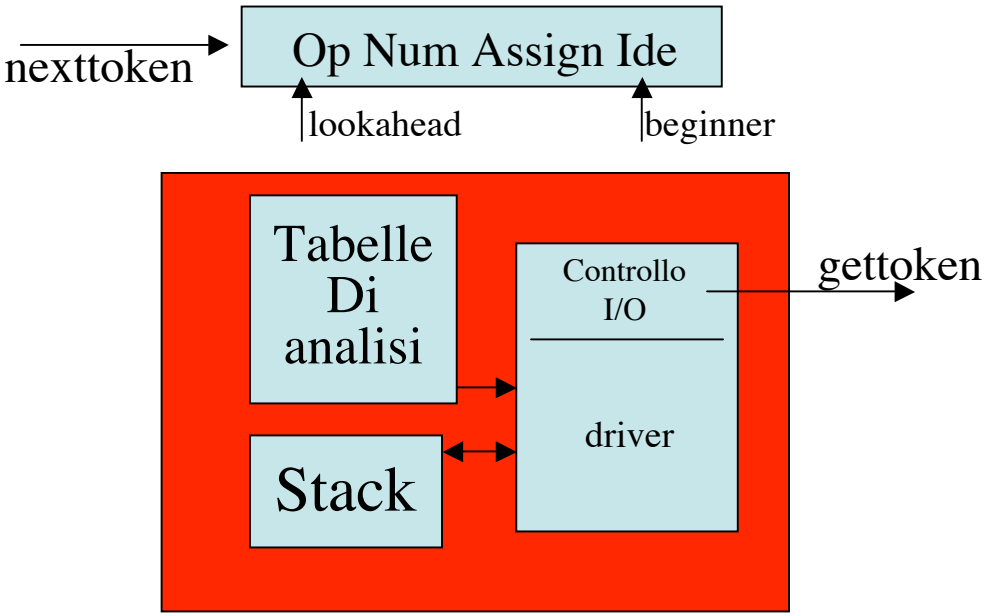
LL(1)-Attributated: Driver

- top = lookahead = $: stop.
- top = lookahead ≠ $: C<-pop;
  D(top,s)<-null;
  lookahead:= nexttoken.
- top≡y_0∈V: C<-pop;
  C<-push($\beta_1 y_1 \ldots \beta_n y_n \beta_{n+1}$).
  dove M(y_0,lookahead)≡ $y_0 ::= \beta_1 y_1 \ldots \beta_n y_n \beta_{n+1}$
- top≡β_j∈Meta: D<-push(<v,undef>). *se* j≤n
- top≡β_j∈Meta: D(top-n,s)<-v; se j=n+1
  D<-pop(n).
  dove $\beta_j \equiv y_j.R = E_j$ *con* $j = j \bmod (n+1)$, $R \in \{i,s\}$
  $v = Sem_{Meta}(E_j[D(top-(j-p-1),R)/y_p.R]_{p<j,\ R\in\{i,s\}})$

Operazioni su stack A:
- D: stack per attributi - due stack accoppiati Di per ereditati, Ds per sintetizzati
- C: stack di controllo
- accesso: A(top-m) = m-esimo elemento dal top di stack A -- nota: A(u,r) sta per Ar(u)
- inserimento: A<-push(e) = inserisce e come top di A
- rimozione: A<-pop(m) = rimuove m elementi a partire dal top -- nota: pop sta per pop(0)

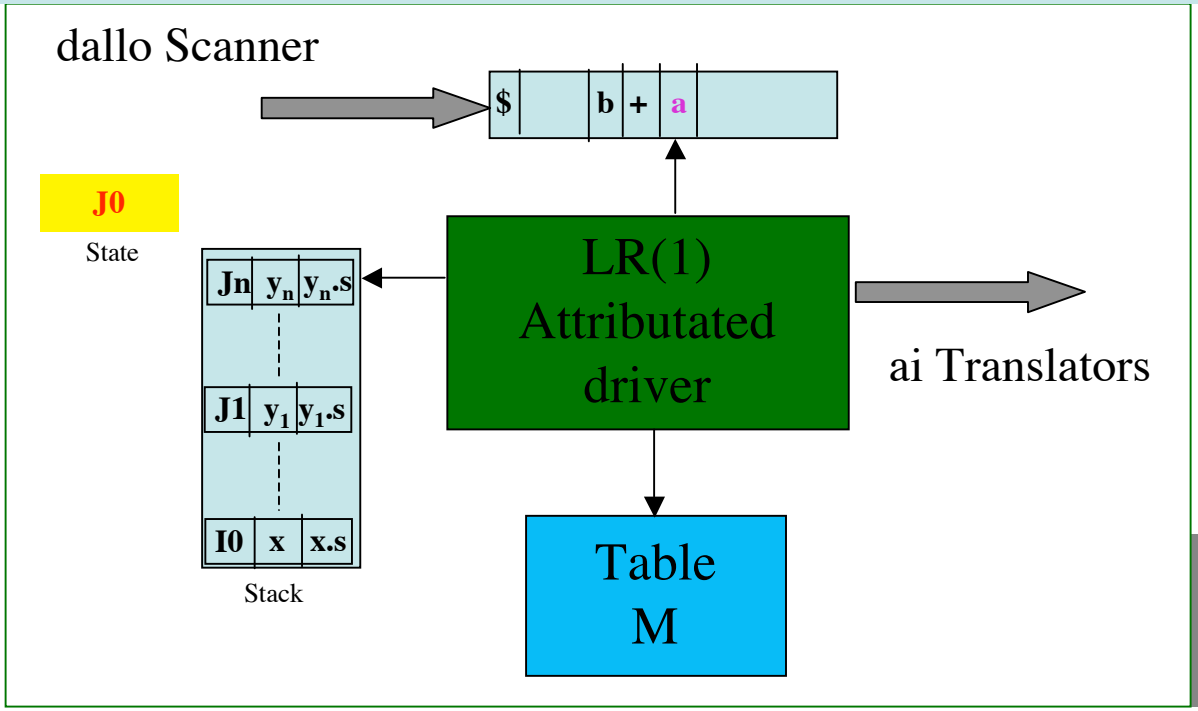
Source: Marco Bellia - Dip. Informatica, Univ. Pisa

LR(1)-Attributated: View



Source: Marco Bellia - Dip. Informatica, Univ. Pisa

LR(1)-Attributated: Structure



Applicazione della produzione (s-attributata): $y_0 ::= y_1 \ldots y_n \beta$

dove $\beta \equiv y_0.s = E_j(y_1.s, \ldots, y_n.s)$ per arbitrarie epressioni $E \in$ Meta

**** Stack - unico: contiene

- stato Ji dell'automa di viable prefix per LR(LALR-SLR), e
- simbolo attraversato o ridotto e
- attributo (sintetizzato) $y_j.s$ dei (figli) simboli attraversati,

+ Un registro State per il "futuro" Top

+ β è valutata ad ogni riduzione e il valore calcolato è posto nello stack

Source: Marco Bellia - Dip. Informatica, Univ. Pisa

LR(1)-Attributated: Driver

ACTION(State,Look) =
- S/k: A <- push(<State, Look, SymTab(Look,s)>);
  State <- k.
- R/q: A(top-n) <- <A(top-n,I), y_0 , v>;
  A <- pop(n-1);
  State <- GOTO(A(top,I),A(top,Y))
  dove q≡ $y_0 ::= y_1 \ldots y_n \beta$
  $\beta \equiv y_0.s=E$
  $v=Sem_{Meta}(E\,[A(top-(n-p),s)/y_p.s]_{p \in 1..n}$
-Acc: stop.

Operazioni su stack A:
-Assunzioni: State *registro* stato, Look *sta per* Lookahead,
Usiamo la SymbolTable: contiene attributo ”s“per i token -- SymTab(Look,s)
- selettori: Sia A(top)=<J,x,a> una tripla. Allora A(top,I)=J, A(top,Y)=x, A(top,S)=a
- accesso: A(top-m) = m-esimo elemento dal top di stack A
- inserimento: A<-push(e) = inserisce e come top di A
- rimozione: A<-pop(m) = rimuove m elementi a partire dal top -- nota: pop sta per pop(0)

Source: Marco Bellia - Dip. Informatica, Univ. Pisa