BNF, the standard form

(* the start symbol: E; Follow(E) ={$} initially. *)

(* ε , the empty string; First(ε)={ ε } initially. *)

(* Terminal: {+, *, (, id }; First(+)={+}, … , initially.*)

(* NT: {E, E' ,T, T' , F}; First: LHS, Follow: RHS. *)

$E \rightarrow TE'$

$E' \rightarrow +TE'$

$E' \rightarrow \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT'$

$T' \rightarrow \varepsilon$

$F \rightarrow (E)$

$F \rightarrow id$

(* How ? Do … until no change ! *)

(1)-(a)(* A->B, B->C, C->t, First(A)=First(B)=First(C)=First(t)={t} *)

FIRST(E) = FIRST(T) = FIRST(F) = {(,id }

FIRST(E') = {+, ε }

FIRST(T') = {*, ε }

(1)-(b)

FOLLOW(E) = {$} $\cup$ {) } = {) , $ }

FOLLOW(E') = FOLLOW(E) = {) , $ }

FOLLOW(T) = FIRST(E') = {+, ε } = {+, ε }- { ε } $\cup$ FOLLOW(E)
= {+}$\cup${) , $ } = { +,) , $ }

FOLLOW(T') = FOLLOW(T) = { +,) , $ }

FOLLOW(F) = FIRST(T') ={*, ε } = {* ε }- { ε } $\cup$FOLLOW(T)
= {*} $\cup${ +,) , $ } = { +, *,) , $ }

(2)-(a) (* A->B, B->C, C->t, First(t)=t, First(C) = First(t)={t},
First(B)=First(C), First(A)=First(B) *)

FIRST(F) = {(,id }

FIRST(T) = FIRST(F) = {(,id }

FIRST(E) = FIRST(T) = FIRST(F) = {(,id }

FIRST(E') = {+, ε }

FIRST(T') = {*, ε }

2-(b) the same as 1-(b).

(3)-(a)+(b) (* First, Follow, at the same time ! *)

Then, the predictive set for each production rule.

Predict(A→X1 ⋯Xm)＝

if $\lambda \in$ First(X1 ⋯Xm)

(First(X1 ⋯Xm)－λ)∪Follow(A)

else

(First(X1 ⋯Xm))

E → TE' PS(E→ TE')=FIRST(T E')= {(,id }

E' → +TE' PS(E' → +TE')=FIRST{+ TE'}={+} !!

E' → ε PS(E' → ε)=FOLLOW(E')= { ）, $ } !!

T → FT' PS(T→ FT')=FIRST(F T')= {(,id }

T' → *FT' PS(T' → *FT')=FIRST(*FT')={*} !!!

T' → ε PS(T' → ε)=FOLLOW(T')= { +, ）, $ } !!!

F → (E) PS(F → (E))= FIRST((E))={(} !!!!

F → id PS(F → id)= FIRST(id)={id}!!!!

Finally, the parsing table based on the predictive set for each rule.

For each production A → α of the grammar

For each terminal a in FIRST(α), add A → α to M[A, a].

If ε is in FIRST(α), add A → α to M[A, b] for each terminal b in FOLLOW(A).

If ε is in FIRST(α) and $ is in FOLLOW(A), add A → α to M[A, $].

LL(1) ??

LL(1) contains exactly those grammars that have disjoint predict sets for productions that share a common left-hand side