

9. Syntax Error Handling

Goal

- to maximize input text
- to report actual errors

9.1 Syntax Error

Syntax Error

- actual Error made by programmer
 - depends on the Language
- formal definition of actual Errors detected by a particular parser
 - reflects the characteristics of parsing technique

- parser - defined error in $\omega = xy$

pair($|x|, 1:y$) or $1:y$,
if x is the longest prefix of ω

- Let M : pda with initial stack contents γ_s

- M has the correct prefix property if the statement $\gamma_s | xy\$ \xRightarrow{*} \gamma | y\$$ can be true in M only if x is a prefix of some sentence in $L(M)$

- M has the longest correct prefix property if M has correct prefix property and for any terminal string xy in which x is a prefix of some sentence in $L(M)$ there is a stack string γ s.t.

$$\gamma_s | xy\$ \xRightarrow{*} \gamma | y\$ \text{ in } M$$

- Any computation on a nonsentence can be continue up to the parser-defined error.

[Theorem 9.1]

For $k \geq 0$, any LR(k) prser of a reduced grammar has the correct prefix property.

proof)

Let M : LR(k) parser of G .

Consider a computation

$$[\$] | xy\$ \xRightarrow{*} \varphi | y\$ \text{ in } M$$

Here

$$\varphi = [\$][\$X_1] \dots [\$X_1 \dots X_n], n \geq 0$$

where $X_1 \dots X_n$: viable prefix of G .

Moreover

$X_1 \dots X_n$ derives x in G , by L.6.29, L.6.49

As any terminal string derived by a viable prefix of a G is a prefix of some sentence.

We conclude theorem.

□ Q.E.D.

[Theorem 9.2]

For $k \geq 0$, any LL(k) arser of a reduced grammar has the correct prefix property.

Proof)

Let M : canonical LL(k) of G .

Consider a computation

$$[\$][\$] | [\$s] | xy\$ \xRightarrow{*} \varphi | y\$ \text{ in } M$$

Here

$$\text{stack string } \varphi = [\$][\$X_1] \dots [\$X_1 \dots X_n] n \geq 0$$

s.t. S derives $xX_n \dots X_1$ by 8.30

As G is reduced x is a prefix of some sentence in $L(G)$

□ Q.E.D.

Let M : pda with input alphabet 2

ω : nonsentence

Φ : error configuration.

- M detects an error in ω at Φ if M has an ω a com-putaion that ends with Φ .

- M detects the parser-defined error in ω if M detects an error in ω at a configuration of the form $\$ \gamma / y \$$ in which $1:y$ is the parser-defined error in ω .

[Fact 9.3]

Let M : pda that defects an error in every nonsense and has longest correct prefix property.

Then

M detects the parser-defined error in every nonsense.

[Theorem 9.4]

An LR(0) or LL(1) parser of a reduced grammar detects the parser-defined error in every nonsense.

proof)

Let M : LR(1) parser of G .

M has the correct prefix property (T.9.1)

Let $xy \in \Sigma^*$ s.t. x is a prefix of some sentence in

$L(M) = L(G)$ for some $y', xy' \in L(M)$

$\$ [\$] / xy \$ \xrightarrow{*} \$ \phi y \$$ in M

if $x = \epsilon$: $\$ \phi / y \$$: initial conf.

otherwise : $x = x' a xy' \in L(M)$

there is $\$ [\$] / x' a y' \$ \xrightarrow{*} \$ \phi / a y' \$ \Rightarrow \$ \phi / y' \$$

As the length of lookahead = 1

$\$ [\$] / x' a y \$ \xrightarrow{*} \$ \phi' / a y \$ \Rightarrow \$ \phi / y \$$ in M

□ Q.E.D.

- LL(k), LR(k) s.t. $k > 1$.
 - declare error and halt before the entire correct prefix has been consumed.
- simple precedence parser (bottom up parser) non-deterministic shift-reduce parser (bottom up parser)
 - not possess the correct prefix property.

9.2 Error Recovery in SLL(1) Parsers

- Error detection in Recursive Descent Parser.

case 1 : beginning of procedure A .

current symbol $\neq \text{First}_1(A \text{ Follow}_1(A))$

- detected begin of the procedure A .
- scan input string until legal follower of the A .

case 2 : having parsed prefix α

$A \rightarrow \alpha a \beta$, current symbol $\neq a$

- search remaining input for $\text{First}_1(\beta \text{ Follow}_1(A))$
- scan input until $\text{First}_1(\beta \text{ Follow}_1(A))$

⇒ Problem : unanalyzed program text.

- Parsing within Error Procedure.
 - search for starters of some nonterminals
 - corresponding parsing procedure is called.

⇒

$b \in \text{First}_1(\beta)$ cause call for procedure A in the procedure error

only if B is in some reasonably defined subset of the set of all nonterminals, and other nonterminals, and for all other nonterminal C in this set, $b \notin \text{First}_1(C)$

- X begins A (sec 5.5)

$A \rightarrow \alpha X \beta$, where $\alpha \xrightarrow{*} \epsilon$

In recovering Process, calling A capture all those possibilities that arise from calling X

[Fact 9.5]

For any reduced LL(1) grammar, the relation begins is partial order

- maximal nonterminals
 - those that do not appear as leftmost components in the pairs in begins

- **Error Handling Procedure**
 - Computing all sets $First_1(B)$

Where B is a maximal nonterminal
 - Pairwise intersection of sets

: those symbols that are no intersection $First_1(B) \cap First_1(C)$ where C is another maximal nonterminal
- **Error Recovery in R.D.P not loop forever any input string**
 - Assume : Parsing Procedure A : error detect

→ Procedure error called.
 - recovery action calls procedure B .

$b \in First_1(B)$: shift b .
 - recovery action back to procedure A .

left-recursive nonterminal.

9.3 Error Recovery in LR(1) Parsers

Phrase-level recovery

- Let M : LR(0)-based LR(1) parser
 - Error conf. $\Phi = \$q_0 \dots q_m / a_1 \dots a_n$

Action[q_m, a_1] = "error"
 - Error phrase

$q_{i+1} \dots q_m / a_1 \dots a_{j-1}$
 if G has nonterminal $s.t.$ $Goto[q_i, A] \neq \emptyset$
 Action[$Goto[q_i, A], a_j$] \neq "error"

 - q_i : recovery state
 - A : (reduction) goal
 - $Goto[q_i, A]$: goal state
 - a_j : recovery symbol
- $\Rightarrow \$q_0 q_1 \dots q_i q_A / a_j \dots a_m$
 $q_A = Goto[q_i, A]$

SLL(1) Parser

- **Error Conf.**

$\Psi X/y\$ \quad I:y\$ \notin First_1(X\Psi^R\$)$
 - **Recovery Strategy**
 - Pop X from the parsing stack.
 - Scan the remaining inputstring $y\$$ until
 - ① $\$$ or $First(\Psi^R\$)$
 - ② a symbol that belongs to exactly one set $First_1(B)$ where B : maximal nonterminal with respect to the relation begins.
- ① $\rightarrow \$\Psi/z\$$ $\left\{ \begin{array}{l} \text{accepting Conf.} \\ \text{at least one parsing action whenever} \\ Z \neq \epsilon \end{array} \right.$
- ② $\rightarrow \$\Psi B/z\$$
 can proceed at least one step

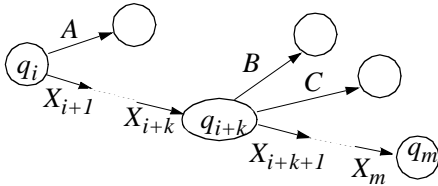
- **Error conf.** $\$q_0 q_1 \dots q_m / a_1 \dots a_n$

feasible reduction goals
 $F(q_{i+1} \dots q_m)$
 $= \{ A \mid A \text{ is a reduction goal of error phrase } q_{i+1} \dots q_m / a_1 \dots a_{j-1} \text{ for some } j,$
 and
 $A \xRightarrow{*} X_{i+1} \dots X_m z \text{ for some } z \in \Sigma^*$
 where X_k is the entry symbol of q_k ,
 $k = i+1, \dots, m.$
 $\}$
 for all $i = 0, \dots, m-1.$
- $(q, B \rightarrow \alpha \bullet \beta)$ on-string-goes-to $Goto(q, \alpha)$
 $(q, B \rightarrow \alpha \bullet \beta)$ reduces-to (q, B)
 $(Goto(q, \alpha), B)$ symbols-in $(q, A \rightarrow \alpha B \bullet \beta)$
 cf : sect 7.5.

[Theorem 9.6]

Nonterminal A is in $F(q_{i+1} \dots q_m)$ iff

q_m on-string-goes-to⁻¹ (reduce-to symbol-in)*
reduces-to(q_i, A) \square



For α, β , $A \rightarrow X_{i+1} \dots X_{i+k} B \alpha$,
 $C \rightarrow X_{i+k+1} \dots X_m \beta$.

q_m on-string-goes-to⁻¹ ($q_{i+k}, C \rightarrow X_{i+k+1} \dots X_m \beta$)
 $\therefore C$: feasible reduction goal for error phrase include-
ing $q_{i+k+1} \dots q_m$ /

Moreover,

if $B \xRightarrow{m}^+ C \gamma$ for some γ ,

then $(q_{i+k}, C \rightarrow X_{i+k+1} \dots X_m \beta)$ (reduce-to symbols-
in)* reduce-to(q_i, A)

$\therefore A$ is feasible for error phrases includeing $q_{i+1} \dots q_m$ /

• Search Error Conf. for an error phrase

- Let (i, j) segment $q_{i+1} \dots q_m$ / $a_1 \dots a_{j-1}$ in error conf.
 $\$q_0 \dots q_m$ / $a_1 \dots a_n \$$

- Search $(m, 1) \rightarrow (0, n)$
whenever (i_1, j_1) precedes (i_2, j_2)
 $i_1 > i_2, j_1 < j_2$

- Until unique important feasible reduction goal is found.

• No candidate error phrase.

- When the parser has recognized a complete sentence but input string is remained.

- restart action :

- Simply delete the topmost stack symbol.
- Used a terminal as a recovery symbol only if it is shiftable at the goal state
: not cause the error recovering parser to loop forever.

Local correction

- generalizing the concept of a recovery action s.t. terminal and empty string are allowable reduction goal.

$\langle b \rangle \rightarrow b$: insertion, replacement.

$\langle b \rangle \rightarrow \langle \epsilon \rangle b$: deletion

$\langle \epsilon \rangle \rightarrow \epsilon$

- case of terminal reduction goal
: as in the nonterminal reduction goal

- case of empty reduction goal

- no goal state

- recovery symbol : shiftable terminal at the recovery state.

- input portion of the error phrase ≤ 1 .
stack portion of the error phrase

\rightarrow use of terminal and empty reductions goals for insert, delete, replacement of a single terminal.

- For higher performance

- respective cost of each allowable local correction is evaluated.

- only one with lowest cost correction is chosen.

Forward-move recovery

- Recovery state for special grammar symbol

Let $M : LR(0)$ machine, $X \in V$.

- recovery state X
 - . union of all states in M that have X as the entry symbol
- successor for recovery state
 - . computed ordinary states in the $LR(0)$ machine until no new state are obtain
- initial recovery state has a transition to the recovery state for X .

• **Forward-move recovery**

- at an error conf.
 - whole stack content
 - initial recovery state
- parsing proceeds.
- if conflict in recovery state or reduce action
 - (whole parsing stack)
 - then a new start will be invoked by emptying stack
 - (except recovery stack)
- if new error is detected at a recovery state then a new error is reported and some recovery process is repeated.

• **Error recovery table**

- $[\epsilon]$: initial recovery state for all $X \in V$
- $[\epsilon X]$: recovery state for X
- $[\epsilon X \alpha]$: for all $\alpha \in V^*$ the α -successor of recovery state for X .

- $R\text{Goto}[[\epsilon \zeta], X] = [\epsilon \zeta X]$ for all $\zeta \in V^*, X \in V$

- $R\text{Action}$: parsing action in the E.R.T

- the Entries of $R\text{Action}$: $[\epsilon X]$ if all states with entry symbol X in the parser have the same parsing action on terminal a then the state $[\epsilon X]$ has this action a

$R\text{Action} [[\epsilon X], a] = \text{"conflict"}$
if not unique action

$R\text{Action} [[\epsilon], a] = \text{"shift}[\epsilon a]$ ", $a \in \Sigma$
 $R\text{Action} [[\epsilon], \$] = \text{"halt"}$

9.4 Error Reporting**Error Message**

- declarative : location of error
automatic error handling
language, Grammar Info
- operational : recovery action
by recovery routine.

SLL(1) parsing error

- 1) shift error : if a terminal can't be shifted topmost element in the stack.
→ prediction of a terminal
∴ expected symbol : terminal to be shifted
- 2) produce error : if a nonterminal can't be expanded topmost element in the stack.
→ prediction of a nonterminal
∴ expected symbol : starter of nonterminal to be expanded.

LR(1) Parsing

Let $M : LR(0)$ based $LR(1)$ parser.

$X \in V$ is expected in state $q = \text{varid}(\gamma)$
if q contains an item of
the form $[A \rightarrow \alpha \bullet X \beta]$

X is expected in Conf. $\$ \eta \mid x \$$ of M
if X is expected in $\eta : I$

valid continuation of $LR(1)$

if $X : \text{expected in Conf.}$
then X (or derived string from X) is
valid continuation of the stack.

Other valid continuation
lookahead symbols of those reduction
action that apply to stack string

Error detection state

topmost state q in the stack in the error
configuration
if topmost state has $[A \rightarrow \omega \bullet]$
expected symbols alone do not form
a sufficient basis for error msg.

if default reductions are used
this situation does not occur
error msg. can be based solely on
the expected symbol.

If only essential symbols are used, this reduces
the number of expected symbols.

symbol X expected in state q is essential
if q contains an item of the form
 $[A \rightarrow \alpha \cdot X \beta]$, $\alpha \neq \epsilon$

Essential expected symbols capture all
the information about the valid contin-
uations obtained by all expected symbols

X is terminallike

if either (1) X is a terminal,
or (2) an nonterminal that has
only unit rules with
terminal like right hand side
Every rule for X is the form $X \rightarrow Y$
where Y is terminallike symbol
name for a set of terminals

Nonterminallike

Maximal essential symbol

- not derived by unit rules from other essential
symbols in that state.
- removing redundancy between the essential
symbols.

Automatic Generation of D.E.M

Case 1 : no error detection state q contains
an essential item of the form $[A \rightarrow \omega \bullet]$

suppose $X_1 \dots X_n$: maximal essential
symbols expected in q

if each symbol X_i : terminallike
then “ X_1 expected” (case $n = 1$)
“ $X_1 \dots X_{n-1}$ or X_n expected”
(case $n \geq 1$)

if each symbol X_i : nonterminallike
then “No X_1 can start with this”
(case $n = 1$)
“Neither $X_1 \dots X_{n-1}$ nor X_n can
start with this” (case $n > 1$)

if $X_1 \dots X_k$ ($k \geq 1$) : terminallike and
 $X_{k+1} \dots X_n$ ($k \leq n$) : nonterminallike
then
“ $X_1 \dots X_{n-1}$ or start of X_n expected”
(case $K = n-1$)
“ $X_1 \dots X_k$ or start of $X_{k+1} \dots X_n$, or
 X_n expected” (case $k < n-1$)

Case 2 : Error detection state q contains
essential item of the form $[A \rightarrow \omega \bullet]$

if the entry symbol X is terminallike

“ X can't be followed by this”

otherwise

“No complete X can be followed this”

cf) if default action is allowed this cannot
occur in any error detection state.

if default action is not allowed

- a number of error detection state

- many lookahead symbols for one
reduce state

- not all lookahead symbol are valid
continuation of stack : no way to
choose the correct one.