# GLL – a simple example

A brief overview of the GLL approach and the functions used in the following example is given in GLL Algorithm Sketch and Terminology on our GLL parsing webpage `www.rhul.ac.uk/computerscience/research/csle/researchareas/gllparsers.aspx`. Below is a GLL parser for the grammar

$$\begin{array}{lcl} S & ::= & a\ d \mid A\ d \\ A & ::= & A\ a \mid \epsilon \end{array}$$

---

read the input into $I$ and set $I[m] := \$$
create GSS node $u_o := (L_0, 0)$
set $c_I := 0$; $c_U := u_0$; $c_N := \$$
set $\mathcal{R} := \emptyset$; $\mathcal{P} := \emptyset$;
**for** $0 \leq j \leq m$ { set $\mathcal{U}_j := \emptyset$ }
**goto** $L_S$

$L_0$: **if** $(\mathcal{R} \neq \emptyset)$ { remove $(L, u, i, w)$ from $\mathcal{R}$
$c_U := u$; $c_I := i$; $c_N := w$; **goto** $L$ }
**else if** (there is an SPPF node $(S, 0, m)$) report success
**else** report failure

$L_S$: **if** $(test(I[c_I], S, ad))$ $add(L_{S_1}, c_U, c_I, \$)$
**if** $(test(I[c_I], S, Ad))$ $add(L_{S_2}, c_U, c_I, \$)$
**goto** $L_0$
$L_{S_1}$: $c_N := getNodeT(a, c_I)$; $c_I := c_I + 1$
**if** $(test(I[c_I], S, d))$ { $c_R := getNodeT(d, c_I)$
$c_I := c_I + 1$; $c_N := getNodeP(S ::= ad., c_N, c_R)$ }
**else goto** $L_0$
$pop(c_U, c_I, c_N)$; **goto** $L_0$
$L_{S_2}$: $c_U := create(R_{A_1}, c_U, c_I, c_N)$; **goto** $L_A$
$R_{A_1}$: **if** $(test(I[c_I], S, d))$ { $c_R := getNodeT(d, c_I)$
$c_I := c_I + 1$; $c_N := getNodeP(S ::= Ad., c_N, c_R)$ }
**else goto** $L_0$
$pop(c_U, c_I, c_N)$; **goto** $L_0$

$L_A$: **if** $(test(I[c_I], A, a))$ $add(L_{A_1}, c_U, c_I, \$)$
**if** $(test(I[c_I], A, \epsilon))$ $add(L_{A_2}, c_U, c_I, \$)$
**goto** $L_0$
$L_{A_1}$: $c_R := getNodeT(a, c_I)$
$c_I := c_I + 1$; $c_N := getNodeP(A ::= a., c_N, c_R)$
$pop(c_U, c_I, c_N)$; **goto** $L_0$
$L_{A_2}$: $c_R := getNodeT(\epsilon, c_I)$
$c_N := getNodeP(A ::= ., c_N, c_R)$
$pop(c_U, c_I, c_N)$; **goto** $L_0$

**Notation usage**
$m$ – length of the input string
\$ – end-of-string symbol
$I$ – array of length $m$ containing the input string and \$
$c_I$ – current input position, an integer between 0 and $m$
$c_U$ – the current GSS node
$c_N$ – the current SPPF node
$c_R$ – an SPPF node, the right child of the node about to be constructed
$\mathcal{R}$– list of pending descriptors
$\mathcal{U}$ – list of all constructed descriptors
$\mathcal{U}_i$ – all elements $(L.u, w)$ such that $(L, u, i, w) \in \mathcal{U}$
$\mathcal{P}$ – list of GSS node pop records

---


Centre for Software Language Engineering, Royal Holloway, University of London