# Shift-reduce Parser Worksheet

| State | *action* | | | | | | *goto* | | |
|---|---|---|---|---|---|---|---|---|---|
| | **id** | + | * | ( | ) | $ | $E$ | $T$ | $F$ |
| 0 | s5 | | | s4 | | | 1 | 2 | 3 |
| 1 | | s6 | | | | acc | | | |
| 2 | | r2 | s7 | | r2 | r2 | | | |
| 3 | | r4 | r4 | | r4 | r4 | | | |
| 4 | s5 | | | s4 | | | 8 | 2 | 3 |
| 5 | | r6 | r6 | | r6 | r6 | | | |
| 6 | s5 | | | s4 | | | | 9 | 3 |
| 7 | s5 | | | s4 | | | | | 10 |
| 8 | | s6 | | | s11 | | | | |
| 9 | | r1 | s7 | | r1 | r1 | | | |
| 10 | | r3 | r3 | | r3 | r3 | | | |
| 11 | | r5 | r5 | | r5 | r5 | | | |

**Fig. 4.31.** Parsing table for expression grammar.

(1) $E \rightarrow E + T$
(2) $E \rightarrow T$
(3) $T \rightarrow T * F$
(4) $T \rightarrow F$
(5) $F \rightarrow (E)$
(6) $F \rightarrow$ **id**

s*i* means shift and stack state *i*,
r*j* means reduce by production numbered *j*,
acc means accept,
blank means error.

```
set ip to point to the first symbol of w$;
repeat forever begin
        let s be the state on top of the stack and
              a the symbol pointed to by ip;
        if action[s, a] = shift s' then begin
              push a then s' on top of the stack;
              advance ip to the next input symbol
        end
        else if action[s, a] = reduce A → β then begin
              pop 2*|β| symbols off the stack;
              let s' be the state now on top of the stack;
              push A then goto[s', A] on top of the stack;
              output the production A → β
        end
        else if action[s, a] = accept then
              return
        else error()
end
```

**Fig. 4.30.** LR parsing program.

| | Stack | Input | Action |
|---|---|---|---|
| (1) | 0 | **id** * **id** + **id** $ | shift |
| (2) | 0 **id** 5 | * **id** + **id** $ | reduce by $F \rightarrow$ **id** |