Algorithme “shift-reduce” et “dangling else”

Une des raisons de préférer “avancer” à “réduire” lors d'un conflit avancer-réduire est liée au problème du “dangling else”. Par exemple, dans le code suivant (volontairement mal indenté !), à quel `if` le `else` doit-il se rapporter ?

```
if e then
    if e then
        i
  else i    // Dangling else!
```

Pour étudier cette question, considérons la grammaire (simplifiée !) suivante (on supposera que e (expression) et i (instr) sont des terminaux…) :

```
PROG          →  BLOC
BLOC          →  IF_STATEMENT
                 | i
IF_STATEMENT  →  if e then BLOC
                 | if e then BLOC else BLOC
```

Dérouler à la main un algorithme shift-reduce sur l'entrée ci-dessus, en supposant que, en cas de conflit shift-reduce

1. on choisit “shift”
2. on choisit “reduce”

BY-NC-SA distribué sous licence creative common | détails sur www.matthieuamiguet.ch