Alternative Abstract Syntax and Semantic Equations for Calculator

Program ::= ExprSeq
ExprSeq ::= Expr | Expr ExprExprSeq
Expr ::= Term | Expr Op Term | Expr Ans | Expr Ans +/-
Term ::= Num | MR | Clr | Term +/-
Op ::= + | – | *
Ans ::= M+ | =
Num ::= Fig. 9.1

Alternative (unambiguous) Fig. 9.5

meaning: Program Æ Integer
perform: ExprSeq Æ (State Æ State)
evaluate: Expr Æ (State Æ State)
compute: Op Æ (State Æ State)
calculate: Ans Æ { +/- } Æ (State Æ State)
value: Num Æ Integer

Alternative semantic function types

meaning [P] = d, where perform [P](0, nop, 0, 0) = (a, op, d, m)
perform [E] = evaluate [E]
perform [E S'] = perform [S] ° evaluate [E]
evaluate [N] (a, op, d, m) = (a, op, value [N], m)
evaluate [MR] (a, op, d, m) = (a, op, m, m)
evaluate [Clr] (a, op, d, m) = (0, nop, 0, 0)
evaluate [T +/-] = calculate [ +/- ] ° evaluate [T]
compute [+] (a, op, d, m) = (op(a, d), plus, op(a, d), m)
compute [–] (a, op, d, m) = (op(a, d), minus, op(a, d), m)
compute [*] (a, op, d, m) = (op(a, d), times, op(a, d), m)
calculate [=] (a, op, d, m) = (a, nop, op(a, d), m)
calculate [M+] (a, op, d, m) = (a, nop, op(a, d), plus(m, op(a, d)))
calculate [+/-] (a, op, d, m) = (a, op, minus(0, d), m)
value [N] = Fig. 9.1

Alternative Fig. 9.8