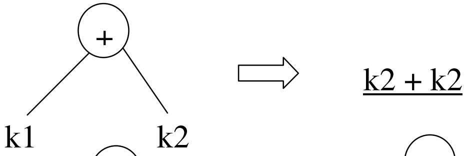
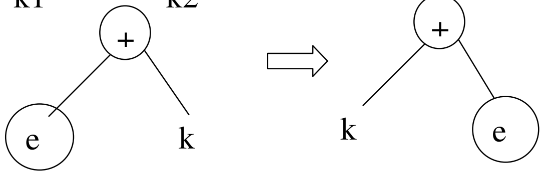
Simplifying Expressions by Tree-Rewriting

Rewrite-rules exploiting commutative and associative rule for + to compute addition of constants; strategy: move constants to the left/ move constants to the uppermost left operand:

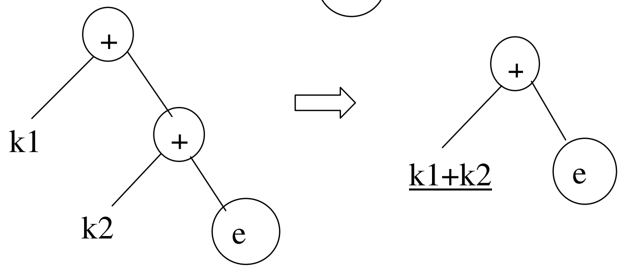
A:



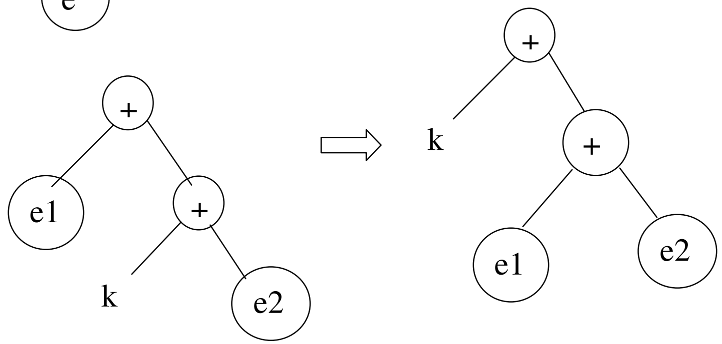
B:



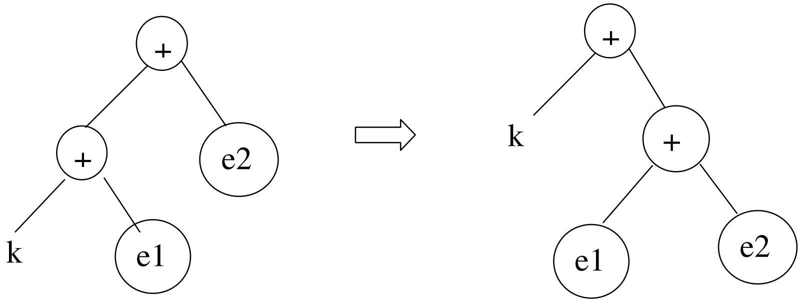
C:



D:

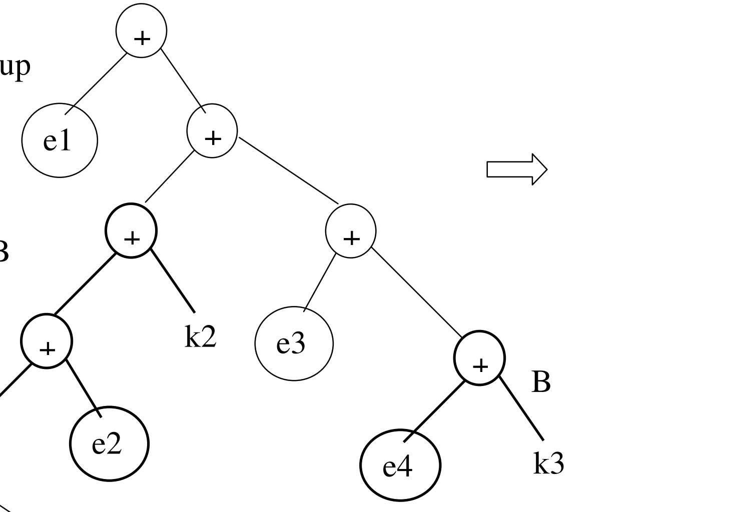


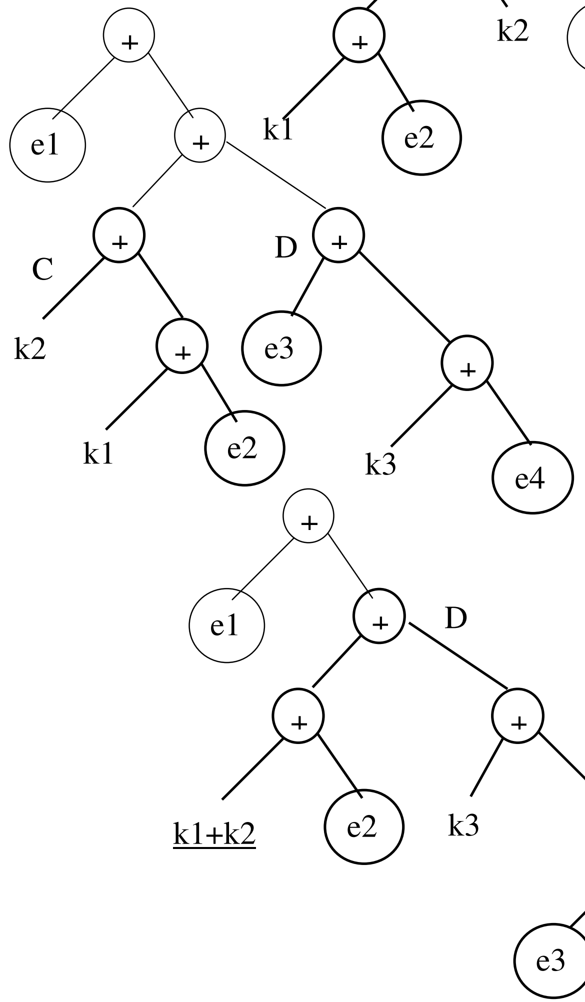
E:



Simplifying Expressions by Tree-Rewriting

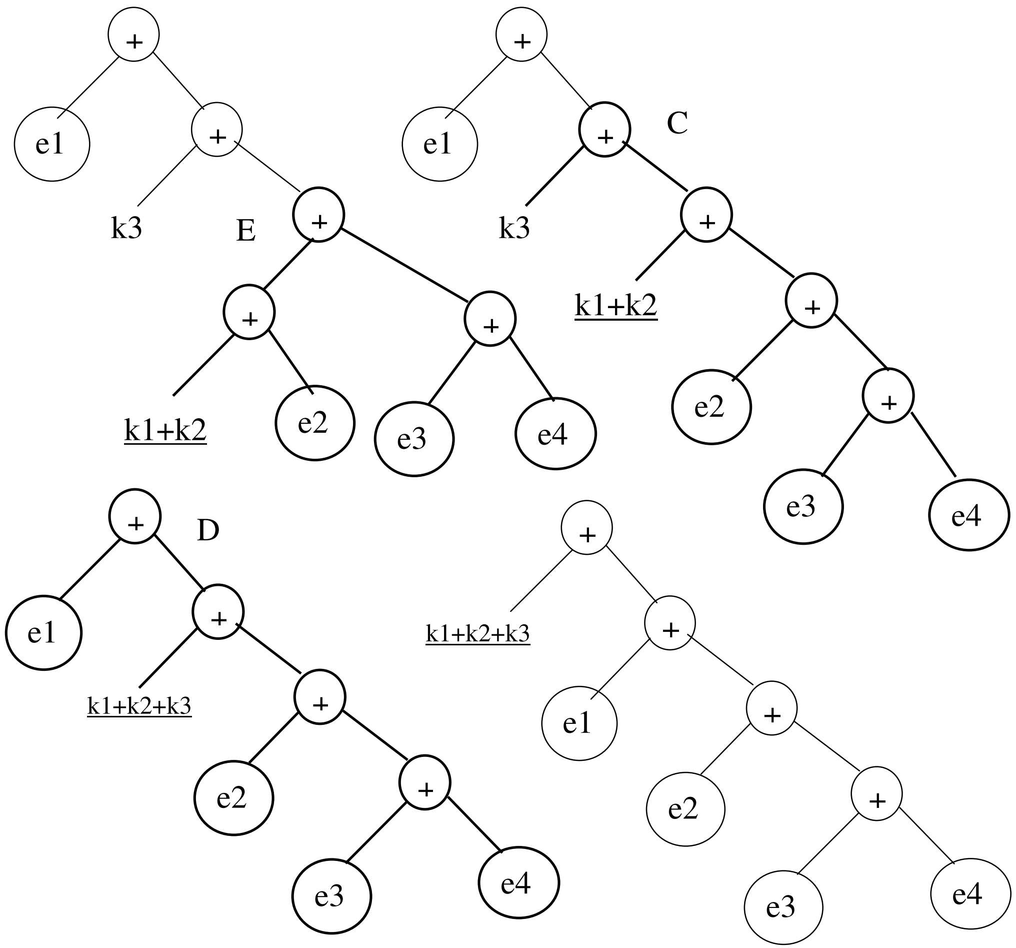
Example:
Apply the rules bottom-up





Simplifying Expressions by Tree-Rewriting

Example: Apply the rules bottom-up



Simplifying Expressions/Folding Bin-Expressions

This is an outline of a proposal for simplifying binary expressions. The idea is at each (+,-) level and (∗,/) level to move all constants to the left.

Splitting

Operator +, - : ⊕: e1 ⊕ e2

split each operand e1 and e2 into a triple: [constant, operator, optional expr]

operand	triple
c ⊕ e	[c, ⊕,e]
c	[c, + , nil]
e (all other expr's)	[0,+,e]

Splitting depends on the context operator

Operator *, / : ⊗: e1 ⊗ e2

split each operand e1 and e1 into a triple: [constant, operator, optional expr]

operand	triple
c ⊗ e	[c, ⊗,e]
c	[c, ∗ , nil]
e (all other expr's)	[1,∗,e]

Folding

Simplify/Fold BinExpr(oprt,e1,e2) , oprt∈ {+,-,∗,/}:

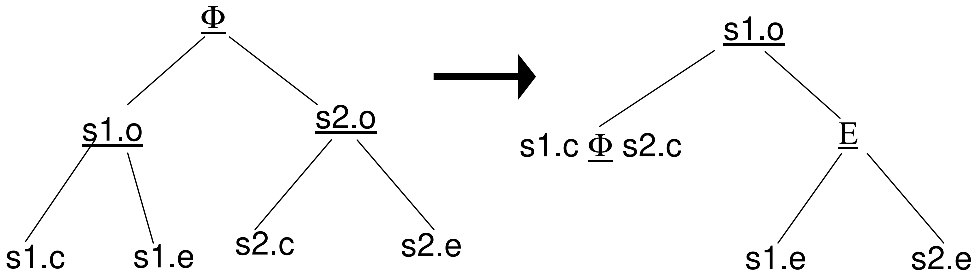
I) fold subexpressions: s1= e1.fold() , s2= e2.fold()

II) split folded subexpressions

  [s1.c, s1.o, s1.e]= split(oprt) s1, [s2.c, s2.o, s2.e]= split(oprt) s2

Then construct the simplified/folded expression as shown below for s1.e and s2.e being non-nil expressions.

<u>Fold BinExpr</u>



(s1.c $\underline{\text{s1.o}}$ s1.e) $\underline{\Phi}$ (s2.c $\underline{\text{s2.o}}$ s2.e) $\Rightarrow$

(s1.c $\underline{\Phi}$ s2.c) $\underline{\text{s1o}}$ (s1.e $\underline{\text{E}}$ s2.e)

where

$\underline{\text{s1.o}}$, $\underline{\text{s2.o}}$, $\underline{\Phi}$, $\underline{\text{E}} \in$ {Plus, Minus} $\vee$
$\underline{\text{s1.o}}$, $\underline{\text{s2.o}}$, $\underline{\Phi}$, $\underline{\text{E}} \in$ {Mult, Div}

$\underline{\text{s1.o}}$	$\underline{\Phi}$	$\underline{\text{s2.o}}$	$\underline{\text{E}}$
+	+	+	+
+	+	-	-
+	-	+	-
+	-	-	+
-	+	+	-
-	+	-	+
-	-	+	+
-	-	-	-

Application of the rules 0+e=e and 1$*$e=e is not shown.