

Rekursiver Abstieg: Von der EBNF-Regel zum Zerteileralgorithmus

Informatik 3, K. Hug, 2012-10-12

Umformungsregeln

EBNF-Ausdruck E	Zerteiler-Anweisung S (E)
(U1) "x"	IF symbol = "x" THEN next ELSE error END
(U2) Nonterminal	ReadNonterminal
(U3) (Expression)	S (Expression)
(U4) [Expression]	IF symbol IN first (Expression) THEN S (Expression) END
(U5) { Expression }	WHILE symbol IN first (Expression) DO S (Expression) END
(U6) Factor0 Factor1 .. Factorn	S (Factor0); S (Factor1); .. S (Factorn)
(U7) Term0 Term1 .. Termn	IF symbol IN first (Term0) THEN S (Term0) ELSIF symbol IN first (Term1) THEN S (Term1) .. ELSIF symbol IN first (Termn) THEN S (Termn) ELSE error END

EBNF-Regeln

(E1) Expression	= [AddOperator] Term { AddOperator Term }.
(E2) AddOperator	= "+" "-".
(E3) Term	= Factor { MulOperator Factor }.
(E4) MulOperator	= "*" "/".
(E5) Factor	= Ident "(" Expression ")".
(E6) Ident	= "a" "b" "c" "d".

Zerteilerprozeduren

ReadExpression
ReadTerm
ReadFactor
ReadIdent

Algorithmus (Implementation, Code) einer Zerteilerprozedur entsteht, indem man die rechte Seiten der EBNF-Regel schrittweise umformt.

Algorithmus zu ReadExpression

```
S ([ AddOperator ] Term { AddOperator Term })  
--> (U6)  
S ([ AddOperator ]); S (Term); S ({ AddOperator Term })
```

Drei Teile einzeln weiter berechnen Erster Teil

```
S ([ AddOperator ])  
--> (U4)  
  IF symbol IN first (AddOperator) THEN  
    S (AddOperator)  
  END  
--> (E2)  
  IF symbol IN first ("+" | "-") THEN  
    S ("+" | "-")  
  END  
--> first berechnen  
  IF symbol IN {"+", "-"} THEN  
    S ("+" | "-")  
  END  
--> (U7)  
  IF symbol IN {"+", "-"} THEN  
    IF symbol IN first ("+") THEN  
      S ("+")  
    ELSIF symbol IN first ( "-") THEN  
      S ( "-")  
    ELSE  
      error  
    END  
  END  
--> first berechnen, toten ELSE-Zweig streichen  
  IF symbol IN {"+", "-"} THEN  
    IF symbol = "+" THEN  
      S ("+")  
    ELSIF symbol = "-" THEN  
      S ( "-")  
    END  
  END  
--> unnötigen Test streichen
```

```

IF symbol IN {"+", "-"} THEN
    IF symbol = "+" THEN
        S ("+")
    ELSE
        S ("-")
    END
END

```

END

--> doppelten Test streichen

```

IF symbol = "+" THEN
    S ("+")
ELSIF symbol = "-" THEN
    S ("-")
END

```

END

--> (U1)

```

IF symbol = "+" THEN
    IF symbol = "+" THEN
        next
    ELSE
        error
    END
ELSIF symbol = "-" THEN
    IF symbol = "-" THEN
        next
    ELSE
        error
    END
END

```

```

END

```

END

--> tote ELSE-Zweige streichen

```

IF symbol = "+" THEN
    IF symbol = "+" THEN
        next
    END
ELSIF symbol = "-" THEN
    IF symbol = "-" THEN
        next
    END
END

```

```

END

```

END

--> doppelte Tests streichen

```

IF symbol = "+" THEN
    next
ELSIF symbol = "-" THEN
    next
END

```

END

--> gemeinsame Zweige zusammenführen

```

IF symbol IN {"+", "-"} THEN
    next
END

```

Zusammenfassung dieser Umformungskette:

S ([AddOperator])

--> (E2)

S (["+" | "-"])

-->

IF symbol IN {"+", "-"} THEN

 next

END

Zweiter Teil

S (Term)

--> (U2)

ReadTerm

Dritter Teil

S ({ AddOperator Term })

--> (E2)

S ({ ("+" | "-") Term })

--> analog zu Code von ReadTerm weiter unten

WHILE symbol IN {"+", "-"} DO

 next;

 ReadTerm

END

Drei Teile zusammenfügen

S ([AddOperator] Term { AddOperator Term })

-->

IF symbol IN {"+", "-"} THEN

 next

END;

ReadTerm;

WHILE symbol IN {"+", "-"} DO

 next;

 ReadTerm

END

Algorithmus zu ReadTerm

S (Factor { MulOperator Factor })

--> (U6)

S (Factor); S ({ MulOperator Factor })

--> (U2)

ReadFactor;

S ({ MulOperator Factor })

Zweiten Teil einzeln weiter berechnen

S ({ MulOperator Factor })

--> (U5)

WHILE symbol IN first (MulOperator Factor) DO

 S (MulOperator Factor)

END

--> first berechnen, (E4)

Nebenrechnung

first (MulOperator Factor) = first (("*" | "/") Factor) = first ("*" | "/") = {"*", "/"}

-->

WHILE symbol IN {"*", "/" } DO

 S (("*" | "/") Factor)

END

--> (U6)

WHILE symbol IN {"*", "/" } DO

 S (("*" | "/"));

 S (Factor)

END

--> vereinfachen ähnlich wie oben beim ersten Teil von ReadExpression + (U2)

WHILE symbol IN {"*", "/" } DO

 next;

 ReadFactor

END

Zusammenfassung dieser Umformungskette:

S ({ MulOperator Factor })

-->

WHILE symbol IN {"*", "/" } DO

 next;

 ReadFactor

END

Zwei Teile zusammenfügen

S (Factor { MulOperator Factor })

-->

ReadFactor;

WHILE symbol IN {"*", "/" } DO

 next;

 ReadFactor

END