a long time ago in a germany far,
far away...

PLAN KALKÜL

By Bernhard Behr

Overview

- History
- Features of Plankalkül
- Impact & Conclusion

History

- developed by Konrad Zuse during and after World War II
- first proposition of a high level programming language
- intended for use on a successor of the Z3 that was never realized
- completed after the war as post war chaos prevented Zuse from 'more practical' endeavours
- published in its entirety by Zuse in 1972

Features of Plankalkül

- Types
- Syntax

Types

- basic types
- structured types
- example

Basic Types

- Plankalkül knows one primitive type:
  - truth value (values 'L', '0')
  - denoted as S0
  - types of abitrary structure can be derived

Stuctured Types

- a type's strucutre is denoted by a structure index S
- types may be composed of other types
  - a type consisting of n values of type S0 may be denoted as n x S0 or S1.n
- a variable can be of variable structure
  - denoted by a structure variable σ
- certain fixed indices
  - for example S0 (truth value)

Structured Types (cont)

- every non-primitive type has components
  - S1.n denotes n components of type S0
- components are denoted by a component index K
- components may be addressed individually
  - any non-primitive type may be considered an array

Structured Types (cont)

- data type consists of
  - structure
  - type name (optional)
  - constraints(optional)

Example

- representation of decimal digits

$$A3 = \begin{pmatrix} S1 \cdot 4 \\ B3 \end{pmatrix}$$

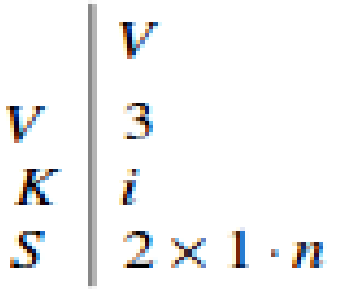
  - A3 denotes the type digit
  - A3 is of structure S1.4 (4 bits)
  - A3 is subject to set of constraints B3
    - digits 0-9 can be represented with a subset of S1.4
    - with a0-a3 as bits in A3, B3 = ¬a3 V ¬(a1 V a2)
  - optional type name T*n* is not used

Syntax

- Variables & Assignment
- Calculation Plans
- Conditionals
- Loops
- Example Program

Variables & Assignment

- variables are denoted by a variable index V
- type, name and relevant component of a variable are denoted by their corresponding indices in a line-based notation
  - two-dimensional syntax



- variables may be (re-)assigned values via an assignment operator “ $\Rightarrow$ “

Variables & Assignment (cont)

- type may be denoted by structure index or type
  - S or A for “Struktur” or “Art” respectively
- relevant component of a variable may be given by component index (see previous slide) or by other variables

	V	Z
V	0	1
K		
A	$1.n$	9

Calculation Plans

- Plankalkül is based on calculation plans
- plans can be interpreted as programs, functions or loops
- plans can call other plans
- indexed by a plan index P

Calculation Plans (cont)

- plans can have input and output parameters, intermediate results and constants
  - result parameter R*n* of plan P*m* with input value x is addressed as:

    P*m*.*n*(x)
- intermediate results and constants are invisble outside their respective plan

Calculation Plans (cont)

- plan consists of a header (“Randauszug”) and a body

$$
\begin{array}{lc|ccc}
P17 & & R(V) & \Rightarrow & (R, \quad R) \\
 & V & 0 & & 0 \quad 1 \\
 & S & \sigma & & \sigma \quad 0
\end{array}
$$

- header of plan P17 that takes one parameter V0 of structure σ and produces two result values, one of the type of V0 and one single bit

Calculation Plans (cont)

- it might be necessary or handy to abort a plan once a certain event occurs
  - Fin-symbol to stop the current plan
    - Fin stops the plan it occurs in
    - Fin² stops the next “higher” plan as well

Conditionals

- Plankalkül features conditional execution

- ```
  max( V0, V1 ){
     If( V0 >= V1 )then
        { R0 = V0 }
     else
        { R0 = V1 }
  }
  ```

	Maj(V,	V)	$\Rightarrow$	R
V	0	1		0
A	8	8		8

$V \geq V$	$\rightarrow$	$(V \Rightarrow R)$
0 1		0 0

$\overline{V \geq V}$	$\rightarrow$	$(V \Rightarrow R)$
0 1		1 0
  ```

Loops

- Plankalkül features repetetive plans
- marked by identifier *W*
- plan is only executed while a (set of) conditions holds
  - while-loop
- plan is executed n times
  - for-loop

$$W \left[\begin{array}{ccc} F & \vec{\cdot} & P \\ \overline{F} & \Rightarrow & \mathrm{Fin}^2 \end{array} \right]$$

$$\begin{array}{r|c|l|l} 1 \Rightarrow Z & W0 & (v) & \left[\begin{array}{ccc} Z \times v \Rightarrow Z \\ 0 \quad 0 \quad 0 \end{array} \right] \end{array} \Big| \; \begin{array}{l} Z \Rightarrow R \\ 0 \quad 0 \end{array}$$

Loops (cont)

- W0 to W5 mark special forms of for-loops

$$
\begin{array}{lll}
W0(n) & [P\,] & 0 \Rightarrow \varepsilon \mid \; W \; [\, \varepsilon < n \dot{\rightarrow} [P \quad \mid \varepsilon + 1 \Rightarrow \varepsilon \,]] \\
W1(n) & [P(i)] & 0 \Rightarrow i \mid \; W \; [\, i < n \dot{\rightarrow} [P(i) \mid i + 1 \Rightarrow i \,]] \\
W2(n) & [P(i)] & n-1 \Rightarrow i \mid \; W \; [\, i > 0 \dot{\rightarrow} [P(I) \mid i - 1 \Rightarrow i \,]] \\
W3(n,m) & [P(i)] & n \Rightarrow i \mid \; W \; [\, i < m \dot{\rightarrow} [P(i) \mid i + 1 \Rightarrow i \,]] \\
\underline{(n \leq m)} & & \\
W4(n,m) & [P(i)] & n \Rightarrow i \mid \; W \; [\, i > m \dot{\rightarrow} [P(i) \mid i - 1 \Rightarrow i \,]] \\
\underline{(n \geq m)} & & \\
W5(n,m) & [P(i)] & n \Rightarrow i \mid \; W \; \left[i \neq m \dot{\rightarrow} \left[\begin{array}{l} P(i) \\ m > n \dot{\rightarrow} (i + 1 \Rightarrow i) \\ m < n \dot{\rightarrow} (i - 1 \Rightarrow i) \end{array} \right] \right]
\end{array}
$$

Example Program

- a program for sorting a list by Zuse
- similar to insertionsort
- takes an unsorted list of m elements of structure σ

Example Program (cont)

P3.27

$W1(m-1)$

$$\begin{array}{l|ll} & \text{Ord } 1(V) & \Rightarrow R \\ & 0 & 0 \\ & m \times \sigma & m \times \sigma \end{array}$$

$$\begin{array}{l|ll} & V & \Rightarrow Z \\ V & 0 & 0 \\ S & m \times \sigma & m \times \sigma \end{array}$$

$$\begin{array}{l|l} & W1(m-1) \left[\begin{array}{l|ll} & Z \Rightarrow Z & i \Rightarrow \varepsilon \\ V & 0 \quad 1 & \\ K & \sigma \quad \sigma & 1.n \quad 1.n \\ S & W \left[\varepsilon \geq 0 \dot{\rightarrow} \left[\begin{array}{ll|l} Z < Z \dot{\rightarrow} & \left[\begin{array}{ll|l} Z \Rightarrow Z & & \varepsilon - 1 \Rightarrow \varepsilon \\ 0 \quad 0 & & \\ \varepsilon \quad \varepsilon + 1 & & \\ \sigma \quad \sigma & & \end{array} \right] & \\ \overline{Z < Z} \dot{\rightarrow} & \left[\begin{array}{ll|l} Z \Rightarrow Z & & \text{Fin}^3 \\ 0 \quad 0 & & \\ \varepsilon \quad \varepsilon + 1 & & \\ \sigma \quad \sigma & & \end{array} \right] & \end{array} \right] \right] \\ V & \varepsilon \quad = -1 \dot{\rightarrow} Z \Rightarrow Z & \\ K & 1 \quad 0 & \\ S & 0 & \\ & 1.n \quad 1.n \quad \sigma \quad \sigma & \end{array} \right] \end{array}$$

$$\begin{array}{l|ll} & Z & \Rightarrow R \\ V & 0 & 0 \\ S & m \times \sigma & m \times \sigma \end{array}$$

Example Program (cont)

```
• Sort( V0 ){
    Z0 = V0;
    for( int i = 0; i < ( m-1 ); i++ ){
      Z1 = Z0[i+1];
      e = i;
      while( e >= 0 ){
        if( Z1 < Z0[e] ){
          Z0[e+1] = Z0[e];
          e = e-1;
        }
```

Example Program (cont)

```
      else if( !(Z1 < Z0[e] )){
        Z0[e+1] = Z1;
        break;
      }
    }
    if( e == -1 )
      Z0[0] = Z1;
  }
  R0 = Z0;
  return R0;
}
```

Impact

- Plankalkül was never practically used
- went largely unnoticed at the time
- first implemented in 1975

Plankalkül
Ahead of its time?

- many features of modern programming languages
- Zuse planned to construct a logic-computer running Plankalkül
- higher programming languages only started appearing in the mid-1950ies
- very complex/impractical syntax
- several details open to interpretation or flawed
- building a working compiler might have been a bigger problem than designing the language

Bibliography

- Zuse K., “Der Plankalkül”, 1946
- Bauer F.L., Wössner H., “The 'Plankalkül' of Konrad Zuse: A Forerunner of Today's Programming Languages, Communications of the ACM, 1972
- Wikipedia