



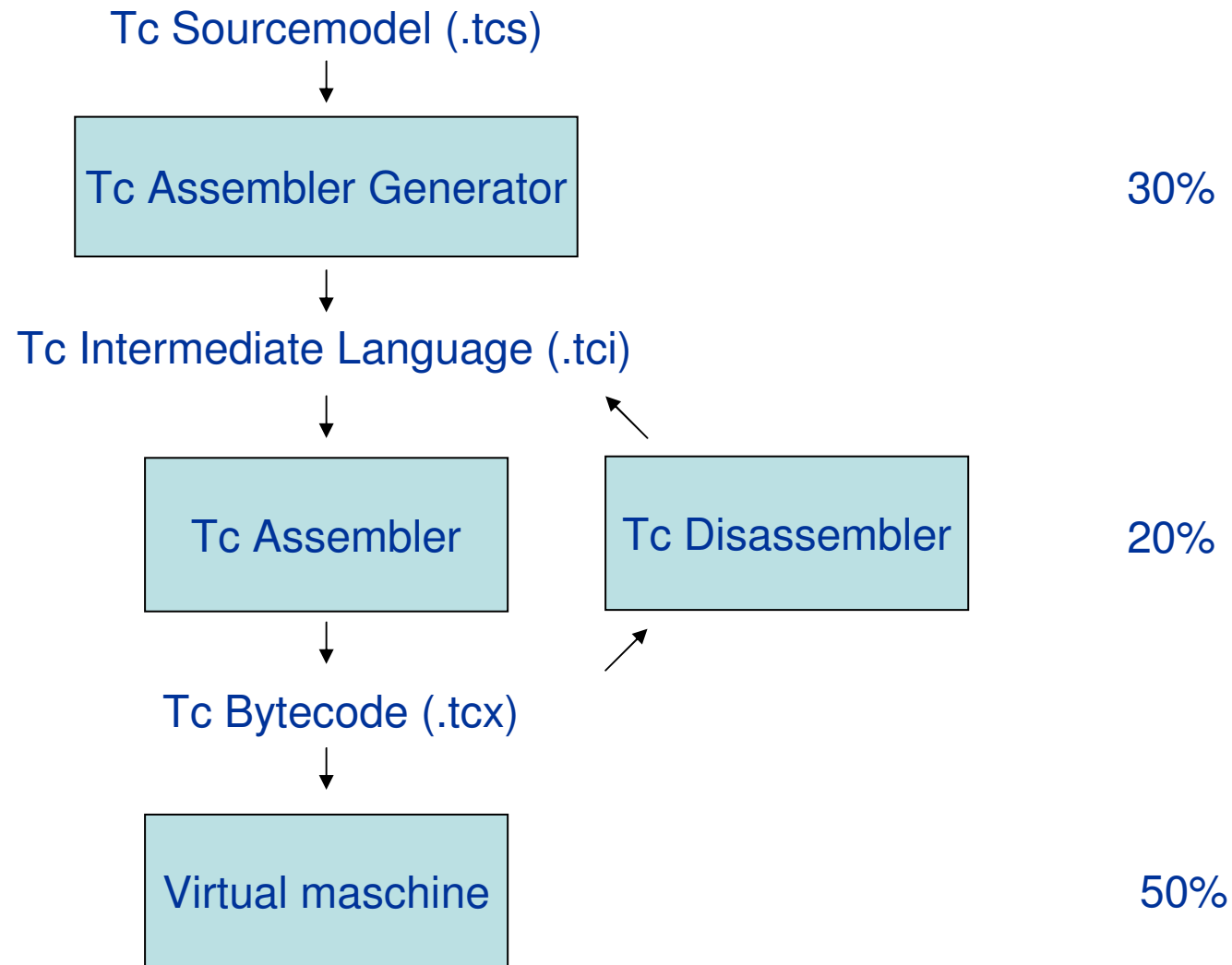
# TreeCalc Virtual Machine

Virtual machines 185.A49 UE

SS 2012 28.6.2012

Stefan Neubauer

- Starting point
  - TreeCalc language + Java generator
  - Test models and test data (JUnit)
- Goal
  - Virtual Machine with same functionality
  - Implementation in Java



- Variables
  - Numbers instead of names
  - Enhance symbol tables
- Nice and useful
  - Comments in output (variable names, ...)
  - Constants handled by Assembler
- Short-circuiting and, or
  - label-handling



```
A_LI_MainLayers.compute =  
  F_LI_Indexation_Perc > 0 ? F_LI_TariffDuration : 1
```

tcs

↓ Tc Assembler Generator

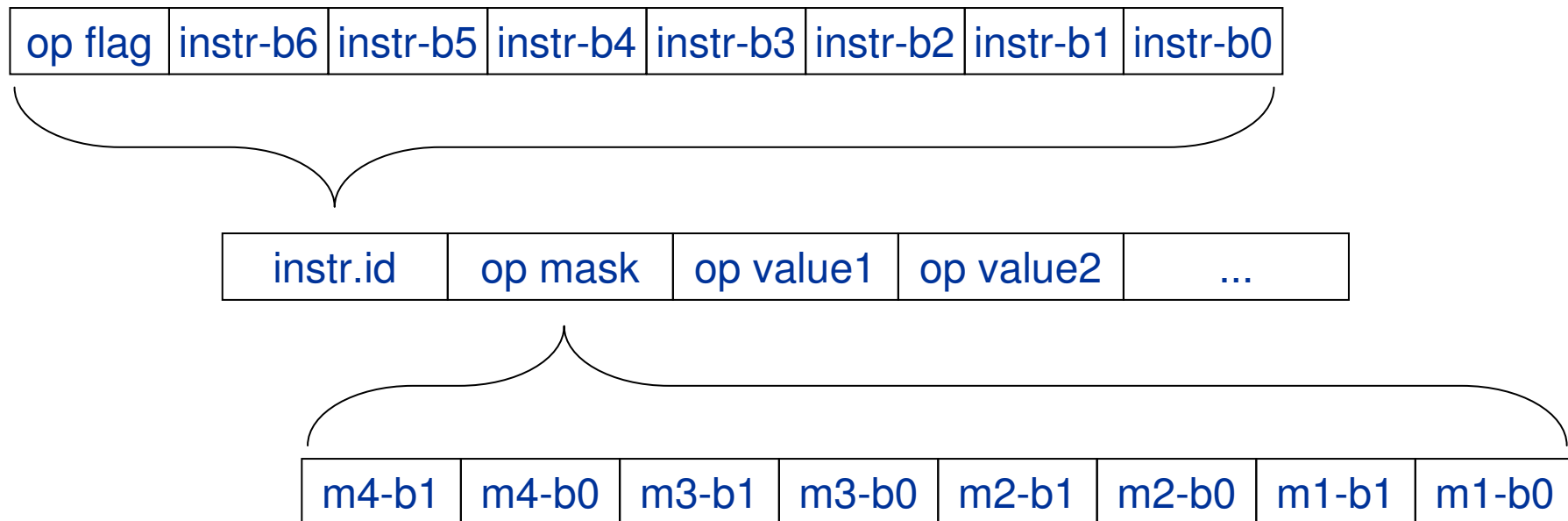
```
.formula formula=506 simple=false ; line 3182  
  //start of if statement, line 3182  
  : callfunc 65 0 ; F_LI_INDEXATION_PERC  
  : pushconst 0  
  : cmpbig  
  : iffalse L0  
  : callfunc 90 0 ; F_LI_TARIFFDURATION  
  : goto L1  
L0:  
  : pushconst 1  
L1:  
  //end of if statement  
  : return  
.formuladone
```

tci

- Configuration
  - Instructions array
    - id, name
    - stackeffect
    - list of <opname, optype>
- Main action
  - Parsing (ANTLR) + Error handling
  - Formulas
    - Manage constant pool
    - Resolve labels (backpatching)
    - Encode bytecode



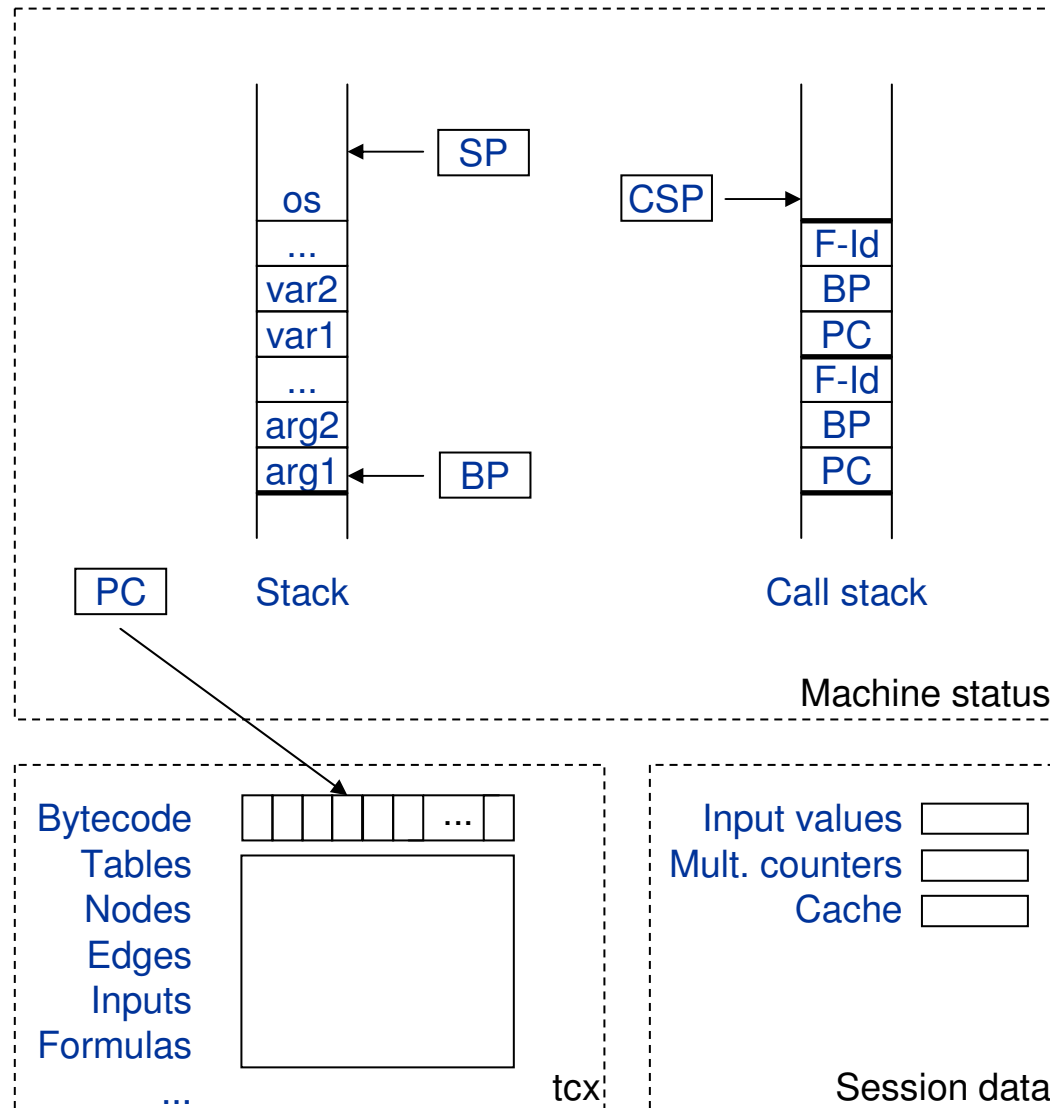
# Tc Bytecode - Formulas



b1	b0	operand value
0	0	no operand
0	1	1 byte
1	0	2 bytes, big-endian
1	1	4 bytes, big-endian



# Tc Virtual machine - Data





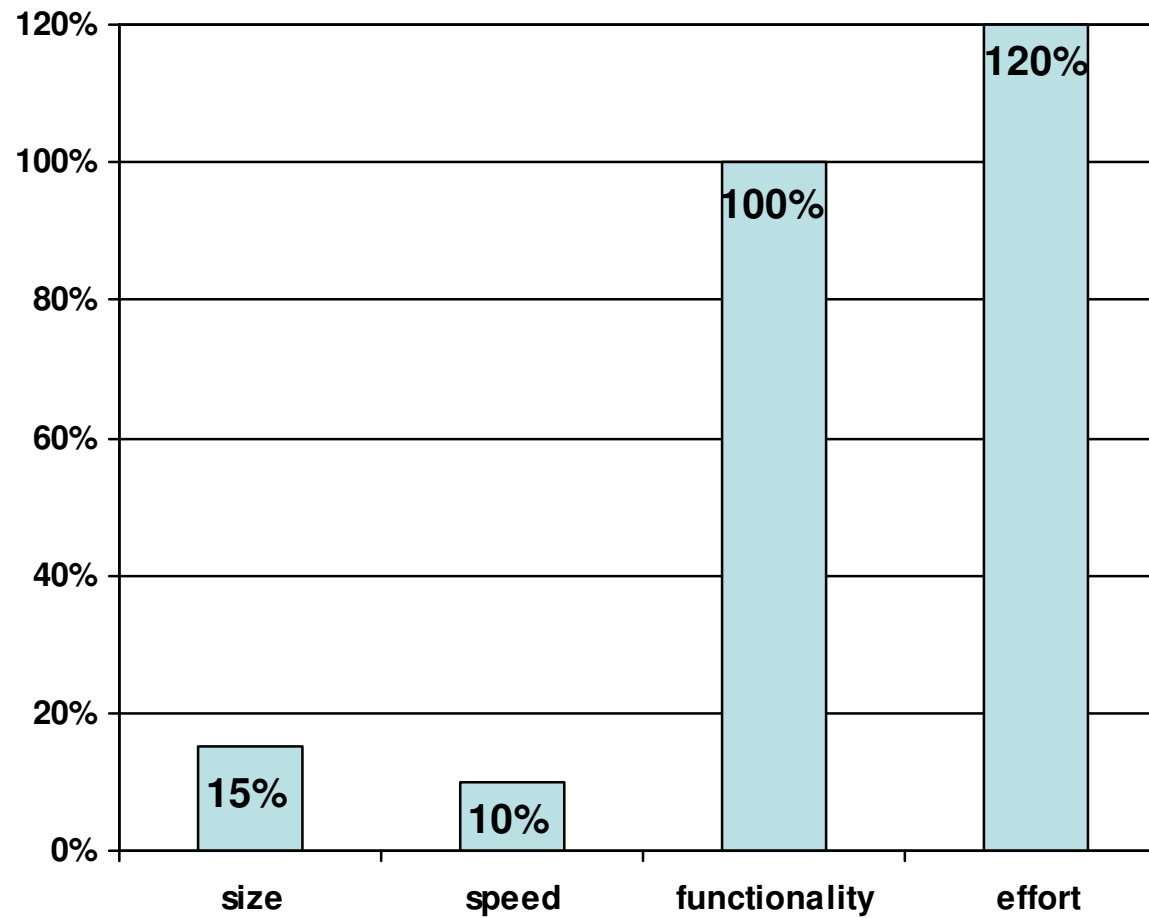


- Decode instruction → instrid, op1, op2, op3, op4
- switch(instrid) { ... }
- Operand stack
  - set freed elements to null → automatic GC

```
case INSTR_ADD: //a b -- a+b
    stack[sp-1] = V.getInstance(stack[sp-1].doubleValue() + stack[sp].doubleValue());
    stack[sp--] = null;
    break;
```

- Trace
  - PC, instruction, operands
  - status before+after instruction execution
- Base classes
  - Values: V, VString, VDouble, VList, ...
  - TreeCalc standard functions
  - Table access functions

- Macro programs for tree access
  - need special instructions
    - dynamic call of formula
    - get children / linked nodes
    - check if result is defined (own/subnodes)
    - ...
  - quite long (~2 x 200 instructions)
  - hard to debug
- Alternatives
  - recursive call to TcVM machine
  - tree actions / results in extra stacks



- Assembler
  - very useful layer
  - easy to implement
- Bytecode
  - compact vs. easy to handle
- Assembler generator
  - locatability of instruction names important
  - simpler than Java source code generator
- Virtual machine
  - for prototype: optimize later
  - trace output essential
  - tree handling: macro programming not best solution



<https://github.com/sneubauer/TreeCalc>

[sneubauer@gmx.at](mailto:sneubauer@gmx.at)