JumpVM

Das neue Tool für die Compilerbau I + II Übungen

Tim Wiederhake

Institut für Programmiermethodik und Compilerbau

04.11.2014

Zusammenfassung

Die „Java Unified Multi Paradigm Virtual Machine“ (JumpVM) ist das neue Übungswerkzeug für die Vorlesungen „Einführung in den Übersetzerbau“ und „Übersetzung fortgeschrittener Sprachkonzepte“ und löst die bisherigen Werkzeuge „pmach“, „wim“ und „mama“ ab.
In diesem Vortrag werden die Schwächen der bisher genutzten Werkzeuge gezeigt, die Erweiterungen der JumpVM gegenüber diesen Werkzeugen vorgestellt und die grundlegende Verwendung der JumpVM erläutert.

Über mich

- Informatik-Student (Master) der Universität Ulm
- Vorlesungen Compilerbau I + II gehört
- JumpVM entsteht als Projektmodul

Inhalt der Vorlesungen

Übersetzerbau für

- Imperative Sprachen (Pascal)
- Funktionale Sprachen (Haskell)
- Logische Sprachen (Prolog)
- Objektorientierte Sprachen (C++)

Inhalt der Übungen

- Vertiefen
- Anwenden
- Üben

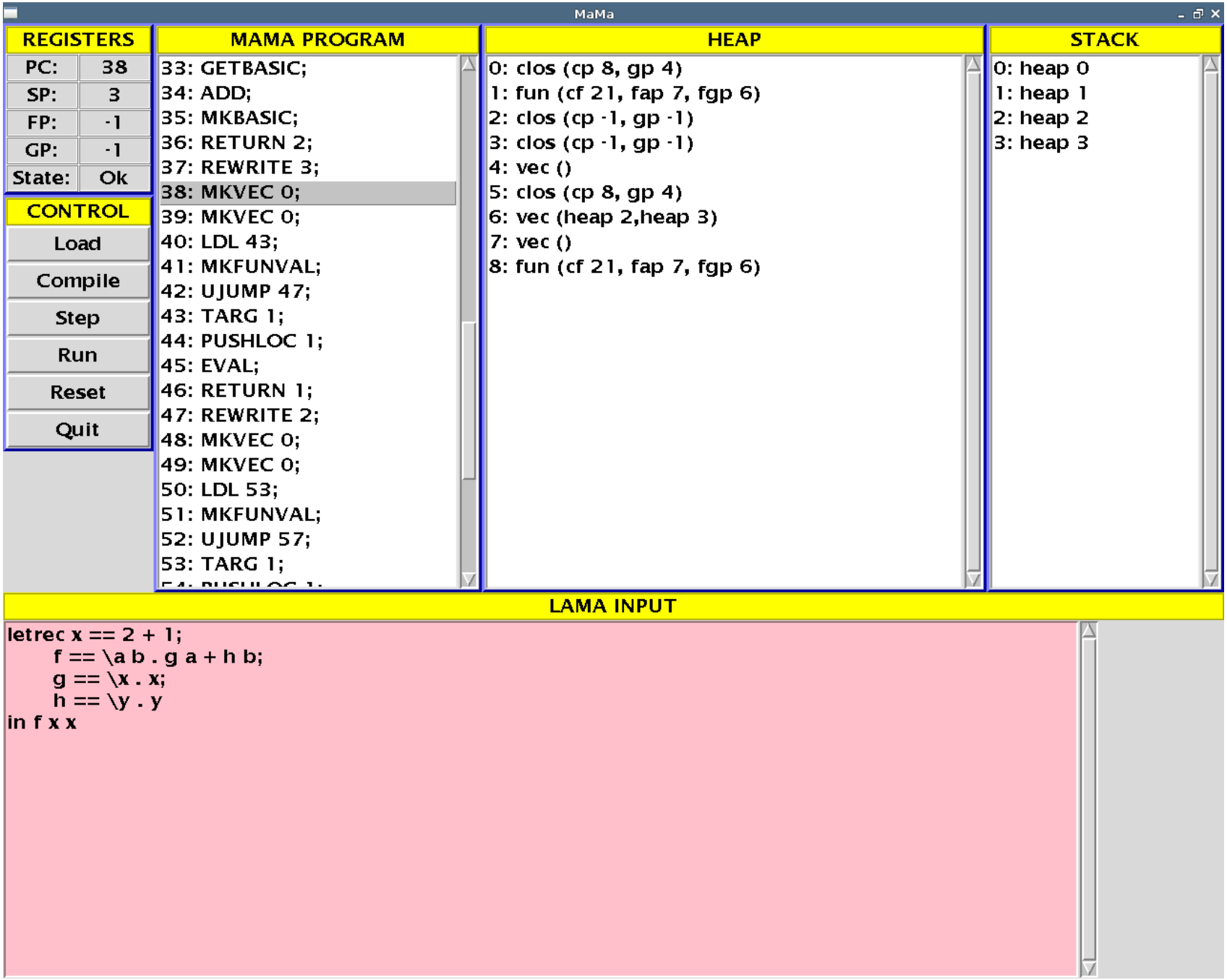
Inhalt der Übungen

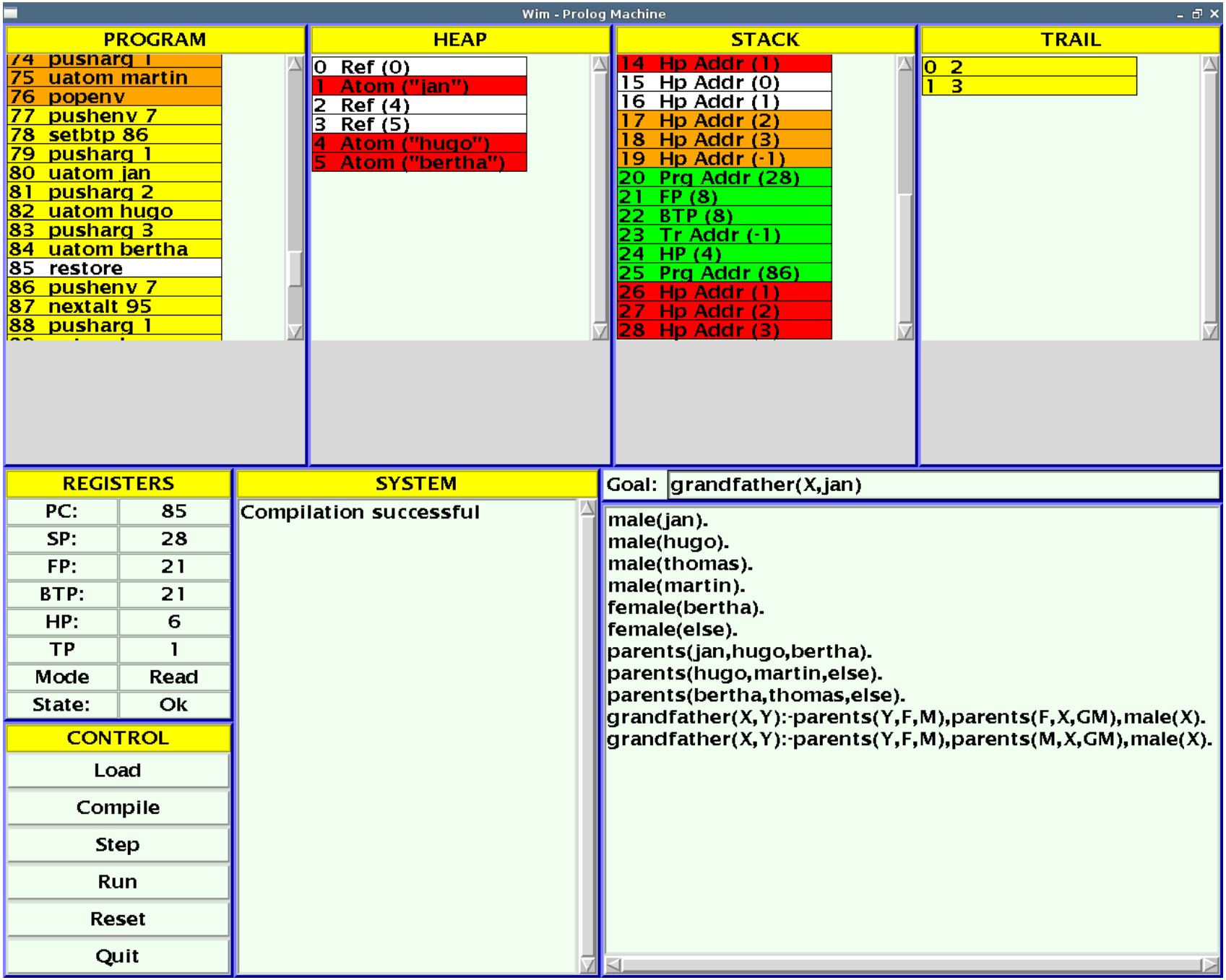
- Vertiefen
- Anwenden
- Üben

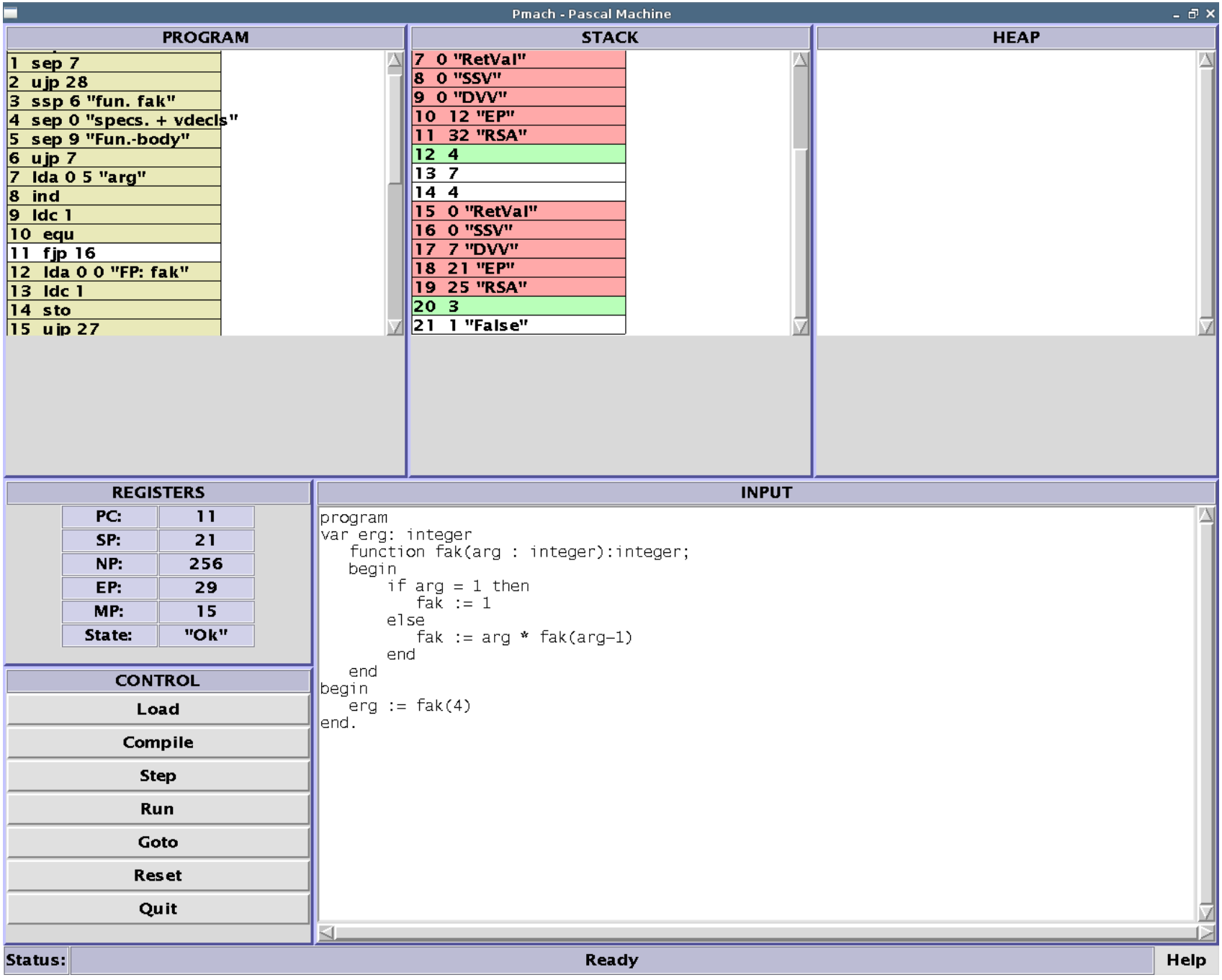
$\Rightarrow$ JumpVM

MaMa, Pmach und Wim

- Geschrieben in Gofer
- Grafische Oberfläche mit Tcl / Tk
- Entstanden 1995 – 1998
- Installiert in den Linux-Rechnerpools der SGI unter `/opt/Abteilungen/pm/Gofer_VM/bin`.


MaMa
REGISTERS
PC: 38
SP: 3
FP: -1
GP: -1
State: Ok
CONTROL
Load
Compile
Step
Run
Reset
Quit
MAMA PROGRAM
33: GETBASIC;
34: ADD;
35: MKBASIC;
36: RETURN 2;
37: REWRITE 3;
38: MKVEC 0;
39: MKVEC 0;
40: LDL 43;
41: MKFUNVAL;
42: UJUMP 47;
43: TARG 1;
44: PUSHLOC 1;
45: EVAL;
46: RETURN 1;
47: REWRITE 2;
48: MKVEC 0;
49: MKVEC 0;
50: LDL 53;
51: MKFUNVAL;
52: UJUMP 57;
53: TARG 1;
HEAP
0: clos (cp 8, gp 4)
1: fun (cf 21, fap 7, fgp 6)
2: clos (cp -1, gp -1)
3: clos (cp -1, gp -1)
4: vec ()
5: clos (cp 8, gp 4)
6: vec (heap 2,heap 3)
7: vec ()
8: fun (cf 21, fap 7, fgp 6)
STACK
0: heap 0
1: heap 1
2: heap 2
3: heap 3
LAMA INPUT
letrec x == 2 + 1;
f == \a b . g a + h b;
g == \x . x;
h == \y . y
in f x x


Wim - Prolog Machine
PROGRAM
74 pusharg 1
75 uatom martin
76 popenv
77 pushenv 7
78 setbtp 86
79 pusharg 1
80 uatom jan
81 pusharg 2
82 uatom hugo
83 pusharg 3
84 uatom bertha
85 restore
86 pushenv 7
87 nextalt 95
88 pusharg 1
HEAP
0 Ref (0)
1 Atom ("jan")
2 Ref (4)
3 Ref (5)
4 Atom ("hugo")
5 Atom ("bertha")
STACK
14 Hp Addr (1)
15 Hp Addr (0)
16 Hp Addr (1)
17 Hp Addr (2)
18 Hp Addr (3)
19 Hp Addr (-1)
20 Prg Addr (28)
21 FP (8)
22 BTP (8)
23 Tr Addr (-1)
24 HP (4)
25 Prg Addr (86)
26 Hp Addr (1)
27 Hp Addr (2)
28 Hp Addr (3)
TRAIL
0 2
1 3
REGISTERS
PC: 85
SP: 28
FP: 21
BTP: 21
HP: 6
TP 1
Mode Read
State: Ok
CONTROL
Load
Compile
Step
Run
Reset
Quit
SYSTEM
Compilation successful
Goal: grandfather(X,jan)
male(jan).
male(hugo).
male(thomas).
male(martin).
female(bertha).
female(else).
parents(jan,hugo,bertha).
parents(hugo,martin,else).
parents(bertha,thomas,else).
grandfather(X,Y):-parents(Y,F,M),parents(F,X,GM),male(X).
grandfather(X,Y):-parents(Y,F,M),parents(M,X,GM),male(X).


Pmach - Pascal Machine
PROGRAM
1 sep 7
2 ujp 28
3 ssp 6 "fun. fak"
4 sep 0 "specs. + vdecls"
5 sep 9 "Fun.-body"
6 ujp 7
7 lda 0 5 "arg"
8 ind
9 ldc 1
10 equ
11 fjp 16
12 lda 0 0 "FP: fak"
13 ldc 1
14 sto
15 ujp 27
STACK
7 0 "RetVal"
8 0 "SSV"
9 0 "DVV"
10 12 "EP"
11 32 "RSA"
12 4
13 7
14 4
15 0 "RetVal"
16 0 "SSV"
17 7 "DVV"
18 21 "EP"
19 25 "RSA"
20 3
21 1 "False"
HEAP
REGISTERS
PC: 11
SP: 21
NP: 256
EP: 29
MP: 15
State: "Ok"
INPUT
program
var erg: integer
function fak(arg : integer):integer;
begin
if arg = 1 then
fak := 1
else
fak := arg * fak(arg-1)
end
end
begin
erg := fak(4)
end.
CONTROL
Load
Compile
Step
Run
Goto
Reset
Quit
Status:
Ready
Help

Wart- und Erweiterbarkeit

Gofer (Haskell-Dialekt, †1994)

```
dostep :: Status -> Edit -> GUI ()
dostep s ou = do
  (m@(rg@(pc',_,_,_,_,_,_),he,st,tr,err),p) <- readGVar s
  if length p == 0
    then do writeGVar s ((rg,he,st,tr,Stopped),p)
            sysout ou "No Program"
    else
    if (err /= Ok) || (pc' >= length p)
      then do writeGVar s ((rg,he,st,tr,Stopped),p)
              tk_showError "You must reset the machine first
                  "
      else do if ((fst.fst) (p !! pc')) == "halt"
                then sysout ou (varstr st he)
                else done
              let m'@(_,_,_,_,err') = step (snd (p !! pc'))
                                           (incpc m)
(...)
```

Installierbarkeit

- Nur in den Linux-Pools der SGI
- Keine Windows- oder Mac-Versionen
- Abhängigkeiten (Tk 4.2 $\rightarrow$ 8.6 in Debian Jessie)
- Eigene Gofer/Tk-Bindings
- Unbekannte Lizenzierung
- WiSe 2014: Umstellung des Linux-Pools auf Fedora: Nicht mehr ausführbar

Benutzerfreundlichkeit

- Uneinheitliche GUI (Register, Speicher, Buttons)
- Stürzt ohne Fehlermeldung bei falscher Syntax ab
- Bugs in „wim“ ($\rightarrow$ „`equals(a, b)`“)
- Bugs in „PMach“ ($\rightarrow$ „`true = false = 0`“)
- Gofer-Interpreter stürzt bei größeren Programmen ab
- Ausführung verlangsamt sich mit jeder Instruktion

Ziele

Ziele

- Plattformunabhängigkeit
- Populäre Programmiersprache

Ziele

- Plattformunabhängigkeit
- Populäre Programmiersprache $\Rightarrow$ Java

Ziele

- Plattformunabhängigkeit
- Populäre Programmiersprache $\Rightarrow$ Java
- Vereinheitlichung der Oberfläche
- Generalisierung von Code

Ziele

- Plattformunabhängigkeit
- Populäre Programmiersprache ⇒ Java
- Vereinheitlichung der Oberfläche
- Generalisierung von Code ⇒ JumpVM-„Framework“

Ziele

- Plattformunabhängigkeit
- Populäre Programmiersprache ⇒ Java
- Vereinheitlichung der Oberfläche
- Generalisierung von Code ⇒ JumpVM-„Framework“
- Einhalten von Standards

Ziele

- Plattformunabhängigkeit
- Populäre Programmiersprache ⇒ Java
- Vereinheitlichung der Oberfläche
- Generalisierung von Code ⇒ JumpVM-„Framework“
- Einhalten von Standards ⇒ Checkstyle

Ziele

- Plattformunabhängigkeit
- Populäre Programmiersprache ⇒ Java
- Vereinheitlichung der Oberfläche
- Generalisierung von Code ⇒ JumpVM-„Framework“
- Einhalten von Standards ⇒ Checkstyle
- Ausführliche Dokumentation

Ziele

- Plattformunabhängigkeit
- Populäre Programmiersprache ⇒ Java
- Vereinheitlichung der Oberfläche
- Generalisierung von Code ⇒ JumpVM-„Framework“
- Einhalten von Standards ⇒ Checkstyle
- Ausführliche Dokumentation ⇒ JavaDoc + LaTeX

Ziele

- Plattformunabhängigkeit
- Populäre Programmiersprache ⇒ Java
- Vereinheitlichung der Oberfläche
- Generalisierung von Code ⇒ JumpVM-„Framework“
- Einhalten von Standards ⇒ Checkstyle
- Ausführliche Dokumentation ⇒ JavaDoc + LaTeX
- Tests!

Ziele

- Plattformunabhängigkeit
- Populäre Programmiersprache ⇒ Java
- Vereinheitlichung der Oberfläche
- Generalisierung von Code ⇒ JumpVM-„Framework“
- Einhalten von Standards ⇒ Checkstyle
- Ausführliche Dokumentation ⇒ JavaDoc + LaTeX
- Tests! ⇒ JUnit

Ziele

- Plattformunabhängigkeit
- Populäre Programmiersprache ⇒ Java
- Vereinheitlichung der Oberfläche
- Generalisierung von Code ⇒ JumpVM-„Framework“
- Einhalten von Standards ⇒ Checkstyle
- Ausführliche Dokumentation ⇒ JavaDoc + LaTeX
- Tests! ⇒ JUnit
- Eindeutige Lizenzierung

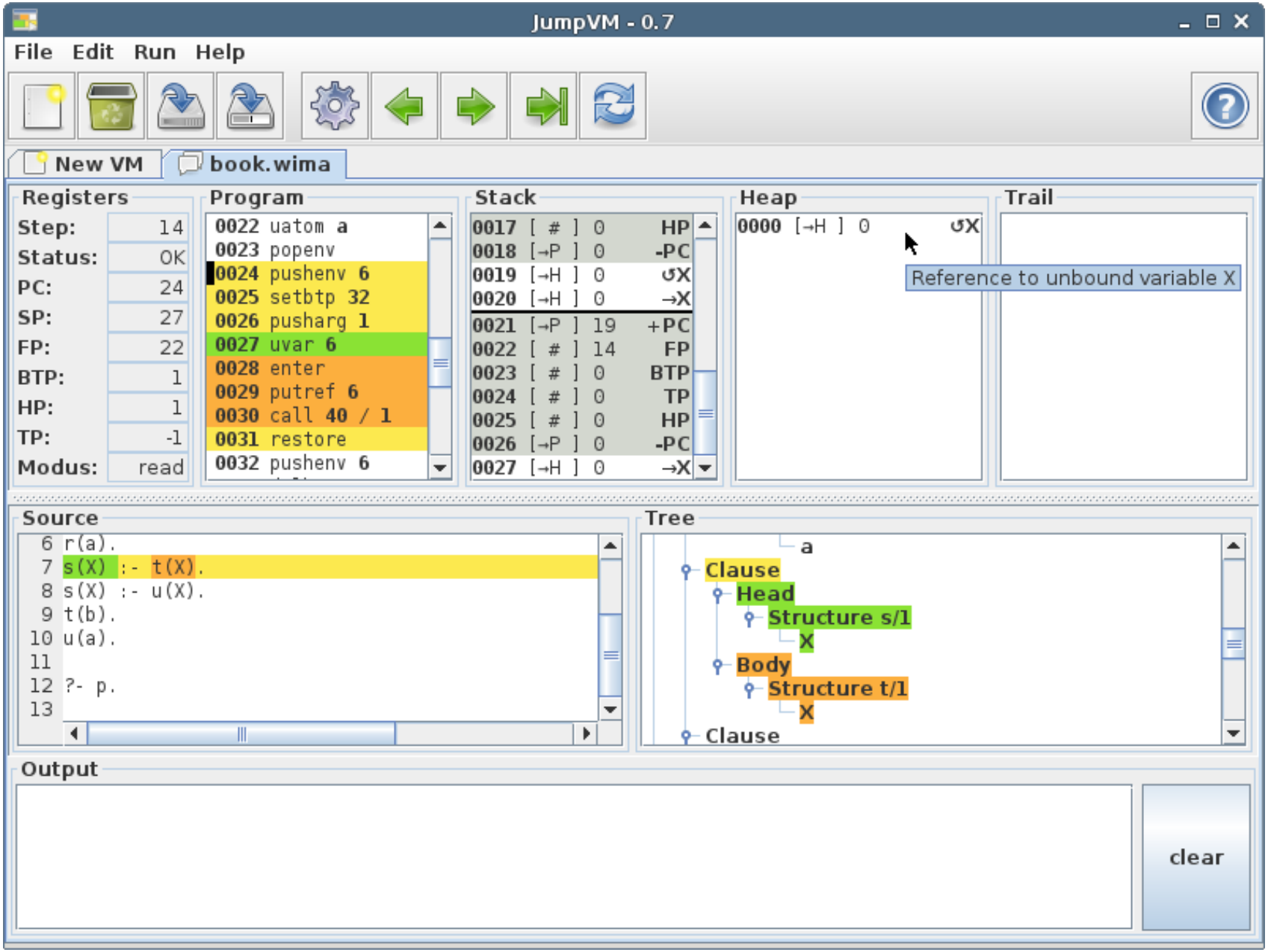
Ziele

- Plattformunabhängigkeit
- Populäre Programmiersprache ⇒ Java
- Vereinheitlichung der Oberfläche
- Generalisierung von Code ⇒ JumpVM-„Framework“
- Einhalten von Standards ⇒ Checkstyle
- Ausführliche Dokumentation ⇒ JavaDoc + LaTeX
- Tests! ⇒ JUnit
- Eindeutige Lizenzierung ⇒ Gnu GPL 3+

Ziele

- Plattformunabhängigkeit
- Populäre Programmiersprache ⇒ Java
- Vereinheitlichung der Oberfläche
- Generalisierung von Code ⇒ JumpVM-„Framework“
- Einhalten von Standards ⇒ Checkstyle
- Ausführliche Dokumentation ⇒ JavaDoc + LaTeX
- Tests! ⇒ JUnit
- Eindeutige Lizenzierung ⇒ Gnu GPL 3+

⇒ „Paretodominante“ Lösung


JumpVM - 0.7
File Edit Run Help
New VM
book.wima
Registers
Step: 14
Status: OK
PC: 24
SP: 27
FP: 22
BTP: 1
HP: 1
TP: -1
Modus: read
Program
0022 uatom a
0023 popenv
0024 pushenv 6
0025 setbtp 32
0026 pusharg 1
0027 uvar 6
0028 enter
0029 putref 6
0030 call 40 / 1
0031 restore
0032 pushenv 6
Stack
0017 [#] 0 HP
0018 [→P] 0 -PC
0019 [→H] 0 ↺X
0020 [→H] 0 →X
0021 [→P] 19 +PC
0022 [#] 14 FP
0023 [#] 0 BTP
0024 [#] 0 TP
0025 [#] 0 HP
0026 [→P] 0 -PC
0027 [→H] 0 →X
Heap
0000 [→H] 0 ↺X
Reference to unbound variable X
Trail
Source
6 r(a).
7 s(X) :- t(X).
8 s(X) :- u(X).
9 t(b).
10 u(a).
11
12 ?- p.
13
Tree
a
Clause
Head
Structure s/1
X
Body
Structure t/1
X
Clause
Output
clear

Fähigkeiten

Fähigkeiten

- Vollständiger Funktionsumfang der „PMach“, „wim“ und „mama“

Fähigkeiten

- Vollständiger Funktionsumfang der „PMach“, „wim“ und „mama“
- Brainfuck-Compiler als Beispiel für die Verwendung des JumpVM-Frameworks

Fähigkeiten

- Vollständiger Funktionsumfang der „PMach“, „wim“ und „mama“
- Brainfuck-Compiler als Beispiel für die Verwendung des JumpVM-Frameworks
- Fehlermeldungen, Hilfetexte

Fähigkeiten

- Vollständiger Funktionsumfang der „PMach“, „wim“ und „mama“
- Brainfuck-Compiler als Beispiel für die Verwendung des JumpVM-Frameworks
- Fehlermeldungen, Hilfetexte
- Erweiterte Beispiele

Fähigkeiten

- Vollständiger Funktionsumfang der „PMach“, „wim“ und „mama“
- Brainfuck-Compiler als Beispiel für die Verwendung des JumpVM-Frameworks
- Fehlermeldungen, Hilfetexte
- Erweiterte Beispiele
- Anzeige und Export des Syntaxbaumes und der Maschinenbefehle

Fähigkeiten (forts.)

- Ändern von Registerinhalten, Ändern des Speicherinhaltes

Fähigkeiten (forts.)

- Ändern von Registerinhalten, Ändern des Speicherinhaltes
- Durchgehende Annotation und Typisierung von Werten

Fähigkeiten (forts.)

- Ändern von Registerinhalten, Ändern des Speicherinhaltes
- Durchgehende Annotation und Typisierung von Werten
- Schrittweises Rückgängigmachen der Ausführung

Fähigkeiten (forts.)

- Ändern von Registerinhalten, Ändern des Speicherinhaltes
- Durchgehende Annotation und Typisierung von Werten
- Schrittweises Rückgängigmachen der Ausführung
- Abbildung Sourcecode $\leftrightarrow$ Syntaxbaum $\leftrightarrow$ Maschinenbefehle

Fähigkeiten (forts.)

- Ändern von Registerinhalten, Ändern des Speicherinhaltes
- Durchgehende Annotation und Typisierung von Werten
- Schrittweises Rückgängigmachen der Ausführung
- Abbildung Sourcecode $\leftrightarrow$ Syntaxbaum $\leftrightarrow$ Maschinenbefehle
- Befehle für Eingabe und Ausgabe

Demonstration

Bezugsquelle

`https://github.com/twied/jumpvm/releases` 🖱

... und bald™ im Linux-Pool vorinstalliert.

Danke für Ihre Zeit!