

Syntax-Diagrams for Lua

Knut Lickert

October 1, 2006

<http://lua.lickert.net>

Contents

1	The Complete (Original) Syntax of Lua	2
2	Lua-Syntax with Rail-diagramms	3
2.1	Programm structure	3
2.2	Statements	4
2.3	Branches (if)	5
2.4	Loops	6
2.4.1	For-Loop	6
2.4.2	While-Loop	7
2.4.3	Repeat-Until-Loop	7
2.5	Functions	7
2.6	Value Assignment	8
2.7	Expressions	9
2.8	Tables	11
2.9	Operators	13
3	Additional Diagramms	14

List of Figures

1	Rail-Diagramm chunk	3
2	Rail-Diagramm block	4
3	Rail-Diagramm stat	4
4	Rail-Diagramm ifstatement	5
5	Rail-Diagramm ifstatement1	5
6	Rail-Diagramm ifstatement2	6
7	Rail-Diagramm forstat	6
8	Rail-Diagramm forstat1	6
9	Rail-Diagramm forstat2	7

10	Rail-Diagramm while	7
11	Rail-Diagramm repeat	7
12	Rail-Diagramm funcname	7
13	Rail-Diagramm varlist1	8
14	Rail-Diagramm var	8
15	Rail-Diagramm namelist	8
16	Rail-Diagramm init	8
17	Rail-Diagramm explist1	9
18	Rail-Diagramm exp	9
19	Rail-Diagramm prefixexp	10
20	Rail-Diagramm functioncall	10
21	Rail-Diagramm args	10
22	Rail-Diagramm function	10
23	Rail-Diagramm funcbody	11
24	Rail-Diagramm parlist1	11
25	Rail-Diagramm parlist1variant	11
26	Rail-Diagramm tableconstructor	11
27	Rail-Diagramm fieldlist	12
28	Rail-Diagramm field	12
29	Rail-Diagramm fieldsep	12
30	Rail-Diagramm binop	13
31	Rail-Diagramm unop	13
32	Rail-Diagramm string	14
33	Rail-Diagramm print	14

1 The Complete (Original) Syntax of Lua

Standard EBNF:

{**x**} Zero or multiple occurrence of *x*

[] Optional

```
chunk -> { stat [ ';' ] }
block -> chunk
stat -> varlist1 '=' explist1
      | functioncall
      | do block end
      | while exp do block end
      | repeat block until exp
      | if exp then block { elseif exp then block } [ else block ] end
      | return [ explist1 ]
      | break
      | for Name '=' exp ',' exp [ ',' exp ] do block end
```

```

    | for Name { ',' Name } in explist1 do block end
    | function funcname funcbody
    | local function Name funcbody
    | local namelist [ init ]
funcname -> Name { '.' Name } [ ':' Name ]
varlist1 -> var { ',' var }
var -> Name | prefixexp '[' exp ']' | prefixexp '.' Name
namelist -> Name { ',' Name }
init -> '=' explist1
explist1 -> { exp ',' } exp
exp -> nil | false | true | Number | Literal
    | function | prefixexp | tableconstructor | exp binop exp | unop exp
prefixexp -> var | functioncall | '(' exp ')'
functioncall -> prefixexp args | prefixexp ':' Name args
args -> '(' [ explist1 ] ')' | tableconstructor | Literal
function -> function funcbody
funcbody -> '(' [ parlist1 ] ')' block end
parlist1 -> Name { ',' Name } [ ',', '...' ] | '...'
tableconstructor -> '{' [ fieldlist ] '}'
fieldlist -> field { fieldsep field } [ fieldsep ]
field -> '[' exp ']' '=' exp | name '=' exp | exp
fieldsep -> ',' | ';'
binop -> '+' | '-' | '*' | '/' | '^' | '..'
    | '<' | '<=' | '>' | '>=' | '==' | '~='
    | and | or
unop -> '-' | not

```

2 Lua-Syntax with Rail-diagramms

2.1 Programm structure

chunk

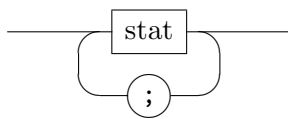


Figure 1: Rail-Diagramm chunk

```
chunk -> { stat [ ';' ] }
```

block



Figure 2: Rail-Diagramm block

block -> chunk

2.2 Statements

stat

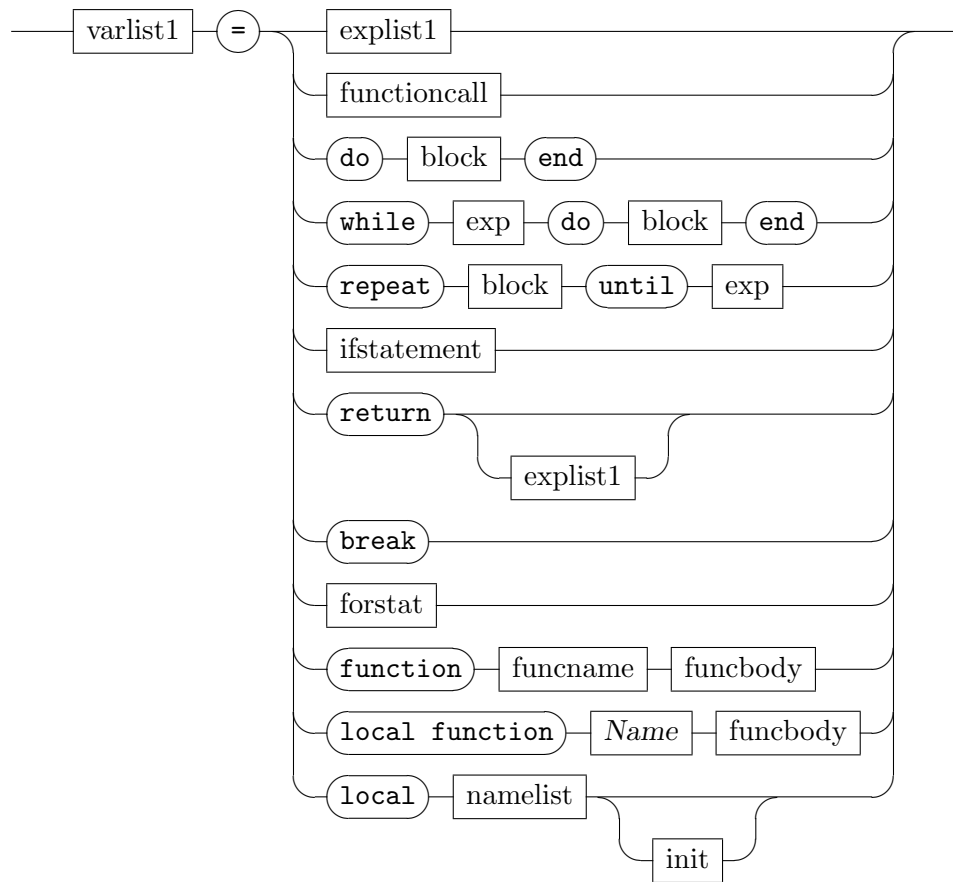


Figure 3: Rail-Diagramm stat

```
stat -> varlist1 '=' explist1
      | functioncall
      | do block end
      | while exp do block end
      | repeat block until exp
```

```

| if exp then block { elseif exp then block } [ else block ] end
| return [ explist1 ]
| break
| for Name '=' exp ',' exp [ ',' exp ] do block end
| for Name { ',' Name } in explist1 do block end
| function funcname funcbody
| local function Name funcbody
| local namelist [ init ]

```

2.3 Branches (if)

In original definition part of *stat*
ifstatement

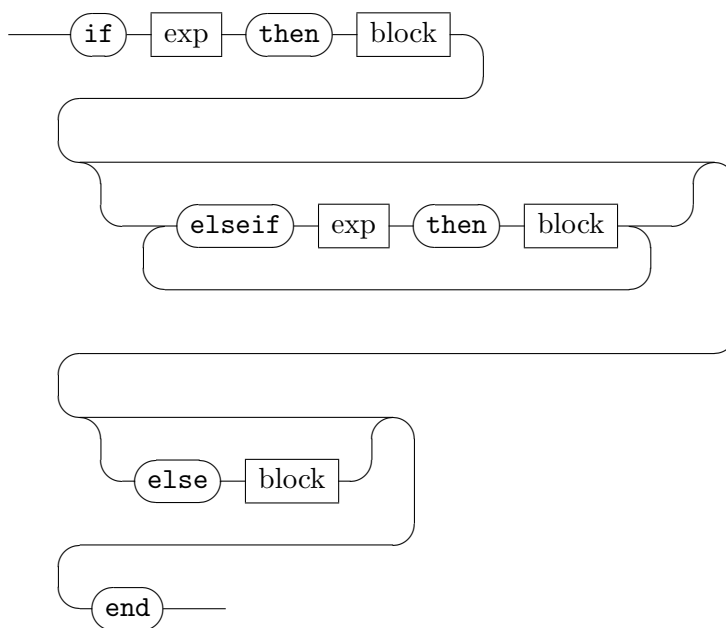


Figure 4: Rail-Diagramm ifstatement

The complete if-statement contains else- and elseif-parts.

ifstatement1

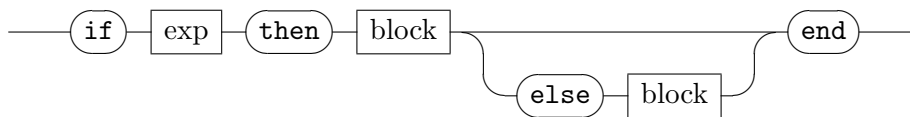


Figure 5: Rail-Diagramm ifstatement1

If without the elseif-statements.

ifstatement2

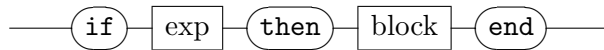


Figure 6: Rail-Diagramm ifstatement2

If without else-statements.

2.4 Loops

In the original EBNF-Definition, this definitions are part of *stat*

2.4.1 For-Loop

forstat

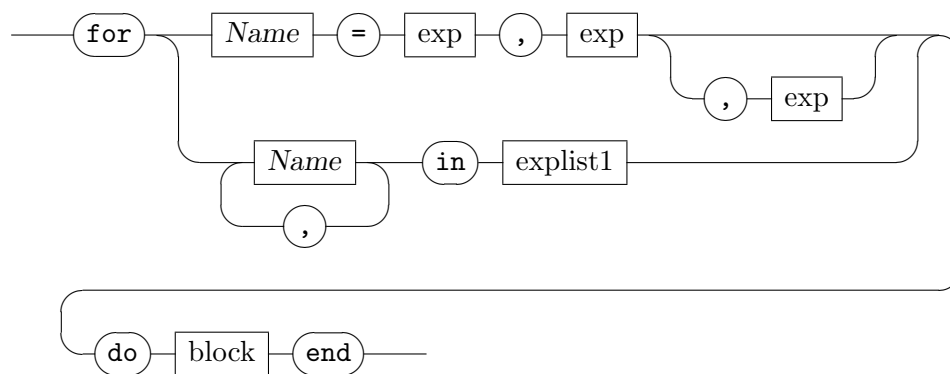


Figure 7: Rail-Diagramm forstat

```
stat ->| for Name '=' exp ',' exp [ ',' exp ] do block end
      | for Name { ',' Name } in explist1 do block end
```

forstat includes the two alternatives. A *normal* one and a version for tables.
forstat1

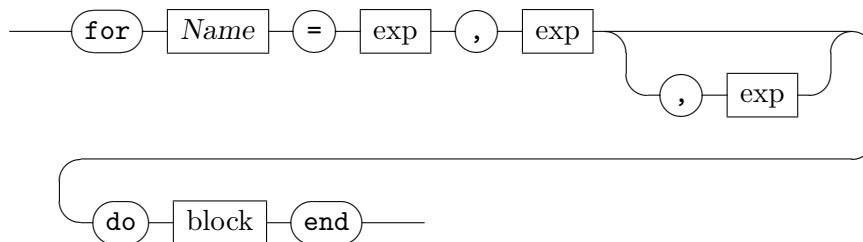


Figure 8: Rail-Diagramm forstat1

forstat2

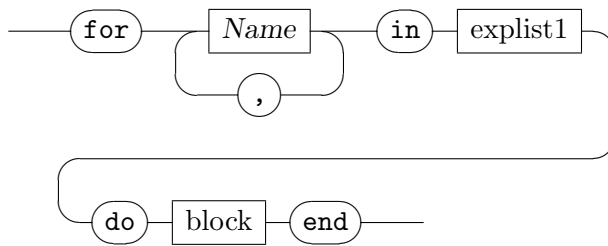


Figure 9: Rail-Diagramm forstat2

2.4.2 While-Loop

while

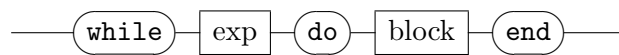


Figure 10: Rail-Diagramm while

stat -> while exp do block end

2.4.3 Repeat-Until-Loop

repeat

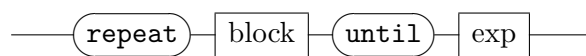


Figure 11: Rail-Diagramm repeat

stat -> repeat block until exp

2.5 Functions

funcname

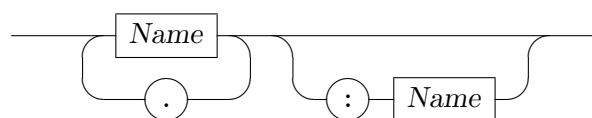


Figure 12: Rail-Diagramm funcname

funcname -> Name { `'.'` Name } [`':'` Name]

varlist1

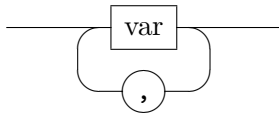


Figure 13: Rail-Diagramm varlist1

varlist1 -> var { ',' var }

var

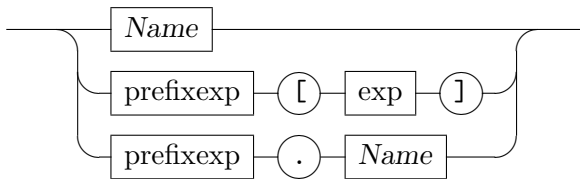


Figure 14: Rail-Diagramm var

var -> Name | prefixexp '[' exp ']' | prefixexp '.' Name

namelist

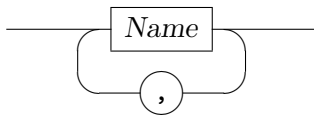


Figure 15: Rail-Diagramm namelist

namelist -> Name { ',' Name }

2.6 Value Assignment

init

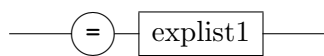


Figure 16: Rail-Diagramm init

init -> '=' explist1

explist1

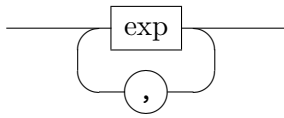


Figure 17: Rail-Diagramm explist1

`explist1 -> { exp ',' } exp`

2.7 Expressions

exp

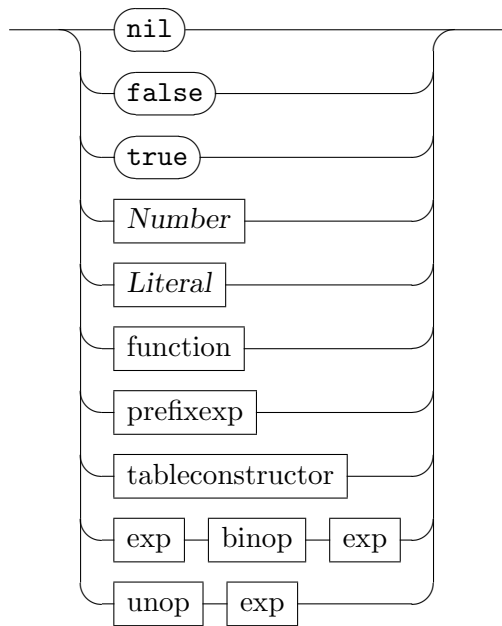


Figure 18: Rail-Diagramm exp

`exp -> nil | false | true | Number | Literal
| function | prefixexp | tableconstructor | exp binop exp | unop exp`

prefixexp

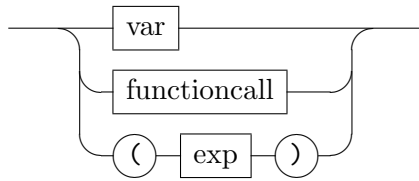


Figure 19: Rail-Diagramm prefixexp

`prefixexp -> var | functioncall | '(' exp ')'`

functioncall

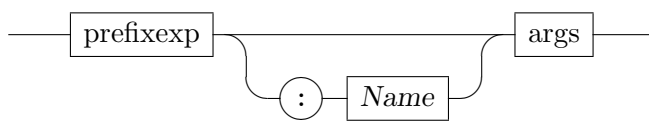


Figure 20: Rail-Diagramm functioncall

`functioncall -> prefixexp args | prefixexp ':' Name args`

args

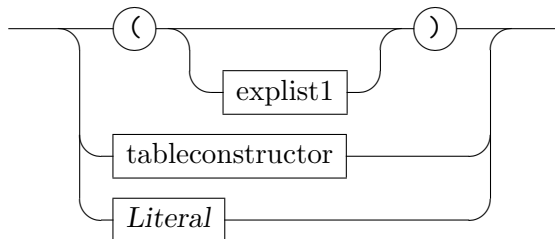


Figure 21: Rail-Diagramm args

`args -> '(' [explist1] ')'` | tableconstructor | Literal

function

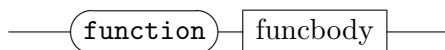


Figure 22: Rail-Diagramm function

`function -> function funcbody`

funcbody

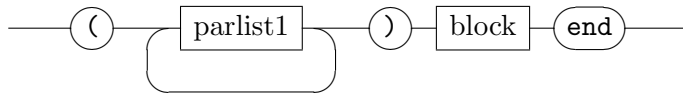


Figure 23: Rail-Diagramm funcbody

`funcbody -> '(' [parlist1] ')' block end`

parlist1

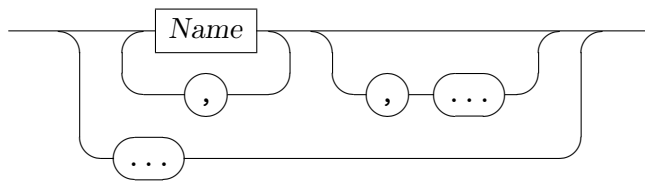


Figure 24: Rail-Diagramm parlist1

This *parlist1* defines again *namelist*. *parlist1variant* is a shorter variant with *namelist-parlist1variant*

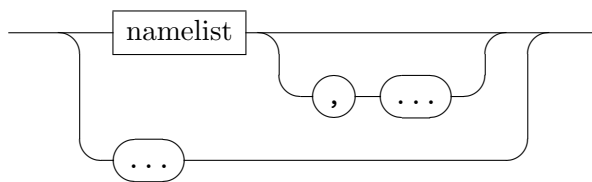


Figure 25: Rail-Diagramm parlist1variant

`parlist1 -> Name { ',' Name } [',' '...'] | '...'`

2.8 Tables

tableconstructor



Figure 26: Rail-Diagramm tableconstructor

`tableconstructor -> '{' [fieldlist] '}'`

fieldlist

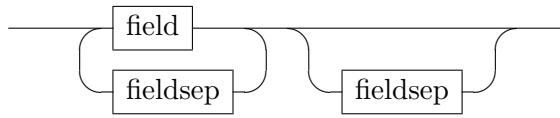


Figure 27: Rail-Diagramm fieldlist

`fieldlist -> field { fieldsep field } [fieldsep]`

Why is there a concluding *fieldsep*?
field

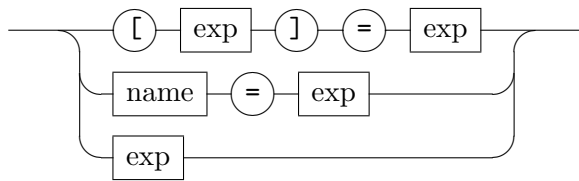


Figure 28: Rail-Diagramm field

`field -> '[' exp ']' '=' exp | name '=' exp | exp`

fieldsep

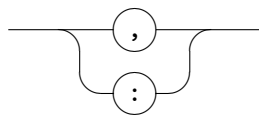


Figure 29: Rail-Diagramm fieldsep

`fieldsep -> ',' | ';'`

2.9 Operators

binop

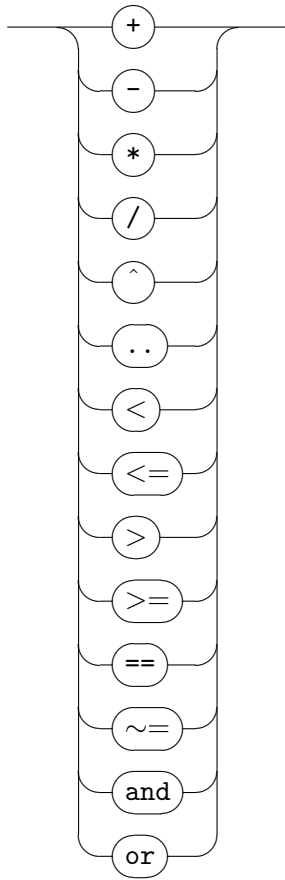


Figure 30: Rail-Diagramm binop

```
binop -> '+' | '-' | '*' | '/' | '^' | '..'  
        | '<' | '<=' | '>' | '>=' | '==' | '~='  
        | and | or
```

unop

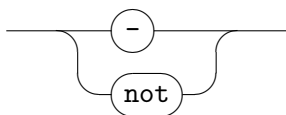


Figure 31: Rail-Diagramm unop

```
unop -> '-' | not
```

3 Additional Diagramms

string

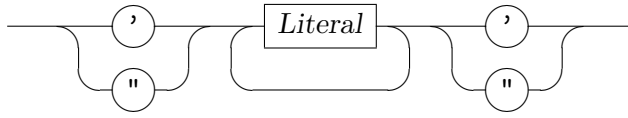


Figure 32: Rail-Diagramm string

print

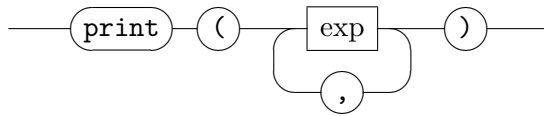


Figure 33: Rail-Diagramm print