Å Official Documentation

The official specification for Å

Contents

Abstract **1**

Using the Virtual Machine **2**

Parsing **3**
- Lexical Analysis . . . 3
- Parse tree (PT) . . . 3
- Abstract Syntax Tree (AST) . . . 3
- Simplifying the AST . . . 3

Static Analysis **4**
- Type declarations . . . 4
- Function Matching . . . 4
- Building Inheritance Trees . . . 4
- Control path verification . . . 4

Type System **5**
- Static Type Environment . . . 5
- Dynamic Type Environment . . . 5
- Typechecking . . . 5

Internals of the Virtual Machine **6**
- Executing bytecode . . . 6
- The operand and call stacks . . . 6
- The Ref Value Environment . . . 6
- The Garbage Collector . . . 6

Structure of an Å-binary file **7**

Compiler **8**
- Compiling the AST . . . 8
- Verifying Å Bytecode . . . 8

Appendix **9**
- Appendix 1: Complete Overview of Grammar . . . 9
- Appendix 2: Complete Overview of Operational Semantics . . . 11
- Appendix 3: Complete Overview of Typing Rules . . . 12
- Appendix 4: Complete Overview of Compile Errors . . . 13
- Appendix 5: Complete Overview of Instruction List . . . 14

Abstract

This document is the official document for the Å programming language. It shows how a program is parsed from source code into a binary format and then how the binary is executed. Thus this document will also contain the semantics of the language and an explanation for the default implementation of the language. Besides explaining the semantics and syntax, this document will also explain how the source code is parsed. Additionally, how this parse result is operated on to convert it into a valid piece of Å code.

Anything written or discussed within this document is only safely assumed to be applicable for Å version 1.0. Therefore, code and other semantics may be outdated or even too new (or features not implemented yet). This document should, therefore, only act as documentation or as guidance for the previously specified version of Å.

Using the Virtual Machine

The virtual machine is first and foremost a command-line tool. It can - on Windows 10 - be run as an executable, wherein it will expect some arguments be given to it. The arguments given to the application will then be used to determine what the application will actually do.

The machine is capable of compiling source code into Å byte code. This byte code can then be executed by the machine directly after compilation (Such that it acts as an interpreter of the source code) or be executed later. Besides compiling and executing Å code, the machine is also capable of generating a readable text version of the generated byte code. Letting the user read the result of the compilation. To add to this, the compiler can also unparse the given Abstract Syntax tree that was created during the parsing of the source code. This can help in giving the programmer a bigger insight into the interpretation of the code.

Parsing

...

Lexical Analysis

...

Parse tree (PT)

...

Abstract Syntax Tree (AST)

...

Simplifying the AST

...

Static Analysis

...

Type declarations

...

Function Matching

...

Building Inheritance Trees

...

Control path verification

...

Type System

...

Static Type Environment

...

Dynamic Type Environment

...

Typechecking

...

Internals of the Virtual Machine

...

Executing bytecode

...

The operand and call stacks

...

The Ref Value Environment

...

The Garbage Collector

Structure of an Å-binary file

...

Compiler

...

Compiling the AST

...

Verifying Å Bytecode

...

Appendix

The following pages are the appendices of this specification and contain full overview over the grammar, semantics, type rules, compile errors, runtime errors, and instruction list.

Appendix 1: Complete Overview of Grammar

The following is a BNF grammar - extended with '*' (Kleene Star)[1] and '+' (Kleene Plus).

Statement ::= *Expr*;
| *Id* = *Expr*;

Expr := (*Expr*)
| *Expr BinaryOp Expr*
| *PreUnaryOp Expr*
| *Expr PostUnaryOp*
| *IntLit*
| *FloatLit*
| *DoubleLit*
| *BoolLit*
| *NullLit*
| *StringLit*
| *CharLit*
| *Id*

Decl ::= var *Id* = *Expr*;
| const var *Id* = *Expr*;
| *TypeId Id* = *Expr*;
| class *Id* { *Decl** }
| class *Id*() { *Decl** }
| class *Id*(*Param*) { *Decl** }
| *TypeID Id*() { *Statement** }
| void *Id*() { *Statement** }
| *TypeID Id*(*Param*) { *Statement** }
| void *Id*(*Param*) { *Statement** }

Param ::= *TypeID Id* | *Param*, *Param*

BinaryOp ::= + | - | * | / | % | < | > | <= | >= | == | != | += | -= | ||
| && | | | & | << | >>

PreUnaryOp ::= - | ! | #

PostUnaryOp ::= ++ | --

Digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

IntLit ::= *Digit*+

FloatLit ::= *Digit*+ . *Digit*+f

DoubleLit ::= *Digit*+ . *Digit*+

[1] https://en.wikipedia.org/wiki/Kleene_star

BoolLit	::= `true` | `false`
NullLit	::= `null`
Letter	::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Å | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | å
Id	::= $Letter^{+}Digit^{*}Letter^{*}$ | $\_Id$
Character	::= *Letter* | *Digit* | | _ | `+` | `-` | `*` | `/` | `%` | `#` | < | > | `=` | | | `&` | `$` | `€` | `@` | `£` | `!` | `?` | `:` | `;` | `,` | `.` | `^` | `¨` | `~` | `(` | `)` | `{` | `}` | `[` | `]` | §
StringLit	::= $\text{¨}Character^{*}\text{¨}$
CharLit	::= *'Character'*
TypeID	::= `string` | `bool` | `int` | `float` | `char` | `Any` | *Id*

Appendix 2: Complete Overview of Operational Semantics

...

$$\textsc{Rule}\ \frac{top}{bottom}$$

Appendix 3: Complete Overview of Typing Rules

...

Appendix 4: Complete Overview of Compile Errors

Appendix 5: Complete Overview of Instruction List

A complete and fully up to date list can be found here.

#	Opcode	Arguments	Stack before	Stack after	Description	Notes	Version
0	NOP		TBD	TBD	No operation		V1.0
1	PUSHC		TBD	TBD	No operation		V1.0
2	PUSHN		TBD	TBD	No operation		V1.0
3	PUSHV		TBD	TBD	No operation		V1.0
4	PUSHWS		TBD	TBD	No operation		V1.0
5	ADD		TBD	TBD	No operation		V1.0
6	SUB		TBD	TBD	No operation		V1.0
7	MUL		TBD	TBD	No operation		V1.0
8	DIV		TBD	TBD	No operation		V1.0
9	MOD		TBD	TBD	No operation		V1.0
10	NNEG		TBD	TBD	No operation		V1.0
11	CONCAT		TBD	TBD	No operation		V1.0
12	LEN		TBD	TBD	No operation		V1.0
13	INC		TBD	TBD	No operation		V1.0
14	DEC		TBD	TBD	No operation		V1.0
15	SETVAR		TBD	TBD	No operation		V1.0
16	GETVAR		TBD	TBD	No operation		V1.0
17	SETFIELD		TBD	TBD	No operation		V1.0
18	GETFIELD		TBD	TBD	No operation		V1.0
19	GETELEM		TBD	TBD	No operation		V1.0
20	SETELEM		TBD	TBD	No operation		V1.0
21	JMP		TBD	TBD	No operation		V1.0
22	JMPF		TBD	TBD	No operation		V1.0
23	JMPT		TBD	TBD	No operation		V1.0
24	LJMP		TBD	TBD	No operation	Not used	V1.0
25	CALL		TBD	TBD	No operation		V1.0
26	VCALL		TBD	TBD	No operation		V1.0
27	XCALL		TBD	TBD	No operation		V1.0
28	RET		TBD	TBD	No operation		V1.0
29	CMPE		TBD	TBD	No operation		V1.0
30	CMPNE		TBD	TBD	No operation		V1.0
31	LE		TBD	TBD	No operation		V1.0
32	GE		TBD	TBD	No operation		V1.0
33	GEQ		TBD	TBD	No operation		V1.0
34	LEQ		TBD	TBD	No operation		V1.0
35	LNEG		TBD	TBD	No operation		V1.0
36	LAND		TBD	TBD	No operation		V1.0
37	LOR		TBD	TBD	No operation		V1.0
38	BAND		TBD	TBD	No operation		V1.0
39	BOR		TBD	TBD	No operation		V1.0
40	TUPLECMP		TBD	TBD	No operation		V1.0
41	TUPLENEW		TBD	TBD	No operation		V1.0
42	TUPLEGET		TBD	TBD	No operation		V1.0
43	ALLOC		TBD	TBD	No operation	Not used	V1.0
44	ALLOCARRAY		TBD	TBD	No operation		V1.0
45	CTOR		TBD	TBD	No operation		V1.0
46	TRY		TBD	TBD	No operation		V1.0
47	THROW		TBD	TBD	No operation		V1.0
48	BRK		TBD	TBD	No operation		V1.0
49	POP		TBD	TBD	No operation		V1.0
50	CASTI2F		TBD	TBD	No operation		V1.0
51	CASTF2I		TBD	TBD	No operation		V1.0

52	CASTF2I		TBD	TBD	No operation		V1.0
53	CASTS2F		TBD	TBD	No operation		V1.0
54	CASTF2S		TBD	TBD	No operation		V1.0
55	CASTL2F		TBD	TBD	No operation		V1.0
56	CASTL2I		TBD	TBD	No operation		V1.0
57	CASTL2S		TBD	TBD	No operation		V1.0
58	CASTF2L		TBD	TBD	No operation		V1.0
59	CASTI2D		TBD	TBD	No operation		V1.0
60	CASTS2D		TBD	TBD	No operation		V1.0
61	CASTF2D		TBD	TBD	No operation		V1.0
62	CASTL2D		TBD	TBD	No operation		V1.0
63	CASTD2I		TBD	TBD	No operation		V1.0
64	CASTD2S		TBD	TBD	No operation		V1.0
65	CASTD2F		TBD	TBD	No operation		V1.0
66	CASTD2L		TBD	TBD	No operation		V1.0
67	WRAP		TBD	TBD	No operation		V1.0
68	UNWRAP		TBD	TBD	No operation		V1.0
69	BCKM		TBD	TBD	No operation		V1.0
70	BDOP		TBD	TBD	No operation		V1.0
71	EXTTAG		TBD	TBD	No operation		V1.0