Compiler

Nick Kramer, Moritz Hinkel, Patrick Küsters, Jiesen Wang

Übersicht

1. Syntax

Allgemeines

Erkannte Zeichenfolge	Zweck
";"	Semikolon zum Abschluss von Zuweisungen / Deklarationen / Funktions-Forward-Deklarationen
"(" ")"	Öffnende / Schließende Klammer zum Klammern von Ausdrücken und für die Kapselung von Parametern in Funktionen
\n	Neue Zeile im Programmcode wird erkannt, die Zeilennummer wird um 1 erhöht
[\t\r]	Leerzeichen, Wagenrücklauf und Tabstopps werden erkannt und ignoriert
"//".*\n	Ein Kommentar wird erkannt. Der Inhalt des Kommentars wird vom Compiler ignoriert
"$debug"	Ermöglicht eine Ausgabe auf der Kommandozeile (Syntax: $debug(“Ausgabe”);)

Deklaration von Variablen, Konstanten

Datentyp / Datenstruktur	Identifier-Erkennung	Literal-Erkennung	Beispiel für Deklaration
Integer	"int"	0 [1-9][0-9]*	int a = 10;
Character	"char"	"""[a-zA-Z0-9_]?"""	char a = 'c';
Boolean	"bool"	("TRUE"|"true") ("FALSE"|"false")	bool a = true;
Float	"float"	([1-9][0-9]*|0)\.[0-9]*	float a = 1.45;
String	"string"	"\""[^\"]*"\""	string a = “abc”;
Array		“[“ “]” “[Integer]” (Angabe der Größe bei Deklaration)	float [2] a[];
Array		"{" "}" "," (Separation von Array-Einträgen)	float [2] a[] = {1,2};
Variable		[a-zA-Z_][a-zA-Z0-9_-]*	datatype x = value;
Konstante	(“const”|”CONST)”		const int a = 10;

Wertzuweisungen

Operation/Vergleich	Zeichen-Erkennung
Zuweisung	“=”
Addition	“+”
Subtraktion	“-”
Multiplikation	“*”
Division	“/”
Potenzieren	“^”
Modolo	“%”
Gleichheit	“==”
Kleiner als	“<”
Kleiner gleich	“<=”
Größer als	“>”
Größer gleich	“>=”
logisches und	“&”
logisches oder	“|”
logisches nicht	“!”

Kontrollstrukturen

Kontrollstruktur (grob)	Kontrollstruktur (einzeln)	Zeichen-Erkennung
if-then(-elif-else)-end	if	("if"|"IF")
	then	("then"|"THEN")
	else if	("elif"|"ELIF")
	else	("else"|"ELSE")
	end	("end"|"END")
while-do Schleife	while	("while"|"WHILE")
	do	("do"|"DO")

Deklarationen und Zuweisungen

Eine korrekte Deklaration folgt folgender Syntax:

```
Datentyp Variablenname;
Datentyp [Array-Größe(int)] Arrayname;
const Datentyp Konstantenname = Wert;
```

Einer Variable kann bei der Deklaration auch ein Wert zugewiesen werden. Dies erfolgt nach folgender Syntax:

```
Datentyp Variablenname = Wert;
Datentyp [Array-Größe(int)] Arrayname = {Wert, Wert, ...};
```

Ausdrücke

Ausdrücke werden nach Präzedenz aufgelöst. Hier die Ebenen in aufsteigender Reihenfolge:

- logische-Operationen (&, |)
- Strich-Operationen (+, -)
- Punkt-Operationen (*, /, %)
- Potenz-Operationen (^)
- Literale, Klammern und logisches nicht

Unter Literalen sind alle Werte eines Ausdrucks zusammengefasst. Dies umfasst positive/negative Zahlen, logische Werte (true/false), sowie Variablen (auch negativ) und einzelne Elemente aus einem Array.

Es werden unvollständig und vollständig geklammerte Ausdrücke unterstützt.

Kontrollstrukturen

Der Sprachumfang umfasst zwei Kontrollstrukturen, zum einen die if-else Kontrollstruktur, zum anderen die while-Schleife.

```
IF Bedingung THEN
      Programm;
ELSE
      Programm2;
END
```

```
IF Bedingung THEN
      Programm;
ELIF Bedingung THEN
      Programm2;
ELSE
      Programm3;
END
```

```
WHILE Bedingung
DO
      Programm
END
```

```
DO
      Programm
WHILE Bedingung
END
```

Eine Bedingung kann jegliche Ausdrücke (expressions) annehmen. Ein Programm kann eine Deklarationen, eine Zuweisungen und weitere Kontrollblöcke sein.

2. Beispielprogramme

Eine Reihe von Testprogrammen kann im Github-Verzeichnis unter */testfiles/* gefunden werden.

3. Nutzung des Compilers

Der Compiler wurde unter Linux (Ubuntu) entwickelt und getestet. Ein Skript zum Bauen liegt unter */src/CompileYaccLex.sh* ab.

```
./CompileYaccLex yacc.y lexAnalysis.l
```

Im Verzeichnis */src/* befindet sich nun der Compiler unter dem Namen “iguanacompiler”.

Um den Compiler ohne “./” ausführen zu können, muss vorher das install-Script durchgeführt werden:

```
./install.sh
```

Um ein Programm zu kompilieren, muss folgendes auf der Kommandozeile stattfinden:

```
iguanacompiler programm [-v] [-s]
```

Die Optionen sind optional und haben folgende Auswirkungen:

- -v Ausgabe der Variablenliste
- -s Ausgabe des Syntaxbaumes

Dabei wird immer zuerst die Variablenliste ausgegeben.

4. Ausblick

4.1 Kontrollstrukturen

Aufnahme der Kontrollstrukturen (IF ELIF ELSE) in den Syntaxbaum.
Spezielle Nodes für Kontrollstrukturen, die Bedingungen und Programme als Kinder besitzen.
Die Bedingung ist in Form einer expr_lvl1. Sie werden anhand des Konzepts “truthiness” ausgewertet (0 → FALSE, Alles andere → TRUE)
Das Programm wird nur ausgeführt, wenn die Bedingung erfüllt wird (muss bei der Auswertung beachtet werden)

4.2 Funktionen

Implementierung eines “Programmstack” für Funktionsaufrufe auf verschiedenen Ebenen (Ähnlich wie progRoot → eigener Syntaxbaum)
Zusätzlich eine eigene linked List für Variablem im Scope der Funktion.
Beim Funktionsaufruf werden die Variablen mit den Ausdrücken der Argumentliste des Aufrufs gefüllt.
Funktionsaufrufe befinden sich im Syntaxbaum des Hauptprogrammes/des Parentaufrufes.
Nach Auswertung der Funktion wird das Ergebnis in den Baum des Hauptprogrammes/des Parentaufrufes gelinkt.

In der Variablenliste der Funktion gibt es eine Variable, die den return Wert angibt.

4.3 Auswertung Syntaxbaum (Interpreter)

Ablauf des Syntaxbaums in inorder Reihenfolge.
Auswertung der Ausdrücke und rückwirkendes Einsetzen in den Syntaxbaum.

Ziel → Die Variablenliste enthält am Ende die errechneten Werte.

4.4 Generelle Features

4.4.1 Cascading Cast

Bei der Typangleichung von Expressions (INT → FLOAT) werden die expressionValues aller Kindknoten auf FLOAT gesetzt, um bei der Auswertung die Numerischen Werte casten zu können.

4.4.2

Zuweisung an einzelne Stellen eines Undefinierten (Nur Deklaration) Arrays.
Dabei wird bei der Deklaration das gesamte Array allokiert.
Bei dem Zugriff auf nicht definierte Arrayfelder muss zur Laufzeit ein Fehler gemeldet werden.

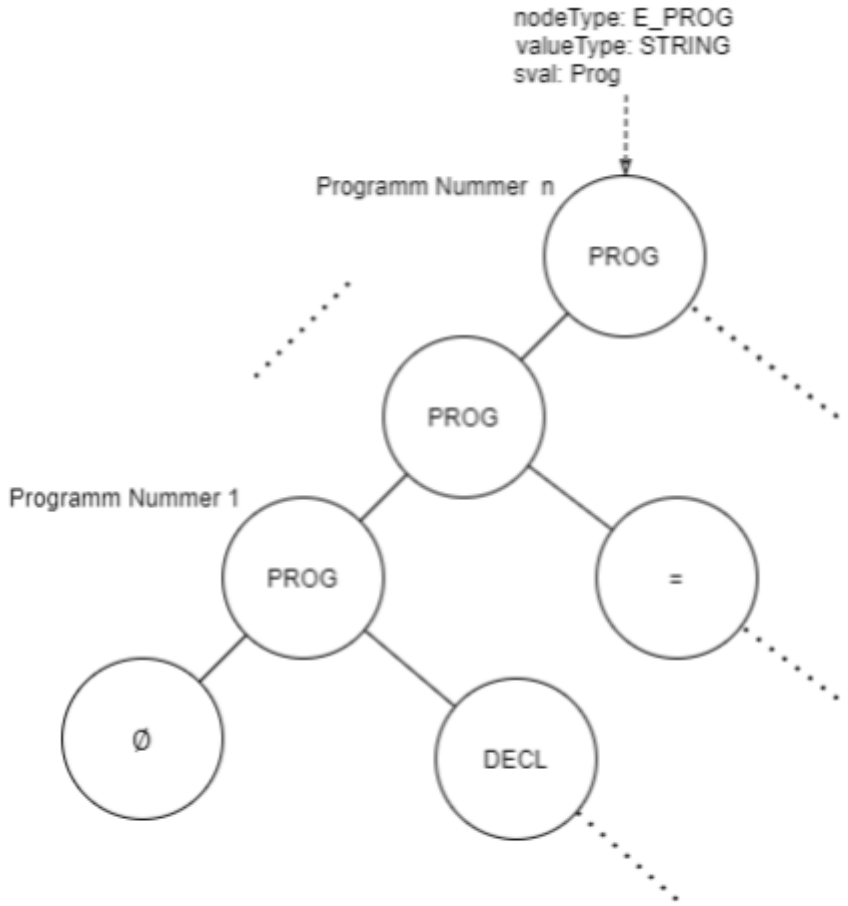
4.4.3 Truthiness bei Booleschen Variablen

Bools können auch numerische Werte (5 ,3.141) annehmen. Der Wert 0 entspricht FALSE alles ander TRUE.

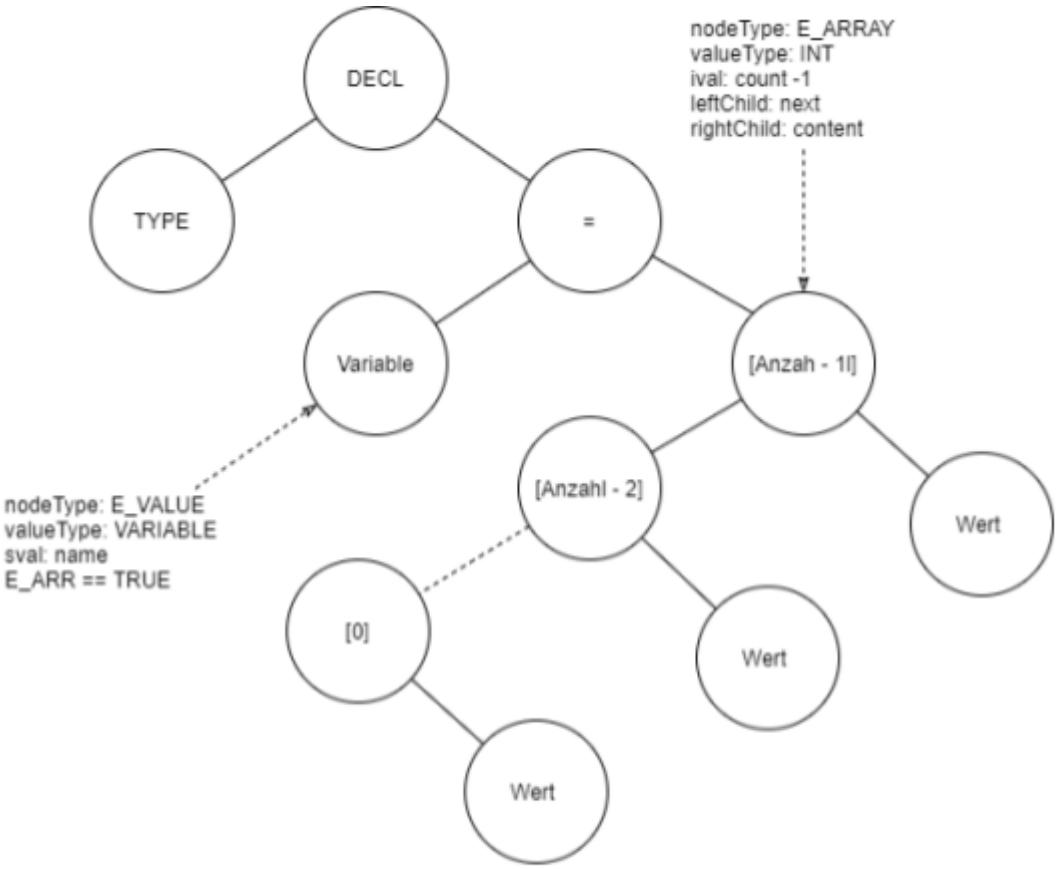
4.4.4 Array Zugriffe mittels Variablen und expressions

Bei der Auswertung des Syntaxbaumes werden die Werte dann eingesetzt und überprüft (Index out of bounce etc).

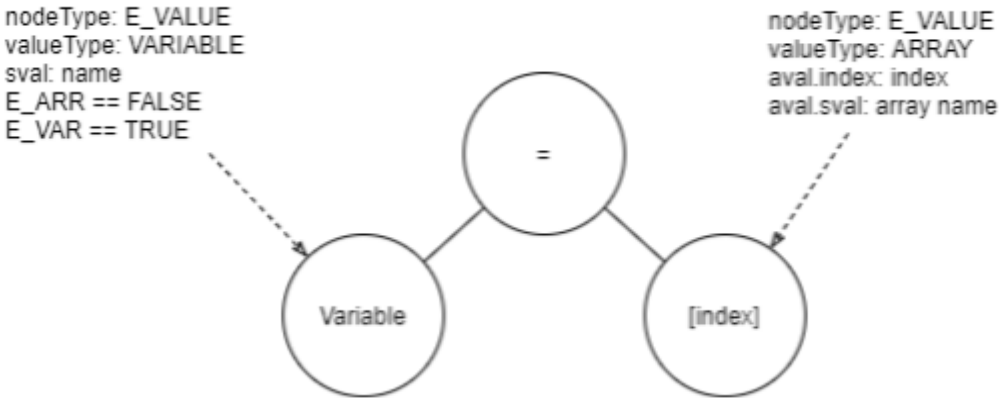
Anhang



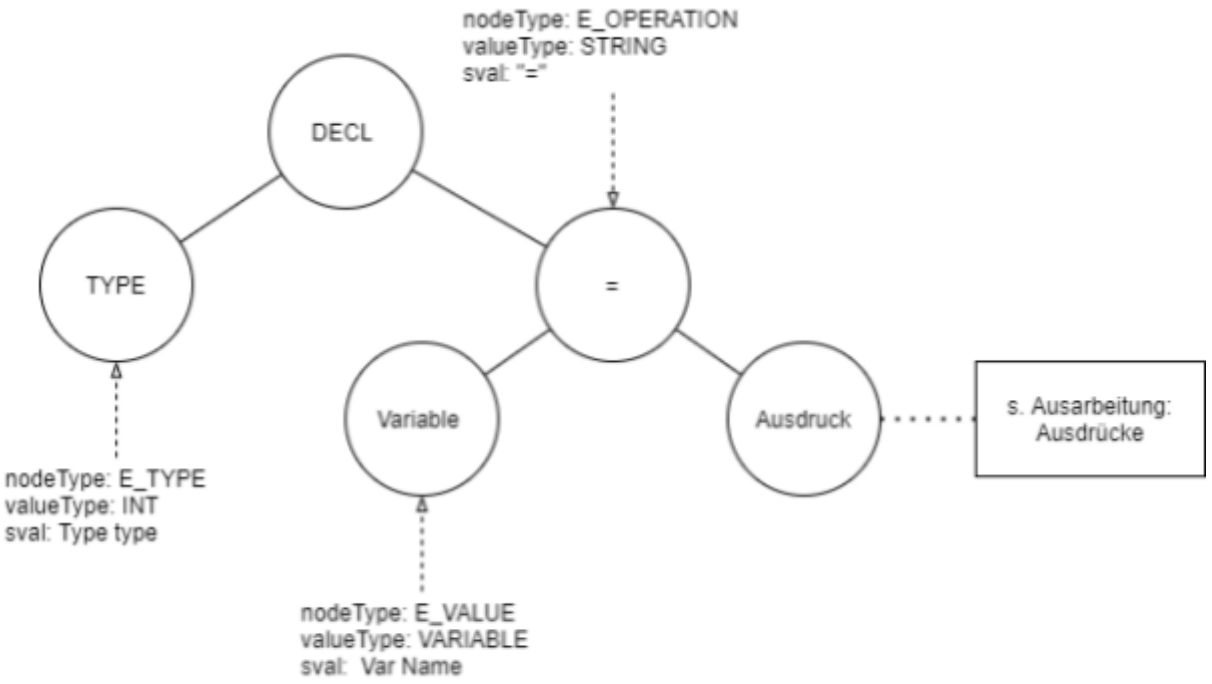
Syntaxbaumausschnitt einer Programmstruktur



Syntaxbaum der Arraydeklaration

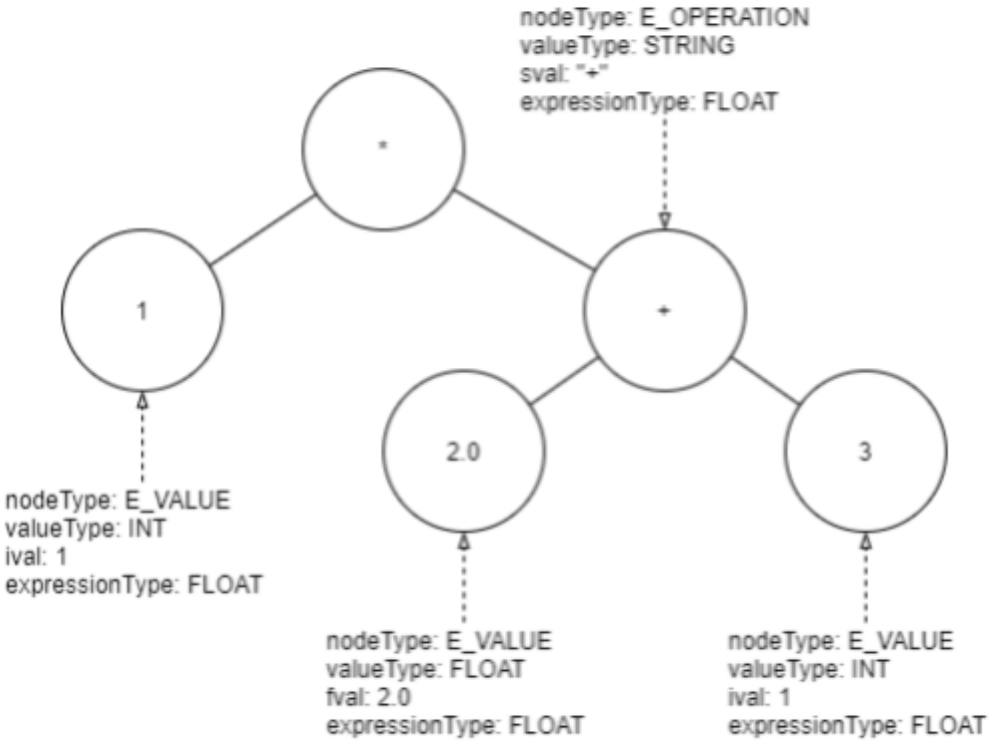


Syntaxbaumausschnitt eines Array-Blatts



Syntaxbaumausschnitt einer Zuweisung

1 * (2 + 3)



Beispiel: Ausdruck