

Christian Maurer

m μ ,
die winzige Programmiersprache

v. 29. November 2010

Freie Universität  Berlin

Dr. Christian Maurer
Keithstr. 16
10787 Berlin

<http://murus.org/>

Die Quelltexte des Modula-2 UML Sums sind mit größter Sorgfalt entwickelt und werden laufend gepflegt. Kein Programmsystem dürfte jedoch jemals frei von Fehlern sein; deshalb ist auch nicht damit zu rechnen, daß *dieses* System fehlerfrei ist: Es darf nur benutzt werden „wie es ist“.

Die Modula-2- und die Java-Quelltexte von Murus sind freie Software. Sie können sie unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation veröffentlicht, weitergeben und/oder modifizieren, entweder gemäß Version 3 der Lizenz oder (nach Ihrer Wahl) jeder späteren Version. Ihre Veröffentlichung erfolgt in der Hoffnung, daß sie Ihnen von Nutzen sein könnten – aber *ohne irgendeine Garantie*, auch ohne die implizite Garantie der *Marktreife* oder der *Verwendbarkeit für einen bestimmten Zweck*.

Der Originaltext der GPL ist im weltweiten Netz unter der Adresse www.gnu.org/licenses/gpl.html zu finden; eine deutsche Übersetzung unter www.gnu.de/documents/gpl-3.0-de.html.

Die Quelltexte von Murus sind nur zu Lehrzwecken konstruiert und haben rein akademischen Wert. Ihre Verwendung in Programmen könnte zu *Schäden* führen, z. B. zur Inbrandsetzung von Rechnern, zur Entgleisung von Eisenbahnzügen, zum GAU in Atomkraftwerken oder zum Absturz des Mondes ...

Satz: Autor mit $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\text{T}\mathcal{E}\mathcal{X}$

Hamse't nich'ne Numma kleena . . .

Berliner Schnauze

Vorwort

In der vorliegenden vorerst minimalistischen Dokumentation ist der Kern einiger Ideen notiert, die im Informatik-Unterricht einen Einstieg in echte informatische Inhalte mit minimalem Programmieraufwand ermöglichen.

$m\mu$ (sprich: „mu“) befindet sich zur Zeit noch in Entwicklung; sowohl das Programm $m\mu$ als auch diese Dokumentation sind daher auf absehbare Zeit ein „*moving target*“.

Christian Maurer

Inhaltsverzeichnis

ALLGEMEINES

Zum Konzept 1

DIE SPRACHE

Die Grammatik von $m\mu$ 2

Bibliotheken 4

DAS PROGRAMM

Hinweise 7

ALLGEMEINES

`mu` ist als Bestandteil des `ModUliverSumS` nach dessen Installation oder Aktualisierung ebenfalls installiert.

Voraussetzung für die Installation des `ModUliverSumS` ist die Installation von **Mocka**, dem Modula-2-Compiler der GMD (Hinweise dazu finden sich unter lwb.mi.fu-berlin.de/inf/mocka).

Arbeitsverzeichnis

Das Arbeitsverzeichnis, aus dem heraus `mu` aufgerufen wird und in dem ggf. Daten abgelegt werden, ist `$HOME/.mu` (es sei denn, daß durch die Umgebungsvariable `mu` ein anderes Arbeitsverzeichnis definiert ist). Damit ist sichergestellt, daß Benutzer/innen mit dem Aufruf von `mu` ihre eigenen Daten verwalten.

Wenn dieses Verzeichnis noch nicht existiert, wird es beim ersten Aufruf von `mu` selbsttätig angelegt und es werden Beispieldaten aus dem `ModUliverSumS` dorthinein kopiert, falls es welche gibt.

Parameterfolge	→ (Parameter { , Parameter })
Sequenz	→ Anweisung { Anweisung }
Anweisung	→ Alternative Aktionsaufruf [Terme] wert Term
Alternative	→ wenn BoolTerm dann Sequenz [sonst Sequenz] ende
Nutzung	→ benutzt Bezeichner { Bezeichner }
Modul	→ [Nutzung] Prozedur { Prozedur }
Programm	→ [Nutzung] { Prozedur } Sequenz

Semantische Annotationen

Zahl	<i>Der Wert der Zahl ist $\leq 2^{16}$.</i>
⟨Basistyp⟩Atom	<i>Bezeichner ist in der umgebenden Prozedur vom Basistyp erklärt, Funktionsaufruf ist der Bezeichner einer Basistyp-wertigen Funktion und ggf. passen Terme in Anzahl, Typ und Reihenfolge zu der Folge der Parameter in ihr.</i>
Aktion	<i>In Sequenz kommt wert Term nicht vor.</i>
Funktion	<i>In Sequenz kommt mindestens einmal wert Term vor und in jedem solchen Vorkommen ist Term vom Typ der Funktion.</i>
Anweisung	<i>Aktionsaufruf ist der Bezeichner einer Aktion und ggf. passen Terme in Anzahl, Typ und Reihenfolge zu der Folge der Parameter in ihr.</i>
Nutzung	<i>Bezeichner ist der Name eines Moduls.</i>

Erweiterte Typen

list Basistyp	<i>Listen von Basistyp</i>
praed Basistyp	<i>Funktionen von Basistyp nach bool</i>
funkt Basistyp	<i>Funktionen von Basistyp nach nat</i>
op2 Basistyp	<i>2-stellige Funktionen von Basistyp nach Basistyp</i>
akt Basistyp	<i>Aktionen auf Basistyp</i>

Bibliotheken

Die folgenden Bibliotheken sind implementiert; sie stellen rudimentäre Ansätze für spezielle Verwendungszwecke dar.

Elementare Arithmetik

Die Nachfolger- und Vorgängerfunktionen:

```
funktion N (n: nat): nat
funktion V (n: nat): nat
funktion unendlich (): nat
```

Ein-/Ausgabe

```
aktion nausgeben (n: nat)
funktion nEingabe: nat
aktion bausgeben (n: bool)
funktion bEingabe: bool
aktion causgeben (c: char)
funktion cEingabe: char
```

Graphik

```
aktion Punkt (x, y: nat)
aktion Strecke (x, y, x1, y1: nat)
aktion Kreis (x, y, r: nat)
```

Listen

```
funktion nneu: list nat
funktion nleer (l: list nat): bool
funktion nList (n: nat, l: list nat): list nat
funktion nKopf (l: list nat): nat
funktion nRest (l: list nat): list nat
aktion ntrav (l: list nat; a: akt nat)

funktion bneu: list bool
funktion bleer (l: list bool): bool
funktion bList (b: bool, f: list bool): list bool
funktion bKopf (l: list bool): bool
funktion bRest (l: list bool): list bool
aktion btrav (l: list bool; a: akt bool)

funktion cneu: list char
funktion cleer (l: list char): bool
funktion cList (c: char, f: list char): list char
funktion cKopf (l: list char): char
funktion cRest (l: list char): list char
aktion ctrav (l: list char; a: akt char)
```

Robi

Die Semantik der folgenden Aktionen und Funktionen ist der Spezifikation `Robi.def` aus dem $\text{Mod}_{\text{Ua}}^{\text{Ua}} \text{Univer}^{\text{Sum}}$ zu entnehmen:

```
funktion inLinkerObererEcke: bool
aktion linksDrehen
aktion rechtsDrehen
funktion amRand: bool
aktion laufen1
aktion zuruecklaufen1
funktion leer: bool
aktion leeren1
funktion NachbarLeer: bool
funktion hatKloetze: bool
aktion KloetzeGeben (n: nat)
funktion AnzahlKloetze: nat
aktion legen1
aktion schieben1
funktion markiert: bool
aktion markieren
aktion entmarkieren
funktion NachbarMarkiert: bool
funktion vorMauer: bool
aktion mauern1
aktion entmauern1
aktion editieren
aktion ausgeben (n: nat)
```

DAS PROGRAMM

Hinweise zum Gebrauch von mu

Nach dem Editieren eines $m\mu$ -Programmtexts als Datei mit dem Suffix `.mu` wird der Übersetzer aufgerufen, wobei der Name des Programmtexts (ohne das Suffix `.mu` als Parameter übergeben wird, z. B. `mu Waechter`).

`mu` erzeugt aus dem $m\mu$ -Programmtext einen Modula-2-Quelltext entsprechenden Namens (z. B. `Waechter.mod`) und übersetzt dieses Programm mit Hilfe von Mocka und dem $\text{ModulenUJiver}^{\text{Sum}}$ in ein ablauffähiges Programm, das durch den Aufruf von `Waechter` mit dem Namen der (vorher mit dem Programm `Robiprog`) erzeugten Stadt als Parameter gestartet wird.