Seminarkurs - Alan Turing

Clara Maier, Nico Fröhlich

October 8, 2019

Contents

1 Vorwort

Der Vater des Computer, die Person die wahrscheinlich den zweiten Weltkrieg um zwei Jahre verkürzte, was gibt es da noch zu sagen? Alan Turing eine der bekanntesten und einflussreichsten Mathematiker des letzten Jahrhunderts. Er beeinflusst nicht nur die Mathematik und heutige Informationstechnik, durch die Turingmaschine zum Beispiel, sondern auch die Philosophie, durch zum Beispiel den Turing Test. In seinem meist Zitierten Papier ging es sogar um Biologie. Am bekanntesten, nicht zuletzt durch den Film 'the imitation game', ist jedoch seine Beteiligung am zweiten Weltkrieg, durch die Entschlüsselung der Enigma der Nazis. Ich, als Informatiker, bin natürlich ein großer Fan seiner Arbeit im Bereich Computer Technik sowie , durch mein Zeitweises Interesse in Ethik und Philosophie, sein Papier über Maschinen und Intelligenz. In den Folgenden Kapiteln wird die tragische Geschichte rund um Alan Turing und ein paar seiner Zeitgenossen beleuchtet. Die Turing Maschine welche mit dem Lambda calculus gegenübergestellt wird, der Turing Test und seine Auswirkungen auf die Zukunft, sowie auf (...) werden in den Folgenden Abschnitten beleuchtet.

2 Historie

3 Turing Maschine

4 Lambda calculus

Der Lambda calculus ist ein Datenverarbeitungskonzept erfunden von Alonzo Church, dem Professor von Alan Turing. Turing studierte unter Church einige Zeit an der Princeton Universität bevor er dann wieder zurück nach England ging. Im folgenden Abschnitt wird kurz auf Church eingegangen, danach wird Churches Werk, der Lambda calculus, dessen Gesetze, Notation und Beispiele behandelt. Hier wird ein Einblick in die Welt der funktionellen Programmiersprachen und ihrer Geschichte gegeben.

4.1 Alonzo Church

4.2 Gesetze des Lambda calculus

Der Lambda calculus besteht aus 3 Grundgesetzen.

- Funktionen können definiert werden.
- diese können/müssen Daten Ein und Ausgeben.
- Funktionen können Angewendet werden

Ausführlicher: Es muss ein Weg vorhanden sein eine Funktion zu definieren, dafür gibt es eine Mathematische Definition. Die Erzeugung einer Funktion ist aber nicht auf die Mathematische Notation beschränkt, ganz im Gegenteil, die Notationen im Modernen Lambda Programming weichen, nicht zuletzt aus technischen Gründen, stark von der Mathematischen Notation ab. Dabei müssen im originalen Lambda Ein- und Ausgaben definiert werden. Dies weicht im modernen Lambda ebenfalls von der originalen Definition ab. Hier können Ein- und Ausgaben definiert werden, müssen es aber nicht. Dies führt zu mehr Flexibilität, so können die Lambda Funktionen auch an stellen Eingesetzt werden wo nicht direkt Daten verarbeitet werden. Als letztes kennt man beim Ausführen der Funktion den Inhalt der

Funktion nicht. Das ist der große unterschied zu anderen Mathematischen Funktionen. [1]

4.3 Lambda in Programmiersprachen

Schauen wir uns zur Erläuterung ein Beispiel aus Java an. Es gibt eine Funktionsdeklaration die der Implementation vorgibt welche Ein und Ausgänge eine gewisse Lambda Funktion hat sowie ihren Namen.

```
@FunctionalInterface
public interface TestFunctionalInterface {
        public int test(double d);
}
```

Die Funktion wird nun als Interface übergeben und von einer ausführenden Funktion genutzt. Was hier auffällt ist das die ausführende Funktion nicht weiß was die Funktion eigentlich tut.

```
public void runTest(TestFunctionalInterface interf) {
        int testres = interf.test(89.1);
        // Mache etwas mit dem resultat
}
```

Normalerweise müsste man dieses Interface überschreiben und dort dann implementieren was die Funktion tun würde. Hier gibt es allerdings eine separate schreib weise um eine Funktion zu definieren. Die wie folgt durch die Zeichen Kombination -¿ eingeleitet wird. Rechts von dem sog. Lambda operator stehen die Inputs rechts die Operationen bzw. Outputs Im folgenden wird die gerade erzeugte

```
public void main() {
        runTest(input -> (int)Math.ceil(input));
}
```

4.4 Mathematische Notation

Für den Ursprünglichen Lambda calculus definiert von Alonzo Church gibt es auch eine Mathematische Notation die wie Folgt durch das Zeichen Lambda eingeleitet wird.

$$\lambda x.x + 1 \tag{1}$$

Diese Funktion beschreit eine Inkrement Funktion. Dabei ist es wichtig anzumerken das wir nicht wissen was die Funktion wirklich tut. Wir können nur sagen was in die Funktion herein gegeben wird und was am ende von der Funktion heraus kommt. Wobei die Inputs mit dem λ (lambda) Symbol Gekennzeichnet sind und mit einem Punkt enden. Am Schluss Folgt der Output. Um es nochmal zu betonen wir wissen nicht was die Funktion im inneren tut. Sie (die Funktion) könnte beim ausführen $x + 2 - 1$ rechnen oder $x + 4 - 3$ das kann man in diesem Model nicht beschreiben, klar ist nur es kommt immer $x + 1$ heraus. Die Ursprüngliche Notation ist weit aus Mathematischer als die in derzeitigen Programmiersprachen. Dennoch lassen sich damit beliebig Daten durch das sog. Kodieren wie im Folgenden Beispiel verarbeiten. Gegeben sind zwei Funktionen.

$$TRUE = \lambda x.\lambda y.x \tag{2}$$

$$FALSE = \lambda x.\lambda y.y \tag{3}$$

Diese zwei Funktionen sollen nun zu einem Logischen Und-Gatter zusammen gefügt werden. Dabei entsteht folgende Funktion.

$$AND = \lambda x.\lambda y.xyx \tag{4}$$

Nun wird auf die und Funktion als Beispiel die Funktionen TRUE und FALSE angewendet. Dazu werden einfach die variablen mit den als Eingabe verwendeten Funktionen ausgetauscht.

$$\text{AND TRUE TRUE} = (\lambda x.\lambda y.xyx)\ \text{TRUE TRUE} \tag{5}$$

$$\text{TRUE TRUE TRUE} = (\lambda x.\lambda y.x)\ \text{TRUE TRUE} = \text{TRUE} \qquad (6)$$

[1] [1]

[1]Kapitel 4 von Nico

5 Turing-Church-Hypothese

6 Turingtest

References

[1] P. G. Hutton, "Lambda calculus - computerphile." https://www.youtube.com/watch?v=eis11j_iGMs, 2017. Aufgerufen 24.9.2019.