
Der Begriff Berechnungsmodell

Die Lernumgebung Kara bietet nicht nur einen Einstieg in die Grundlagen von Algorithmen und Programmen, sondern auch eine intuitive Einführung des zentralen Begriffs „Berechnungsmodell“. Welche Aufgaben kann Kara lösen, welche nicht? Kann Kara mehr Aufgaben lösen, wenn er Kleeblätter als Markierungen legen kann? In diesem Kapitel beschäftigen wir uns deshalb mit Berechnungsmodellen und der Frage, wie Kara in der Hierarchie dieser Modelle einzuordnen ist.

Jede Aussage über die Möglichkeit oder Unmöglichkeit einer Berechnung oder deren Komplexität bezieht sich immer auf ein streng mathematisch definiertes Modell, welches die Daten und erlaubten Operationen festlegt. Aussagen über Berechnungen treffen auf einige Modelle zu, und sind falsch in anderen. Strenge Definitionen sind Voraussetzung für Unmöglichkeitsbeweise der Art „es gibt keinen Algorithmus ...“. Zu unterscheiden sind dabei Modelle für spezielle Zwecke und universelle Berechnungsmodelle, die beliebige andere Modelle simulieren können.

Algorithmus und Berechenbarkeit sind ursprünglich intuitive Begriffe. Eine intuitive, informelle Vorstellung genügt meistens, um zu zeigen, dass ein gewünschtes Resultat gemäss einem vorliegenden Algorithmus berechnet wird. Anders ist der Sachverhalt, wenn wir zeigen wollen, dass ein gewünschtes Resultat nicht berechenbar ist. Sofort drängt sich die Frage auf, welche Mittel wir benützen dürfen. Denn alles wäre berechenbar mit Hilfe eines Orakels, das Antworten auf alle denkbaren Fragen kennt. Das Ziel, die Nichtexistenz von Algorithmen mit einem gegebenen Verhalten zu beweisen, zwingt uns, eine strenge Definition des Begriffs Algorithmus zu erfinden.

Die Frage „Was ist algorithmisch berechenbar, und was nicht?“ wurde um 1930 durch die Logiker Emil Post (1897–1954), Alan Turing (1912–1954), und Alonzo Church (1903–1995) studiert. Sie definierten verschiedene formale Berechnungsmodelle, um den intuitiven Begriff „Rechnen durch Anwendung vorgeschriebener Regeln“ scharf zu fassen: Systeme von Produktionen, Turing Maschinen, rekursive Funktionen, Lambda Kalkül. Alle diese Berechnungsmodelle erwiesen sich als äquivalent. Diese Tatsache stärkt unser Vertrauen in die sogenannte These von Church, dass der intuitive Begriff Algorithmus korrekt formalisiert wurde.

5.1 Geometrische Konstruktionen mit Zirkel und Lineal

Die Berechnungsmodelle, die von den erwähnten Logikern in den 30er Jahren eingeführt wurden, sind universell in dem Sinn, dass man damit alles berechnen kann, was man mit einem beliebigen anderen Modell berechnen kann. Jedes dieser Modelle kann jedes andere simulieren.

Bereits viel früher hat die Mathematik beschränkte, „special-purpose“ Berechnungsmodelle studiert. Mit diesen lassen sich nur ganz bestimmte Berechnungen durchführen. Daher eignen sie sich besonders im Unterricht für den Einstieg ins Thema Berechnungsmodelle. Wir betrachten als Beispiel die schon von den alten Griechen gestellte Frage, welche geometrischen Konstruktionen sich mit Zirkel und Lineal durchführen lassen. Eine der klassischen Fragestellungen ist die Dreiteilung eines beliebigen Winkels. Solche Fragestellungen sind allgemein verständlich und allfällige Lösungen lassen sich gut anhand geometrischer Konstruktionen visualisieren.

Um korrekte Aussagen machen zu können, muss das zugrunde liegende Konstruktionsmodell präzise definiert werden. Wir zeigen, dass schon kleinste Änderungen im Modell zu völlig anderen Aussagen führen können. Zuerst betrachten wir das klassische Modell der euklidischen Geometrie. Mit Zirkel und Lineal konstruieren wir geometrische Gebilde, dargestellt als Zeichnungen auf einem Blatt Papier. Ausgehend von gegebenen Strecken und Winkeln können wir neue Strecken und Winkel konstruieren durch Halbierung, Addition und Subtraktion. Basierend auf den Strahlensätzen ist die Multiplikation und Division von Strecken möglich, und aufgrund des Höhensatzes kann aus einer Strecke die Quadratwurzel gezogen werden (Abbildung 5.1). Zudem kann auf die Schnittpunkte von beliebigen Strecken und Kreisen zugegriffen werden.

Erst im 19. Jahrhundert konnte bewiesen werden, dass in diesem Modell, also allein mit Zirkel und Lineal, die Dreiteilung eines beliebigen Winkels nicht möglich ist. Allgemein gilt der mathematische Satz, dass eine geometrische Konstruktion genau dann mit Zirkel und Lineal konstruierbar ist, wenn die Zahlen, welche analytisch die gewünschten geometrischen Elemente definieren, aus den gegebenen Zahlen durch eine endliche Anzahl von rationalen Operationen und das Ziehen von Quadratwurzeln berechnet werden können. Damit kann gezeigt werden, dass zum Beispiel ein reguläres Polygon mit 17 Seiten konstruiert werden kann. Hingegen ist die Dreiteilung eines beliebigen Winkels oder die berühmte Quadratur des Kreises unmöglich. Die Kernidee dieser Beweise ist die Reduktion geometrischer Konstruktionen auf algebraische Berechnungen.

Wenden wir uns nun einem Konstruktionsmodell zu, dass sich nur leicht vom Modell der Konstruktionen mit Zirkel und Lineal unterscheidet. Archimedes beschrieb schon vor über 2000 Jahren ein Verfahren zur Dreiteilung eines beliebigen Winkels $\angle AOB$ mit Zirkel und Lineal, vorausgesetzt der Lineal BC enthalte zwei Marken C und D mit Abstand 1. Der gegebene Winkel $\angle AOB$ soll mit Scheitel im Mittelpunkt in einem Einheitskreis gegeben sein (Abbildung 5.2). Der Lineal soll stets den Punkt B berühren, während sein

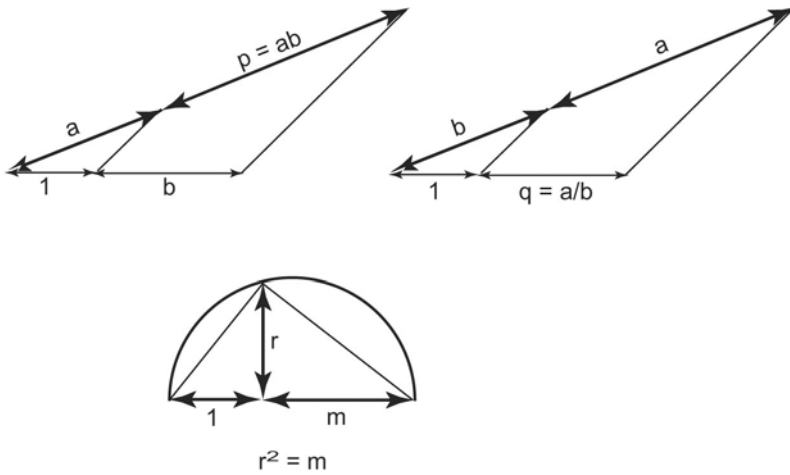


Abb. 5.1. Multiplikation, Division und Wurzel mit Zirkel und Lineal

Ende C sich entlang der Gerade OA bewegt, bis die Marke D genau auf den Einheitskreis zu liegen kommt. In dieser Lage ist der Winkel $\angle ACB$ genau $1/3$ des gegebenen Winkels $\angle AOB$.

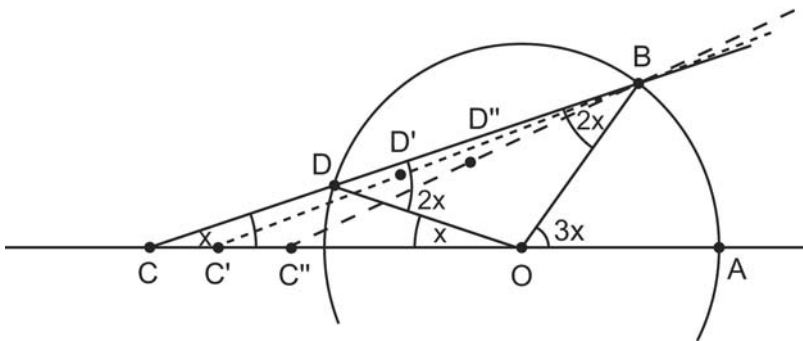


Abb. 5.2. Winkeldreiteilung nach Archimedes

Archimedes erweiterte die Funktionalität der Werkzeuge Zirkel und Lineal auf scheinbar bescheidene Art. Erstens ist auf dem Lineal ein Einheitsintervall markiert; zweitens dürfen wir den Lineal so verschieben, dass eine Marke mit einer Linie zur Deckung gebracht wird. Besonders diese Schiebe-Operation ist im Standardmodell mit Zirkel und Lineal nicht erlaubt. Das Beispiel zeigt, dass kleine Änderungen im Berechnungsmodell drastische Auswirkungen haben können. Es kommt auf die Details der Definition des Erlaubten an!

Im vorangehenden Beispiel haben scheinbar bescheidene Erweiterungen der Funktionalität der Werkzeuge dazu geführt, dass ein im ursprünglichen Berechnungsmodell unlösbares Problem auf einfache Art lösbar wird. Betrachten wir noch den umgekehrten Fall und schränken wir das Berechnungsmodell „Zirkel und Lineal“ ein, indem wir den Lineal weglassen. Auf den ersten Blick scheint die Menge der allein mit dem Zirkel möglichen Konstruktionen drastisch kleiner zu sein. Betrachtet man aber eine Gerade als konstruiert, wenn zwei Punkte der Geraden gegeben sind, so stellt man überraschenderweise fest, dass jede mit Zirkel und Lineal mögliche Konstruktion auch mit dem Zirkel allein möglich ist. Dieses Resultat geht auf Georg Mohr (1640–1697) zurück, wird aber häufiger Lorenzo Mascheroni (1750–1800) zugewiesen.

Zum Beweis dieses Sachverhaltes genügt es, die drei Grundkonstruktionen Schnitt zweier Kreise, Schnitt eines Kreises mit einer Geraden und Schnitt zweier Geraden allein mit dem Zirkel durchzuführen. Als Beispiel sei der Schnitt eines Kreises K mit einer durch zwei Punkte P_1 und P_2 bestimmten Geraden angeführt, die nicht durch das Zentrum des Kreises geht. Die Idee ist einfach: Mit dem Zirkel lässt sich der Mittelpunkt des gegebenen Kreises, und somit der Kreis, an der Geraden spiegeln. Die Schnittpunkte des gegebenen und des gespiegelten Kreises bestimmen dann die Schnittpunkte der Gerade mit dem Kreis.

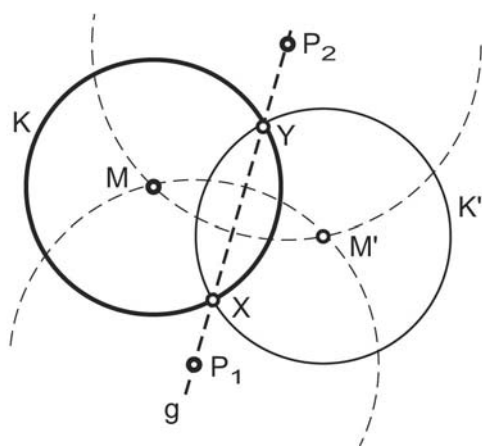


Abb. 5.3. Schnitt Kreis/Gerade nach Mohr-Mascheroni

Einen elementaren Beweis des Satzes von Mohr-Mascheroni findet man beispielsweise in [Hun94]. Natürlich sind jetzt noch weitere Variationen des Modells „Konstruktionen mit Zirkel und Lineal“ möglich. So konnte Grossmann um 1900 zeigen, dass ausgehend von drei linear unabhängigen Kreisen alle geometrischen Konstruktionen mit Zirkel und Lineal auch nur mit dem Lineal allein möglich sind.

Diese Beispiele zeigen, dass Zirkel allein und auch Lineal allein erstaunlich mächtige Werkzeuge sind. Sie illustrieren eine Beobachtung, die in der Theorie der Berechnung immer wieder auftritt: Mit wenigen erlaubten Operationen können komplexe Verfahren implementiert werden. Das ist eines der Grundprinzipien beim Programmieren.

5.2 Welche Aufgaben kann Kara in seiner Welt lösen?

Die Programmierumgebung Kara eignet sich, um Fragen rund um Berechenbarkeit zu thematisieren. Welche Aufgaben kann Kara überhaupt lösen? Das führt zu der für die Informatik wichtigen Frage der Mächtigkeit verschiedener Berechnungsmodelle. Was kann mit welchen Rechenhilfen berechnet werden, und was nicht? Anhand des endlichen Automaten Kara illustrieren wir verschiedene Berechnungsmodelle und zeigen, wie wichtig es auch hier ist, die erlaubten Operationen präzise zu definieren. Der Einfachheit halber betrachten wir eine idealisierte Version von Kara und nehmen an, die Welt sei unbeschränkt gross. Zudem gehen wir davon aus, dass sie keine Pilze enthält.

„Read Only“-Welt – der eigentliche endliche Automat

Als einfachsten Fall nehmen wir an, Kara dürfe die Welt nicht verändern, also keine Kleeblätter legen oder aufnehmen. Der Speicher in Form von Kleeblättern als Markierungen in der Welt wird also von Kara nicht benutzt. In diesem Fall sind seine Programme echte endliche Automaten.

Liegt in der Welt kein einziges Kleeblatt, kann Kara nur in der Welt herumwandern. Welche Art von Wanderungen kann Kara in der leeren Welt machen? Kara kommt schnell entweder zum Stillstand, oder er verfällt in ein periodisches Muster, das er endlos wiederholt (Abbildung 5.4 links). Einen rechteckig-spiralförmigen Weg (Abbildung 5.4 rechts) hingegen kann Kara in einer leeren Welt nicht ablaufen. Kara kann zwar seine Zustände als Speicher verwenden und sich so die aktuelle Seitenlänge der Spirale merken. Um die verschiedenen Seitenlängen der Spirale in einer Welt von unbeschränkter Grösse auseinander halten zu können, müsste Kara aber über unbeschränkt viele Zustände verfügen.

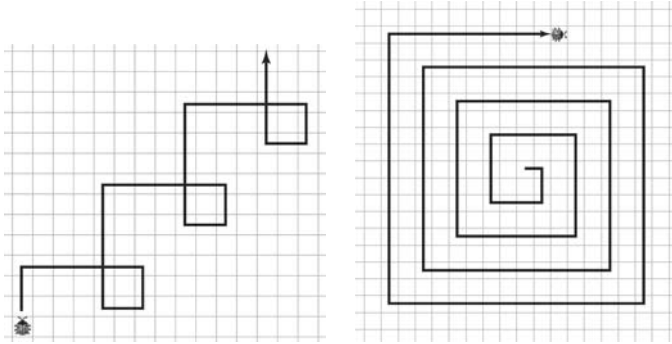


Abb. 5.4. Periodische Wanderung (links) und unmögliche Wanderung (rechts)

Mustererkennung in nicht leeren Welten

Als nächstes betrachten wir den Fall, dass die Welt nicht leer ist, sondern Kleeblätter und Bäume enthält. Der Speicher der Welt wird bei diesen Aufgaben nur lesend, read-only, benutzt. Als Beispiel soll Kara prüfen, ob das vorgegebene Muster „Feld ohne Kleeblatt, Feld mit Kleeblatt, Feld ohne Kleeblatt“ auf den Feldern bis zum nächsten Baum vorkommt.

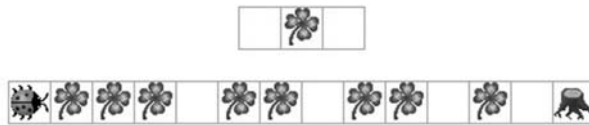


Abb. 5.5. Ein einfaches Muster und die abzusuchende Welt

Das Programm in Abbildung 5.6 erkennt, ob das Muster bis zum nächsten Baum vorkommt oder nicht. Die Antwort auf diese Frage gibt Kara durch die Richtung, in die er sich vor dem Ende der Programmausführung dreht: nach links bedeutet, das Muster kommt vor, nach rechts, es kommt nicht vor. Kara startet im Zustand **start**. Ist das Feld unter ihm leer, macht er einen Schritt vorwärts und wechselt in den Zustand 0. Liegt auf dem Feld unter ihm ein Kleeblatt, so macht er wiederum einen Schritt vorwärts und wechselt in den Zustand 01. Jetzt hat er zwei Drittel des Musters erkannt. Sieht er nun ein leeres Feld, so hat er das Muster vollständig erkannt. Bei einem Kleeblatt wechselt Kara wieder in den Zustand **start** zurück und beginnt von vorn.

Aufgaben zur Mustersuche führen zur Theorie der regulären Sprachen. Eine reguläre Sprache ist eine Menge von Zeichenketten, die nach recht einfachen Gesetzmäßigkeiten aufgebaut sind. Die Frage ist, ob man Kara so programmieren kann, dass er genau die Zeichnungen akzeptiert, die Element einer

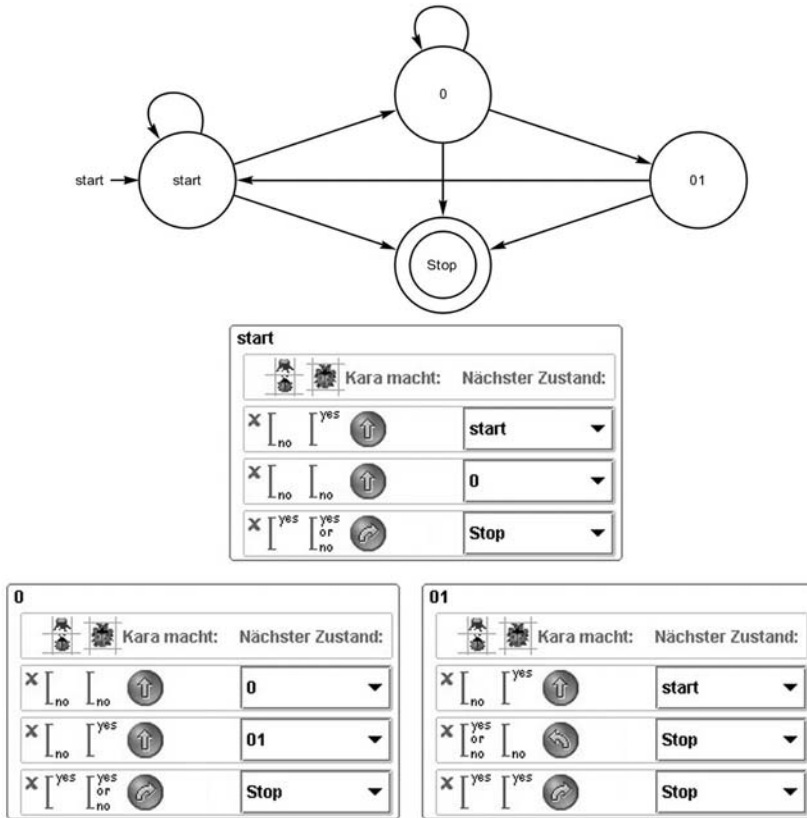


Abb. 5.6. Programm für die Mustererkennung

gegebenen Sprache sind, und alle anderen verwirft. Dabei darf Kara die Welt nicht verändern: Was er sich merken möchte, muss er in seinem eigenen Zustandsraum speichern. Abbildung 5.7 zeigt links eine reguläre Kleeblattsprache. Kara kann erkennen, ob eine gegebene Kleeblattfolge aus einer beliebigen Anzahl „Feld ohne Kleeblatt, Feld mit Kleeblatt“ besteht. Ein einfaches Programm mit zwei Zuständen kann diese Entscheidung treffen. Das Programm funktioniert analog zum Beispiel in Abbildung 5.6.

Hingegen kann Kara nicht entscheiden, ob er zuerst eine beliebige Anzahl von Feldern ohne Kleeblätter und danach die gleiche Anzahl von Feldern mit Kleeblättern sieht (Abbildung 5.7 rechts). Dazu müsste er sich mit Hilfe von verschiedenen Zuständen merken, wie viele Felder ohne Kleeblatt er schon gesehen hat. Das geht nur, wenn die maximale Länge des Patterns vorher bekannt ist und ein Automat für diese maximale Länge erstellt wird. In einer unbeschränkten Welt würden unendliche viele Zustände benötigt.

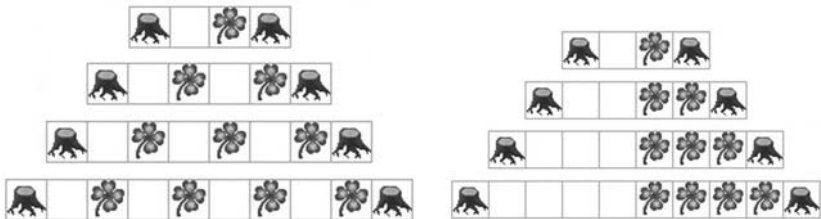


Abb. 5.7. Reguläre und nicht reguläre Kleeblattsprache

„Read-Write“-Welt – Turing-Maschinen

Nehmen wir an, Kara dürfe in einer unbeschränkt grossen Welt an beliebigen Orten Kleeblätter hinlegen und entfernen. Dann dient ihm die Welt als unbeschränkt grosser „read-write“ Speicher. Mit dieser Annahme wird Kara zum mächtigsten Berechnungsmodell in der Hierarchie der Automaten – zur Turing Maschine, die unter geeigneter Kodierung der Eingabe und Ausgabe alles berechnen kann, was algorithmisch berechenbar ist.

Jetzt kann Kara auch nicht reguläre Kleeblattsprachen wie etwa die Sprache aus Abbildung 5.7 (rechts) erkennen. Abbildung 5.8 zeigt, wie sich die Welt während des Programmablaufs verändert. Der Einfachheit halber nehmen wir an, der Käfer stehe zu Beginn bereits vor dem ersten Kleeblatt. Die Position des Startfelds markiert der Käfer, indem er in der Zeile oberhalb ein Kleeblatt ablegt. Dann verschiebt er alle Kleeblätter von rechts soweit wie möglich nach links. Zum Schluss wechselt er in die obere Zeile. Steht er nun auf dem zu Beginn abgelegten Kleeblatt, dann ist die untere Zeile nach dem vorgegebenen Muster „ n Felder ohne Kleeblatt, n Felder mit Kleeblatt“ aufgebaut.

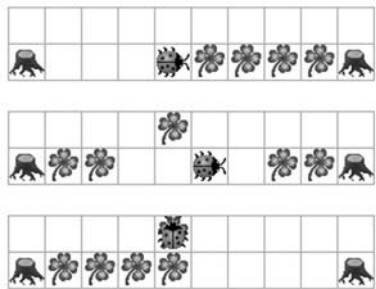


Abb. 5.8. Mustererkennung für „ n Felder ohne, n Felder mit Kleeblatt“

Dieses Beispiel zur Erkennung nicht regulärer Sprachen zeigt die Mächtigkeit des Berechnungsmodell von Kara auf. Sobald Kara Kleebblätter legen und aufnehmen kann und seine Welt unbeschränkt gross ist, ist er so mächtig wie ein Turingmaschine. Es liegt deshalb auf der Hand, die Lernumgebung Kara auch zur Illustration und Animation von Turing-Maschinen einzusetzen. Damit wird die Programmierungsumgebung Kara auch für Überlegungen im Bereich der Theoretischen Informatik nutzbar.

Programmieren mit Kara

Ein spielerischer Zugang zur Informatik

Reichert, R.; Nievergelt, J.; Hartmann, W.

2005, X, 152 S., Softcover

ISBN: 978-3-540-23819-5