Humboldt-Universität zu Berlin

Hausarbeit im Fach Informatik für das Praxissemester

eingereicht von

Michael Röse

bei

Dr. Nguyen-Thinh Le

17. März 2020

Die selbstständige und eigenhändige Ausfertigung versichert an Eides statt

Berlin, 17. März 2020

Michael Röse

Inhaltsverzeichnis

1 Einleitung **1**

2 Die Unterrichtsreihe **2**

2.1 Bedingungsanalyse 2

2.1.1 Klassensituation und Lernvoraussetzungen 2

2.1.2 Einordnung der Stunde in den RLP oder internen Curriculum 4

2.2 Didaktische Strukturierung 5

2.2.1 Sachanalyse 5

2.2.2 Didaktische Analyse 12

2.2.3 Methodische Analyse 15

3 Unterrichtsentwurf einer Doppelstunde **17**

3.1 Einordnung der Stunde in die Unterrichtseinheit 17

3.2 Didaktische Überlegung und Begründung 19

3.3 Unterrichtsentwurf einer Doppelstunde 20

3.4 Geplanter Stundenverlauf 22

3.5 Beschreibung der Unterrichtsdurchführung 22

3.6 Reflexion der Stunde 25

4 Kollegiale Hospitation **27**

4.1 Vorüberlegungen zur Auswertung 27

4.2 Reflexion 27

5 Reflexion des Praktikums **31**

6 Der Alternativentwurf **34**

6.1 Problemanalyse 34

6.2 Lösungsvorschläge 37

6.3 Der überarbeitete Entwurf 38

6.4 Didaktische Begründung 40

Literatur **41**

Anhang **43**

A Raumplan S15 **43**

B Material zur Doppelstunde Einführung deterministischer endlicher Automaten **44**

B.1 Tafelbilder 44

B.2 Schatzkarte 45

B.3 Wegweiserkarten 46

B.4 Arbeitsblätter 48

B.5 Folien 50

C Material zum Alternativentwurf **54**

C.1 Startbild der Animation 54

C.2 Code-Puzzle 54

C.3 Alte Folien 55

C.4 Neue Folien 57

1 Einleitung

Diese Hausarbeit entstand im Rahmen meines Praxissemesters, welches ich vom 01.09.2019 bis zum 31.01.2020 am Leibniz-Gymnasium in Berlin-Kreuzberg absolvierte. In ihr gehe ich detailliert auf die Planung, Durchführung und Reflexion einer Doppelstunde aus der Unterrichtsreihe zum Thema *Sprachen und Automaten* im Informatikleistungskurs Klasse 12 ein. Hierzu gehört auch die vorangestellte Bedingungsanalyse und didaktische Strukturierung in Abschnitt 2. Darüber hinaus wird in Punkt 4 eine kollegiale Hospitation beschrieben und ausgewertet, gefolgt von einer Reflexion des Praktikums in Abschnitt 5. Die Arbeit endet in Punkt 6 mit einer Beschreibung einer Unterrichtsstunde, in welcher Probleme auftraten, analysiert diese und stellt einen Alternativentwurf vor.

2 Die Unterrichtsreihe

2.1 Bedingungsanalyse

Im Zuge der Bedingungsanalyse werden die Einflussfaktoren erörtert, die den geplanten Unterricht bedingen. In Hilbert Meyers Worten: „Es soll geklärt werden, ob die Schülerinnen und Schüler, das, was sie im Unterricht tun sollen, überhaupt können."[8, S.129]

Zur Beantwortung dieser Frage dienen die folgenden Ausführungen, in denen ich näher auf die verschiedenen Voraussetzungen eingehen möchte, mit denen ich konkret konfrontiert war. Dazu gehören die Klassensituation, die Lernvoraussetzungen sowie der Rahmenlehrplan.

2.1.1 Klassensituation und Lernvoraussetzungen

Der in Punkt 3 beschriebene Unterricht fand im Informatik-Leistungskurs Klasse 12 statt. Anders als an anderen Berliner Schulen ist dieser Kurs nicht jahrgangsübergreifend, weswegen von einer geringeren leistungsbezogenen Heterogenität auszugehen war. Dies bestätigte sich auf Nachfrage beim Fachlehrer: Von den insgesamt zehn Schülern des Leistungskurses erbrachten vier durchschnittlich sehr gute Leistungen, drei durchschnittlich gute Leistungen und weitere drei durchschnittlich befriedigende Leistungen im Informatikunterricht.

Auch das fachspezifische Basiswissen der einzelnen Schüler konnte als weitgehend gleichwertig vorausgesetzt werden, da das Leibniz-Gymnasium die Belegung des Leistungskurses Informatik nur dann gestattet, insofern die Schüler das Fach bereits ab Klasse 10 gewählt haben. Da folglich die Schüler des Informatik-Leistungskurses Klasse 12 das Fach nicht nur frühzeitig und vorausschauend, sondern auch freiwillig belegten, war damit zu rechnen, dass sie eine erhöhte Affinität zu den Inhalten des Informatikunterrichts aufweisen würden; wobei erfahrungsgemäß dem Aspekt des Programmierens das größte Interesse zukommt.

Die konkrete Thematik meines Unterrichts betreffend war auf Schülerseite von keinen tieferen Kenntnissen auszugehen. Im vorangegangenen Schuljahr waren lediglich einfache Graphenstrukturen, wie etwa binäre Suchbäume, behandelt worden. Da für das Themengebiet Sprachen und Automaten allerdings keine Vorkenntnisse erforderlich sind, stellte dies keine gesonderte Herausforderung bei der Erarbeitung meiner Unterrichtsgestaltung dar.

Im Hinblick auf das Fach Informatik seien auch die Hackdays erwähnt. Hierbei handelt es sich um drei Projekttage, an denen alle Schüler aus Klasse 10 in Gruppen eigenverantwortlich reale Probleme bzw. Aufgaben im schulischen Umfeld identifizieren, um sich im Anschluss und unter Zuhilfenahme von Informatiksystemen eben dieser Probleme anzuneh-

men. Demnach sammeln die Schüler bereits früh Erfahrungen im Modellieren komplexerer Systeme; eine Kompetenz, deren weitere Ausbildung durch ein im zweiten Semester der Klassenstufe 11 stattfindendes Softwareprojekt ergänzt wird.

Überhaupt liegt ein ausgewiesener Schwerpunkt des Leibniz-Gymnasiums im Bereich Naturwissenschaften. In diesem Zusammenhang wurde es mehrfach als MINT-freundliche Schule mit exzellenter beruflicher Orientierung ausgezeichnet.

Das Leibniz-Gymnasium verfügt über insgesamt drei Informatik-Fachräume, von denen jeder mit jeweils 20 funktionsfähigen Computern ausgestattet ist. Das verwendete Betriebssystem ist Windows 10. Der Unterricht des Leistungskurses findet für gewöhnlich im Raum S15 statt, in dem die Rechner u-förmig entlang der Wände angeordnet sind, so dass die Schüler während ihrer Arbeit an den Computern mit dem Rücken zur Raummitte sitzen. Hier befinden sich weitere Arbeitsplätze, die jedoch über keine PC-Ausstattung verfügen und nach dem Lehrertisch ausgerichtet sind. Der Lehrer-PC ist dauerhaft mit einem Beamer verbunden, welcher ein großes, gut sichtbares Bild projiziert. Bei Bedarf kann der Raum mittels Gardinen abgedunkelt werden. Darüber hinaus verfügt der Fachraum S15 über ein großes, höhenverstellbares Whiteboard, das an der Wand hinter dem Lehrertisch installiert ist. Zu einer Überschneidung von Tafel und Beamerprojektion kommt es dennoch nicht. Ein Raumplan befindet sich in Anhang A.

Im Informatikunterricht am Leibniz-Gymnasium werden keine Lehrbücher verwendet. Auf Nachfrage erfuhr ich, dass die Schule grundsätzlich über keinen Satz an Informatiklehrbüchern verfügt. Als Lehr- und Lernmedien dienen in erster Linie Arbeitsblätter, Kopien aus Lehrbüchern sowie die Plattform MS Teams.

Um die Schüler des Leistungskurses Informatik Klasse 12 noch etwas weiter zu charakterisieren, möchte ich auch auf deren hohe sprachlichen Kompetenzen im Bereich Kommunizieren und Argumentieren eingehen; Fähigkeiten, die auch im Informatikunterricht geschult werden. Die Schüler waren mit abstrakten, symbolischen Schreibweisen vertraut und konnten diese sachgerecht anwenden. Für die Kommunikation innerhalb des Kurses, aber auch für die Bereitstellung von Materialien, kam wiederholt Microsoft Teams zum Einsatz. Die Schüler waren im Umgang mit dieser Software sicher und nutzten sie ohne Probleme. Generell herrschte ein sehr gutes Klassenklima. Die Lernenden ließen einander ausreden, gaben konstruktives Feedback und arbeiteten größtenteils diszipliniert. Zu Unterrichtsstörungen kam es äußerst selten. Typischerweise unterstützten die Schüler einander, etwa bei Verständnisfragen.

Wie ausgeprägt die Eigenverantwortung der Klasse war, zeigte sich nicht zuletzt daran, dass es im Zuge eines krankheitsbedingten Ausfalls des Fachlehrers sogar möglich war, einen der Schüler mit Moderations- und Koordinationsverantwortlichkeiten zu betrauen. Indem dieser die vorgegebenen Aufgabenstellungen präsentierte und das Vergleichen der Ergebnisse organisierte, verlief die Übungseinheit trotz Abwesenheit des Lehrers weitestgehend geordnet.

2.1.2 Einordnung der Stunde in den RLP oder internen Curriculum

Grundsätzlich ist der Berliner Rahmenlehrplan für Informatik in der Oberstufe sehr offen gehalten. Mit dem Themengebiet Sprachen und Automaten sollen die Schüler zur abstrakten, zustandsorientierten Modellierung komplexer Systeme befähigt werden. In den Worten der Senatsverwaltung: „Durch die Einführung des Automatenmodells vertiefen die Schülerinnen und Schüler ihr Verständnis von Informatiksystemen.“[1] Die Doppelstunde *Einführung deterministische endliche Automaten*, welche in Abschnitt 3 vorgestellt wird, legt hierfür den Grundstein.

Das schulinterne Curriculum legt die Kompetenzschwerpunkte in dem Themengebiet auf Informatisches Modellieren, mit Information umgehen und Informatiksysteme verstehen.[6] Die Doppelstunde wirkt auch hier vorbereitend, da in ihr das vom Schulcurriculum vorgeschriebene Modell des endlichen Zustandsautomaten vorgestellt wird. In diesem Zusammenhang werden wesentliche Grundbegriffe und Schreibweisen eingeführt und gefestigt. Auf diese Weise wird den Lernenden der Ausbau besagter Kompetenzen auf dem Feld der Sprachen und Automaten erst ermöglicht.

Gemäß den Empfehlungen der Gesellschaft für Informatik e. V. sollen die Schüler die Fähigkeiten erlangen, Grammatiken in endliche Automaten zu überführen, et vice versa. Sie sollen weiterhin in die Lage versetzt werden, den Zusammenhang zwischen Grammatiken, Sprachen und Automaten zu erläutern. [9, S. 11] Es versteht sich daher von selbst, dass der Informatikunterricht zunächst die hierfür notwendigen Grundlagen bereitstellen muss, d.h. die zentralen Ideen und essentiellen Bausteine der Reihe zu motivieren und verständlich zu machen. Hierzu soll die Doppelstunde einen großen Beitrag leisten.

2.2 Didaktische Strukturierung

Da die Unterrichtsreihe in erster Linie eine Heranführung der Schüler an Automaten, formale Sprachen und Grammatiken leisten soll, wird dieses einigermaßen komplexe Thema zunächst auf die Betrachtung von endlichen Automaten und regulären sowie kontextfreien Grammatiken reduziert. Insbesondere mithilfe des Pumping-Lemmas gelingt eine Charakterisierung von regulären und kontextfreien Sprachen in der Chomsky-Hierarchie. Turingmaschinen und Entscheidbarkeit sind nicht Teil der Unterrichtsreihe. Beide Aspekte sind ans Ende des Semesters verlagert.

Im Folgenden wird nun noch einmal genauer auf die fachlichen Hintergründe der Unterrichtseinheit sowie die Bewandtnis und Struktur ihrer Vermittlung einzugehen sein.

2.2.1 Sachanalyse

Der inhaltlichen Aufbau dieser Sachanalyse orientiert sich stark am Skript zur Vorlesung *Einführung in die Theoretische Informatik* von Prof. Köbler, welche im Wintersemester 2015/16 gelesen wurde (siehe [4]).

Zunächst führen wir einige Grundbegriffe ein, um im Anschluss den endlichen Automaten definieren zu können. Ein *Alphabet* $\Sigma = \{a_1, \ldots, a_m\}$ ist eine Menge von endlich vielen *Zeichen*. So sind beispielsweise $\{A, B\}, \{0, 1\}, \{a, b, \ldots, z\}$, {aufheben, ablegen} jeweils Alphabete. Eine Folge $x = x_1 \ldots x_n$ von n Zeichen heißt *Wort*. Im Spezialfall $n = 0$ bezeichnen wir das *leere Wort* mit ε. Eine Menge von Wörtern über einem Alphabet Σ heißt *Sprache über* Σ. Um ein Beispiel zu nennen: $\{\varepsilon, A, B, AA, AB, BA, BB, AAA, \ldots\}$ ist eine Sprache über $\{A, B\}$. Darüber hinaus setzen wir $\Sigma^0 := \{\varepsilon\}$ sowie $\Sigma^* := \bigcup_{n \geq 0} \Sigma^n$ und $\Sigma^+ \bigcup_{n \geq 1} \Sigma^n$.

Definition 1: Ein *deterministischer endlicher Automat* (oder kurz DEA) ist ein Tupel $M = (Z, \Sigma, \delta, q_0, E)$, wobei

- $Z \neq \varnothing$ eine endliche Menge von *Zuständen*,
- Σ das *Eingabealphabet*,
- $\delta : Z \times \Sigma \to Z$ die *Überführungsfunktion*,
- $q_0 \in Z$ der Startzustand und
- $E \subseteq Z$ die Menge der *Endzustände* ist.

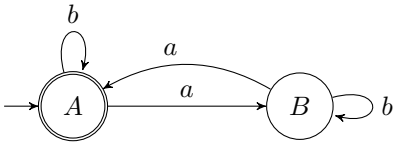


Abbildung 1: Zustandsdiagramm eines Automaten

Neben dieser Tupelschreibweise lassen sich DEAs auch als Zustandsdiagramm visualisieren. Hierbei handelt es sich um einen gerichteten Graphen, dessen Knoten durch die Zustandsmenge Z gegeben sind. Für alle $z \in Z, a \in \Sigma$ wird dann $(z, \delta(z, a))$ als Kante hinzugefügt, wobei das Zeichen a an die Kante geschrieben wird. Knoten zu Endzuständen werden doppelt umrahmt. Der Knoten zum Startzustand wird durch einen auf ihn gerichteten Pfeil (ohne Startknoten) hervorgehoben. Abbildung 1 zeigt beispielhaft das Zustandsdiagramm des DEAs $M = (\{A, B\}, \{a, b\}, \delta, A, \{A\})$ mit Überführungsfunktion gegeben durch

δ	A	B
a	B	A
b	A	B

.

Erhält ein DEA ein Wort x der Länge n als Eingabe, so arbeitet er x von links nach rechts ab. Wir nennen eine Zustandsfolge $q_0, \ldots, q_n \in Z$ *Rechnung* von $M(x)$, falls $\delta(q_i, x_{i+1}) = q_{i+1}$ für alle $i < n$. Sie heißt *akzeptierend*, falls zusätzlich $q_n \in E$. Der DEA *akzeptiert* x, falls er nach dem Lesen des letzten Zeichens auf einem Endzustand endet, d.h. falls eine akzeptierende Rechnung von $M(x)$ existiert.

Definition 2: Die von einem DEA M *akzeptierte Sprache* $L(M)$ ist die Menge aller Wörter, die er akzeptiert:

$$L(M) := \{x \in \Sigma^* : M \text{ akzeptiert } x\}.$$

Eine von einem DEA akzeptierte Sprache nennen wir *regulär*. Die zugehörige Sprachklasse ist

$$\mathsf{REG} = \{L(M) : M \text{ ist ein DEA}\}.$$

Wir können sehr leicht sehr viele Beispiele für reguläre Sprachen angeben. Wie das nächste Lemma zeigt, sind alle Sprachen, die aus einem Wort bestehen, schon regulär.

Lemma 3: Alle Sprachen, die aus einem Wort bestehen, sind regulär.

Beweis. Sei $x \in \Sigma^n$ ein Wort über Σ mit n Zeichen. Wir definieren den Automaten M über das Tupel $(Z, \Sigma, \delta, q_0, E)$ mit $Z = \{q_0, \ldots, q_n, e\}, E = \{q_n\}$ und der Überführungsfunktion δ mit

$$\delta(q, a) = \begin{cases} q_{i+i}, & \text{falls } q = q_i \text{ für ein } i \text{ mit } 0 \leq i < n \text{ und } a = x_{i+1} \\ e, & \text{sonst} \end{cases}$$

für alle $q \in Z, a \in \Sigma$.

Dann ist offensichtlich $q_0, \ldots, q_n$ eine akzeptierende Rechnung für $M(x)$ und es folgt, $x \in L(M)$. Nach Konstruktion von δ ist jede andere Rechnung nicht akzeptierend. Also gilt letztlich $L(M) = \{x\}$. □

Darüber hinaus ist die Sprachklasse REG abgeschlossen unter Komplementbildung sowie unter Schnitt und Vereinigung:

Lemma 4: Seien $L_1, L_2 \in \mathsf{REG}$. Dann sind auch $\overline{L_1} := \Sigma^* \setminus L_1, L_1 \cap L_2$ und $L_1 \cup L_2$ regulär.

Beweis. Gegeben seien zwei DEAs M_1 und M_2 mit $M_i = (Z_i, \Sigma, \delta_i, q_0, E_i)$ für $i = 1, 2$ sowie $L_1 = L(M_1)$ und $L_2 = L(M_2)$. Wir betrachten $\overline{M_1} = (Z_1, \Sigma, \delta_1, q_0, Z \setminus E_1)$. Für $x \in \Sigma^*$ ist die Rechnung von $M_1(x)$ genau dann akzeptierend, falls sie als Rechnung von $\overline{M_1}(x)$ nicht akzeptierend ist. Dies impliziert sofort $\overline{L_1} = L(\overline{M_1})$.

Für den Schnitt betrachten wir den DEA $M = (Z_1 \times Z_2, \Sigma, \delta, (q_0, q_0), E_1 \times E_2)$ mit

$$\delta((q, p), a) = (\delta_1(q, a), \delta_2(p, a))$$

für alle $(q, p) \in Z_1 \times Z_2, a \in \Sigma$. Ist nun $x \in L_1 \cap L_2$ ein Wort mit n Zeichen, so gibt es eine akzeptierende Rechnung $q_0, \ldots, q_n$ von $M_1(x)$ und eine akzeptierende Rechnung $q_0, p_1, \ldots, p_n$ von $M_2(x)$. Leicht sieht man, dass somit $(q_0, q_0), (q_1, p_1), \ldots, (q_n, p_n)$ eine akzeptierende Rechnung von $M(x)$ ist. Andersherum können wir nach Konstruktion für $x \in L(M)$ jede akzeptierende Rechnung $(q_0, q_0), (q_1, p_1), \ldots, (q_n, p_n)$ von $M(x)$ so zerlegen, dass $q_0, \ldots, q_n$ eine akzeptierende Rechnung von $M_1(x)$ und $q_0, p_1, \ldots, p_n$ eine akzeptierende Rechnung von $M_2(x)$ ist. Entsprechend gilt $L(M) = L_1 \cap L_2$.

Da $L_1 \cup L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$, ist schließlich auch diese Sprache regulär. □

Korollar 5: Jede endliche Sprache ist regulär.

Beweis. Dieses Resultat folgt direkt aus Lemma 3 und Lemma 4. □

Sehr ähnlich zu den DEAs führen wir nun die nichtdeterministischen endlichen Automaten ein. Im Unterschied zu den DEAs verfügen sie ggf. über mehrere Startzustände und die

Überführungsfunktion ist von der Form $\Delta : Z \times \Sigma \to \mathcal{P}(Z)$. Dies liefert eine gewisse Flexibilität. So werden wir anschließend sehen, dass die von NEAs akzeptierten Sprachen auch unter Produktbildung und Sternhülle abgeschlossen sind. Gleichzeitig (und auf den ersten Blick vermutlich überraschend) sind die Automaten hinsichtlich der akzeptierten Sprachen den DEAs ebenbürtig. Wir werden sehen, dass jede von einem NEA akzeptierte Sprache regulär ist.

Definition 6: Ein *nichtdeterministischer endlicher Automat* (kurz NEA) ist ein Tupel $N = (Z, \Sigma, \Delta, Q_0, E)$, wobei

- $Z \neq \varnothing$ eine endliche Menge von *Zuständen*,
- Σ das *Eingabealphabet*,
- $\Delta : Z \times \Sigma \to \mathcal{P}(Z)$ die *Überführungsfunktion*,
- $Q_0 \in Z$ die Menge der Startzustände und
- $E \subseteq Z$ die Menge der *Endzustände* ist.

Für $x \in \Sigma^n$ nennen wir eine Folge $q_0, \ldots, q_n \in Z$ *Rechnung* von $N(x)$, falls $q_0 \in Q_0$ und $q_{i+1} \in \Delta(q_i, x_{i+1})$ für alle $i < n$ gilt. Sie heißt wiederum *akzeptierend*, falls $q_n \in E$.

Ein NEA kann bei einer Eingabe also mehrere unterschiedliche Rechnungen aufweisen. Er akzeptiert dabei genau dann ein Wort x, wenn es eine akzeptierende Rechnung von $N(x)$ gibt.

Definition 7: Die von einem NEA N *akzeptierte Sprache* $L(N)$ ist die Menge aller Wörter, die er akzeptiert:

$$L(N) := \{x \in \Sigma^* : N \text{ akzeptiert } x\}.$$

Das folgende Lemma zeigt, dass die von NEAs akzeptierten Sprachen auch unter Produktbildung und Sternhülle abgeschlossen sind:

Lemma 8: Gegeben seien zwei NEAs N_1 und N_2. Dann existieren NEAs N und N^* mit $L(N) = L(N_1)L(N_2) = \{xy : x \in L(N_1), y \in L(N_2)\}$ und $L(N^*) = L(N_1)^*$.

Für den Beweis sei auf das Vorlesungsskript (siehe [4, S. 5]) verwiesen.

Satz 9 (Rabin und Scott)**:** Es gilt $\mathsf{REG} = \{L(N) : N \text{ ist ein NEA}\}$.

Beweis. Die Richtung $\mathsf{REG} \subseteq \{L(N) : N \text{ ist ein NEA}\}$ ist trivial, da jeder DEA auch als NEA aufgefasst werden kann.

Für die Rückrichtung wird die sogenannte Potenzmengenkonstruktion verwendet. Ausgehend von einem NEA $N = (Z, \Sigma, \Delta, Q_0, E)$ definieren wir den DEA $M = (\mathcal{P}(Z), \Sigma, \delta, Q_0, E')$ mit $E' = \{Q \subseteq Z : Q \cap E \neq \varnothing\}$ und

$$\delta : \mathcal{P}(Z) \times \Sigma \to \mathcal{P}(Z), \quad \delta(Q, a) = \bigcup_{q \in Q} \Delta(q, a).$$

Damit ist $\delta(Q, a)$ die Menge der Zustände, zu denen man in N gelangen kann, wenn man von einem beliebigen $q \in Q$ startend das Zeichen a liest. Für diesen Automaten kann man nun zeigen, dass $L(M) = L(N)$ gilt.[4, S. 6] □

Man beachte, dass der Potenzmengenautomat nach Konstruktion $2^{|Z|}$ viele Zustände besitzt. Zwar werden wir uns gleich noch die Minimierung von Automaten ansehen. Jedoch gibt es auch Beispiele für NEAs, bei denen es bei der Potenzmengenkonstruktion zu einem exponentiellen Blowup kommt, der nicht durch eine Minimierung verhindert werden kann.[2, S. 65]

Um potenziell unnötige Zustände aus einem DEA $M = (Z, \Sigma, \delta, q_0, E)$ zu entfernen, betrachten wir zunächst weitere Hilfsautomaten, indem wir den Startzustand variieren. Für $q \in Z$ setze $M_q := (Z, \Sigma, \delta, q, E)$. Wir definieren nun die Relation $\sim$ auf Z via

$$q \sim p \quad \Leftrightarrow \quad L(M_q) = L(M_p).$$

Leicht überprüft man, dass es sich bei $\sim$ um eine Äquivalenzrelation handelt. Schließlich führen wir für eine Menge $Q \subseteq Z$ die Schreibweise $\tilde{Q} = \{[q]_\sim : q \in Q\}$ für die Menge der zugehörigen Äquivalenzklassen ein. Hiermit können wir nun DEAs minimieren.

Satz 10: Sei $M = (Z, \Sigma, \delta, q_0, E)$ ein DEA, der nur Zustände enthält, die vom Startzustand q_0 aus erreichbar sind. Dann ist $M' = (\tilde{Z}, \Sigma, \delta', [q_0]_\sim, \tilde{E})$ mit δ' definiert via

$$\delta'([q]_\sim, a) = \widetilde{\delta(q, a)}$$

ein DEA mit $L(M') = L(M)$ und minimaler Anzahl an Zuständen.

Ein Beweis dieses Satzes ist wiederum in [4, S. 20] zu finden. Hiermit können wir nun einen Algorithmus zur Minimierung von DEAs formulieren. Ein solcher ist mit Algorithmus 1 angegeben. Die Korrektheit der Methode ist mit im Skript enthalten.

Als nächstes beschäftigen wir uns mit dem Pumping-Lemma (für reguläre Sprachen):

Satz 11: Sei L eine reguläre Sprache. Dann existiert eine Zahl ℓ, so dass sich alle Wörter $x \in L$ mit einer Länge $|x| \geq \ell$ in der Form $x = uvw$ zerlegen lassen, wobei

(i) $v \neq \varepsilon$,

(ii) $|uv| \leq \ell$ und

(iii) $uv^i w \in L$ für alle $i \geq 0$.

Die kleinste Zahl ℓ mit diesen Eigenschaften (falls vorhanden) wird *Pumping-Zahl* von L genannt.

Algorithm 1: `min-DEA`(M)

Input: DFA $M = (Z, \Sigma, \delta, q_0, E)$

Output: $M' = (\tilde{Z}, \Sigma, \delta', [q_0]_\sim, \tilde{E})$, wobei für jeden Zustand $q \in Z$ gilt

$$[q]_\sim = \{q' \in Z : \{q, q'\} \notin D\}$$

```
1 begin
2     entferne alle nicht erreichbaren Zustände
3     D' := {{q, q'} : q ∈ E, q' ∉ E}
4     repeat
5         D := D'
6         D' := D ∪ {{q, q'} : ∃a ∈ Σ : {δ(q, a), δ(q', a)} ∈ D}
7     until D' = D
8 end
```

Der Beweis zu Satz 11 nutzt aus, dass reguläre Sprachen von DEAs erzeugt werden und diese eine endliche Anzahl an Zuständen besitzen. Übersteigt die Wortlänge der Eingabe diese Anzahl, so muss der Automat einen Zustand zwangsweise mehrfach durchlaufen. Der gefundene Zykel kann genutzt werden, um die Eingabe gemäß (iii) zu „pumpen“. Der vollständige Beweis ist in [4, S. 23] aufgeführt.

Das Lemma dient in vielen Fällen als hilfreiches Werkzeug um zu zeigen, dass bestimmte Sprachen nicht regulär sind. Aber Vorsicht: Die Aussage des Lemmas liefert nur eine notwendige Bedingung. Es gibt Sprachen, die nicht regulär sind, aber dem Lemma genügen. Als Beispiel hierfür sei auf die nicht reguläre Sprache $L = \{a^i b^j c^k : i = 0 \text{ oder } j = k\}$ verwiesen.[4, S. 25]

Kommen wir nun zu einer anderen Sichtweise auf die formalen Sprachen – zu den Grammatiken.

Definition 12: Eine *Grammatik* ist ein Tupel $G = (V, \Sigma, P, S)$, wobei

- V eine endliche Menge von *Nicht-Terminalen*,
- Σ eine endliche Menge von *Terminalen*,
- $P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$ eine endliche Menge von *Produktionen* und
- $S \in N$ das *Startsymbol* ist.

Seien $\alpha, \beta \in (V \cup \Sigma)^*$. β ist aus α in einem Schritt ableitbar, falls es eine Produktion $u \to v$ sowie Wörter $x, y \in (V \cup \Sigma)^*$ gibt mit

$$\alpha = xuy, \qquad \beta = xvy.$$

In dieser Situation schreiben wir auch $\alpha \Rightarrow \beta$.

Definition 13: Sei $G = (V, \Sigma, P, S)$ eine Grammatik. Die von ihr *erzeugte Sprache* ist

$$L(G) := \{x \in \Sigma^* : \exists n \in \mathbb{N}, \alpha_0, \ldots, \alpha_n \in (V \cup \Sigma)^* : \alpha_0 = S, \alpha_n = x \text{ und } \alpha_i \Rightarrow \alpha_{i+1}\, \forall i < n\}.$$

Definition 14: Sei $G = (V, \Sigma, P, S)$ eine Grammatik.

- G heißt *regulär*, falls für alle Produktionen $u \to v$ gilt: $u \in V$ und $v \in \Sigma V \cup \Sigma \cup \{\varepsilon\}$.
- G heißt *kontextfrei*, falls alle Produktionen von der Form $u \to v$ mit $u \in V$ sind.
- G heißt *kontextsensitiv*, falls für alle Produktionen $u \to v$ gilt: $|v| \geq |u|$.

Wie der Name es schon verrät, erzeugen die regulären Grammatiken genau die regulären Sprachen.[4, S. 26] Damit erhalten wir die folgenden Charakterisierungen für diese Sprachklasse (siehe [4, S. 10/26]):

Satz 15: Sei L eine Sprache. Dann sind die folgenden Aussagen äquivalent:

(i) L ist regulär,

(ii) es gibt einen DEA M mit $L = L(M)$,

(iii) es gibt einen NEA N mit $L = L(N)$,

(iv) L lässt sich über Vereinigung, Produkt und Sternhülle aus endlichen Sprachen gewinnen,

(v) es gibt eine reguläre Grammatik G mit $L = L(G)$.

Mit den verschiedenen Grammatiktypen können wir auch weitere Sprachklassen definieren:

- Die kontextfreien Sprachen

$$\mathsf{CFL} = \{L(G) : G \text{ ist eine kontextfreie Grammatik}\}.$$

- Die kontextsensitiven Sprachen

$$\mathsf{CSL} = \{L(G) : G \text{ ist eine kontextsensitive Grammatik}\}.$$

- Die rekursiv aufzählbaren Sprachen

$$\mathsf{RE} = \{L(G) : G \text{ ist eine Grammatik}\}.$$

Diese können im Hinblick auf Mengeninklusion in einer Rangfolge angeordnet werden, der sogenannten Chomsky-Hierarchie.[4, S. 26ff]

Satz 16 (Chomsky-Hierarchie)**:** Für die Sprachklassen gilt

$$\mathsf{REG} \subset \mathsf{CFL} \subset \mathsf{CSL} \subset \mathsf{RE}.$$

Zuletzt sei noch erwähnt, dass sich zu allen Sprachklassen entsprechende Automatenmodelle finden lassen, welche genau die jeweiligen Sprachen akzeptieren. Für kontextfreie Sprachen gibt es nichtdeterministische Kellerautomaten, für kontextsensitive Sprachen die linear beschränkten Automaten und für die rekursiv aufzählbaren Sprachen schließlich die Turingmaschinen. Diese Automaten sind jedoch nicht Teil der Unterrichtsreihe und werden daher hier nicht weiter thematisiert.

Zur Erinnerung: Turingmaschinen und Entscheidbarkeit sind ans Ende des Semester verlagert.

2.2.2 Didaktische Analyse

Um meine didaktische Analyse möglichst vollständig und übersichtlich zu gestalten, werde ich mich an Klafkis fünf Grundfragen der Unterrichtsvorbereitung orientieren.[5]

Gegenwartsbedeutung

Die Schüler des Informatik-Leistungskurses Klasse 12 sind bereits mit verschiedenen Sprachen vertraut: von natürlichen (wie Deutsch, Englisch und Französisch) über symbolische

(wie in der Mathematik und den Naturwissenschaften) bis hin zu Programmier- und Auszeichnungssprachen (wie Java und html). An diese Vorerfahrungen lässt sich anknüpfen, um eine Abstraktion zu formalen Sprachen zu motivieren, und um Konzepte der Informatik (wie Grammatiken) anhand von Beispielen aus der Lebenswelt der Jugendlichen verständlich zu machen.

Wie zuvor im Gliederungspunkt 2.1.1 *Lernvoraussetzungen* beschrieben, verfügen die Schüler bereits über Kompetenzen im Bereich der Modellierung. Die Thematik der endlichen Automaten erweitert diese im Sinne einer zustandsorientierten Modellierung. Hinzukommt, dass die Lernenden gewisse Automaten aus ihrem Alltag kennen und somit vereinzelte Wirkprinzipien auf die abstrakten Maschinen im Bereich der Informatik übertragen können. Von Seiten der Lehrkraft ist hierbei zu beachten, stets auch die Unterschiede zwischen den alltäglichen und abstrakten Automaten hervorzuheben. Zwar profitiert der Leistungskurs von sehr informatikinteressierten Schülern; da aber deren Fokus mehrheitlich auf dem Programmieren liegt, war nicht davon auszugehen, dass sie mit den zentralen Fragen des neuen Themengebiets befasst sind.

Zukunftsbedeutung

Gerade die Kompetenzen des Modellierens und damit auch die des Problemlösens sind in der heutigen Arbeitswelt relevant und begehrt. Die Unterrichtsreihe hilft, diese Kompetenzbereiche zu schulen und besitzt das Potential, den Schülern ein tieferes Verständnis für Algorithmen und maschinelle Arbeitsweisen zu ermöglichen. In letzter Konsequenz mündet das Themengebiet in den Problemen der Entscheidbarkeit und Berechenbarkeit. Diese Bereiche, die am Ende des Schuljahres bearbeitet werden, beleuchten, wozu Computer überhaupt in der Lage sind und welche Aufgaben unmöglich bleiben. Das endgültige Ziel der Unterrichtsreihe zu besteht darin, den Lernenden ein sachgerechtes Verständnis von Computern zu vermitteln; auch und vor allem in Abgrenzung zu dem allgemein verbreiteten Irrglauben, es handle sich um eine Art „Wundermaschine“, die in der Lage ist, eigentlich jedes Problem „irgendwie“ lösen zu können, insofern man ihr nur ausreichend Zeit lässt.

Der hohe Abstraktionsgrad, den die Inhalte der Unterrichtsreihe verlangen, bereitet die Jugendlichen zudem auf ein späteres Hochschulstudium im informatisch-technischen Bereich vor. Mehrere Abiturienten des Kurses gaben an, ein solches anzustreben.

Sachstruktur

Bei den in der Unterrichtsreihe betrachteten Automaten handelt es sich ausschließlich um DEAs und NEAs. Trotz der geringeren Flexibilität der DEAs, sind sie den NEAs in Hinblick auf die von ihnen akzeptierten Sprachen ebenbürtig. Das heißt, die von NEAs akzeptierten Sprachen stimmen mit jenen von DEAs akzeptierten Sprachen überein. Sie werden in der Klasse der regulären Sprachen zusammengefasst. Grammatiken bieten eine weitere Sichtweise auf die formalen Sprachen. Dabei sind die von regulären Grammatiken erzeugten Sprachen genau die regulären Sprachen. Die Chomsky-Hierarchie zeigt die Abstufungen bzw. Inklusionen der Sprachklassen und setzt damit die regulären und kontextfreien Grammatiken in Relation. Letztere ermöglichen einen größeren Spielraum beim Ableiten von Wörtern, sodass die Menge der kontextfreien Sprachen echt größer als die der regulären ist. Um zu beweisen, dass eine gegebene Sprache kontextfrei und nicht regulär ist, genügt es jedoch nicht eine kontextfreie Grammatik anzugeben, die diese erzeugt. Ein geeignetes Werkzeug hierfür, findet sich stattdessen im Pumping-Lemma.

Exemplarische Bedeutung

Der Themenkomplex *Automaten und Sprachen* beinhaltet die Möglichkeit, Sprachen formal zu beschreiben, und eröffnet den Lernenden eine erweiterte Sichtweise auf bekannte Probleme. Am Ende steht die Fähigkeit alltägliche Phänomene von einem generalisierten Standpunkt betrachten zu können. Darüber hinaus gestattet die Unterrichtsreihe die Einsicht, dass wissenschaftliche Themen oft verschiedene aber gleichwertige Zugänge bieten. Als Beispiel hierfür sind Determinismus und Nichtdeterminismus im Bereich der endlichen Automaten zu benennen. Auch die Äquivalenz solcher Automaten mit regulären Grammatiken hinsichtlich der akzeptierten/erzeugten Sprachen gehört dazu. Vermittelt über die Automaten erhalten die Schüler zeitgleich ein neues Werkzeug zur Problemlösung – die Zustandsmodellierung.

Für mich ist das Themengebiet dahingehend besonders, als dass mich die Aspekte Berechenbarkeit und Komplexität nicht erst seit meinem Informatikstudium begleiten. Bereits in meinem vorangegangenen Studium der Mathematik wählte ich die angewandte diskrete Mathematik, die insbesondere auch Graphentheorie umfasst, als Spezialgebiet und habe seither großes Interesse und Spaß an diesem Bereich.

Zugänglichkeit

Da die Inhalte der Unterrichtsreihe überwiegend abstrakter Natur sind, ist es umso wichtiger, den Schülern entsprechende Hilfestellung zu deren Durchdringung bereitzustellen. Die Automatenmodelle sollen beispielsweise sowohl in Tupelschreibweise als auch als Zustandsdiagramm dargestellt werden. Rollenspiele sowie Simulationen am Rechner erleichtern darüber hinaus spielerisch das Nachvollziehen der zentralen Ideen sowie der Wirkungsweisen. Bezüglich der Grammatiken können zusätzlich zu den Ableitungen Syntaxbäume verwendet werden. Zu guter Letzt helfen auch Vergleiche mit alltäglichen Phänomenen dabei, Gedächtnisstützen aufzubauen und das Verständnis für die Automaten und formalen Sprachen zu schärfen.

2.2.3 Methodische Analyse

Als Grundformen des didaktischen Handels sind in der behandelten Unterrichtsreihe viele Lehr- und Lernmethoden denkbar. Es bieten sich neben dem klassischen Frontalunterricht oder einem fragend-entwickelnden Unterricht auch Simulationen am Computer und Rollenspiele an. Da die Kursgröße mit zehn Schülern relativ klein ist, die Räume über ausreichend Platz verfügen und die Lernenden nicht nur keine Disziplinprobleme aufweisen, sondern ganz im Gegenteil über hohe soziale Kompetenzen verfügen, sind alle Sozialformen denkbar. Neben Einzel-, Partner- und Gruppenarbeit böten sich ebenfalls die Formen des kooperativen Lernens oder, wie bereits erwähnt, Rollenspiele an. Wird Einzelarbeit angestrebt, gilt es mit Blick auf die Lerngruppe darauf zu achten, die Sozialform explizit hervorzuheben. Andernfalls könnte es leicht passieren, dass die Schüler einander gewohnheitsmäßig helfen.

Da die zu behandelnden Themen eher theoretischer Natur sind und sich kein abschließendes Lernprodukt erarbeiten lässt, kommt eine Projektarbeit an dieser Stelle nicht infrage. Auch Schülervorträge oder Gruppenpuzzle sind ungeeignet, weil die Lerninhalte keine überschaulichen Unterthemen umfassen, die sich vom Anspruch her ähneln würden.

Die wesentlichen Herausforderungen der Unterrichtsreihe bestehen in ihrer inhaltlichen Abstraktheit, der Vielzahl an neuen Fachbegriffen und den dazugehörigen symbolischen Schreibweisen. Diesbezüglich bietet es sich an, mit vielfältigen Beispielen und Analogien zu arbeiten; wobei es stets die Grenzen dieser Exempel aufzuzeigen gilt.

Quizze können helfen, die Begriffe und Schreibweisen spielerisch zu wiederholen und zu festigen. Die freie Simulationssoftware *JFLAP* ist ein weiteres motivierendes Werkzeug, das

unterstützend eingesetzt werden kann, um anspruchsvolle Inhalte wie das Pumping-Lemma nach dem Prinzip des entdeckenden Lernens zu beleuchten.

Gerade aufgrund des insgesamt hohen inhaltlichen Anspruchs sollten die verschiedenen Lehrmethoden abwechslungsreich verwendet werden. Ein variierender Fokus auf die lerntheoretischen Ansätze Konstruktivismus, Kognitivismus und Behaviorismus kann dazu beitragen, alle Lerntypen anzusprechen, so dass der gesamte Kurs die entsprechenden Lernziele erreicht.

3 Unterrichtsentwurf einer Doppelstunde

3.1 Einordnung der Stunde in die Unterrichtseinheit

Die Doppelstunde, die ich im Folgenden ausführlich präsentieren möchte, bildet den Einstieg in das Themengebiet Sprachen und Automaten. In der folgenden Tabelle findet sich eine Übersicht zu der gesamten Unterrichtsreihe, wie sie geplant ist:

Datum	**Inhalt der Stunde**	**Lernziele** – Die Schüler können ...
06.01.20	Einführung deterministische endliche Automaten	• die Bestandteile eines DEAs benennen und erklären • DEAs in Tupelschreibweise und grafisch als Zustandsdiagramm darstellen und zwischen den Darstellungen wechseln • Automaten der Informatik von Automaten im Alltag abgrenzen
08.01.20	Übungen zu DEAs und der von ihnen akzeptierten Sprache	• die akzeptierte Sprache eines DEAs bestimmen • zu gegebener (regulären) Sprache einen DEA entwerfen
10.01.20	Vertiefende Übungen zu akzeptierten Sprachen	• ausgewählte Probleme (Teilbarkeit in binär und dezimal, Präfix, Suffix, Teilwort) in Sprachen umwandeln und zugehörige DEAs angeben
15.01.20	Nichtdeterministische endliche Automaten	• DEA und NEA voneinander abgrenzen • die akzeptierte Sprache eines NEAs bestimmen • zu gegebener (regulären) Sprache einen NEA entwerfen
17.01.20	Determinisierung und exponentieller Blowup	• NEAs in DEAs umwandeln und umgekehrt • den (möglichen) exponentiellen Blowup bei der Potenzmengenkonstruktion beschreiben

Datum	**Inhalt der Stunde**	**Lernziele** – Die Schüler können ...
20.01.20	Reguläre Grammatiken	• die durch eine reguläre Grammatik erzeugte Sprache bestimmen • zu regulären Sprachen eine reguläre Grammatik angeben, die diese erzeugt • zu einem DEA/NEA eine reguläre Grammatik angeben, die die vom Automaten akzeptierte Sprache erzeugt
22.01.20	Zusammenhang zwischen den regulären Grammatiken und den endlichen Automaten	• zu einem NEA eine reguläre Grammatik angeben, die die vom Automaten akzeptierte Sprache erzeugt • zu einer regulären Grammatik einen NEA angeben, dessen akzeptierte Sprache die von der Grammatik erzeugte Sprache ist
24.01.20	Kontextfreie Grammatiken und die Chomsky-Hierarchie	• die durch eine kontextfreie Grammatik erzeugte Sprache bestimmen • zu kontextfreien Sprachen eine kontextfreie Grammatik angeben, die diese erzeugt
29.01.20	Das Pumping Lemma	• die Grundidee des Beweises zum Lemma erläutern • für konkrete Sprachen (wie $\{a^n b^n : n \in \mathbb{N}\}$) beweisen, dass sie nicht regulär sind
31.01.20	Wiederholung für Klausur	
	Winterferien	
10.02.20	Klassenarbeit	
12.02.20	Modellierung mit endlichen Automaten und Automaten mit Ausgabe	• komplexe Systeme problemgerecht modellieren (Zustandsmodellierung) • Ausgaben von Mealy-Automaten bestimmen

Turing-Maschinen und Entscheidbarkeit sind nicht Teil der Unterrichtsreihe und wurden ans Ende des Semesters verlagert.

3.2 Didaktische Überlegung und Begründung

Der hohe Anspruch sowie die Komplexität und Abstraktheit der Unterrichtsinhalte verlangen danach, gerade den Einstieg in das Themengebiet mit höchster Sorgfalt anzugehen. Ich plane daher, die Kapazitäten der ersten Doppelstunde weitestgehend auszuschöpfen, um grundlegende Schreibweisen und Funktionsprinzipien im Hinblick auf Sprachen und DEAs verständlich einzuführen. Diese Fokussierung auf den Erwerb von Elementarkenntnissen und die Initiierung eines grundlegenden Verständnisses von Automaten soll Problemen vorbeugen, mit denen die Schüler sich andernfalls im weiteren Verlauf der Unterrichtsreihe konfrontiert sehen könnten. Die zeitliche Investition gleich zu Beginn soll den Übergang zu den anspruchsvolleren Folgeinhalten erleichtern und wiederholten Verzögerungen des Unterrichtsgeschehens aufgrund nachträglicher Erläuterungen vorbeugen.

Um nun einen möglichst nachvollziehbaren Zugang in die Thematik zu gestalten, sehe ich vor, auf Alltagserfahrungen der Lernenden zurückzugreifen. Einer fehlgeleiteten Gleichsetzung solcher alltäglichen Automaten mit jenen, die der Informatikunterricht bespricht, wird durch die Herausarbeitung maßgeblicher Unterschiede vorzubeugen sein. Insgesamt sollen die Schüler auf diese Weise effektive Gedächtnisstützen entwickeln, auf die sie im Zuge der gesamten Unterrichtsreihe fortlaufend zurückgreifen können. Zu diesem Zweck scheint mir der konstruktivistische Ansatz besonders geeignet.

Hinsichtlich der Wahl geeigneter Sozialformen möchte ich vermehrt die Gruppen- und Partnerarbeit einsetzen. Diese bieten sich nicht nur darum an, weil die Schüler bereits mit ihnen vertraut sind und aufgrund ihrer sozialen Kompetenzen erfolgreich in ihnen lernen können. Durch diese Art der Kommunikationsgestaltung ist es dem Einzelnen darüber hinaus möglich, vom Erfahrungsschatz des gesamten Kurses zu profitieren. Ich rechne zudem damit, dass die Schüler sich bei Verständnisfragen ggf. gegenseitig unterstützen.

Um das beschriebene Lernverhalten zu erhalten, eignen sich die Methoden der Gruppendiskussion, Tandem-Aufgaben sowie die Ich-Du-Wir-Methode. Mit der Zielsetzung, einen ansprechenden und abwechslungsreichen Unterricht zu gestalten, sollen alle drei Methoden in der Doppelstunde zum Einsatz kommen. Darüber hinaus soll mittels eines Rollenspiels, das es dem Kurs ermöglicht, erste Erfahrungen mit dem Konzept eines endlichen Automaten zu sammeln, ein besonders motivierender Einstieg realisiert werden.

Den Übergang zum abstrakten, theoretischen Modell und zur symbolischen Schreibweise möchte ich über den Frontalunterricht, vorwiegend durch den Lehrervortrag, gestalten. So

kann ich sicherstellen, dass die neuen Begriffe sauber erarbeitet und formal korrekt definiert werden. Darüber hinaus kann ich so auf unmittelbare Rückfragen eingehen und diese im Klassenverbund klären. Schließlich haben die Schüler hinsichtlich der theoretischen Aspekte kaum Vorkenntnisse und weisen demzufolge einen vergleichbar niedrigen Wissensstand auf.

Dieser geringe Wissensstand auf dem weiten Themenfeld Automaten und Sprachen stellt für mich eine der größten Herausforderungen in meiner Unterrichtsplanung dar. Da sich nur schwer vorhersagen lässt, wie zügig es dem Kurs gelingt, entsprechende Aufgaben zu bearbeiten, sprich wie lang die Dauer von Erarbeitungs-, Vertiefungs- und Sicherungsphasen sein wird, sehe ich die Frage zur zeitlichen Koordination der Doppelstunde kritisch. Hinzukommt, dass der Unterricht am ersten Schultag im Jahr 2020 stattfindet. Damit bricht auch das letzte Semester der Abiturienten an. Es besteht die Möglichkeit, dass dieser Umstand für eine gewisse Aufregung unter den Schülern sorgt. Ich plane auch darum ausreichend Zeit für die einzelnen Unterrichtsphasen ein. Für den Fall, dass die Lernenden ihre Arbeitsaufträge frühzeitig erfüllen, bereite ich weiterführende Fragen vor. Alternativ kann die Zeitersparnis auch im Sinne eines allgemeinen Ausblicks auf die folgenden Stunden genutzt werden, so dass der Kurs die Relevanz der Thematik gut überblicken und die neu gelernten Begriffe grob einordnen kann.

3.3 Unterrichtsentwurf einer Doppelstunde

Diese Überlegungen bringen mich nun zu den folgenden Stationen für die Doppelstunde; Ich starte mit einem Rollenspiel in Form einer Schatzsuche, um mit einem Einstieg mittendrin den Zustandsverlauf eines DEAs während der Bearbeitung einer Eingabe zu simulieren. Sieben Schüler fungieren als Wegweiser, die auf sieben Inseln verteilt sind. Die restlichen (hier drei) sind Schatzsucher und starten auf der Pirateninsel. Ihr Ziel soll es sein, einen Weg zur Schatzinsel zu finden und dabei die Schatzkarte zu vervollständigen. Auf jeder Insel können sie sich für Schiff A oder Schiff B entscheiden. Der Wegweiser teilt ihnen anschließend das Ziel des gewählten Schiffes mit. Der Schatzsucher notiert die Information auf seiner Schatzkarte und begibt sich zu dem genannten Ziel.

Dieses Rollenspiel wurde mir während eines Informatikseminars an der Universität vorgestellt und hat mich aufgrund seiner anschaulichen und spielerischen Methodik sofort begeistert, so dass ich entschied, es im Kontext der Doppelstunde zu verwenden. Das Originalmaterial zur Schatzsuche ist auf der Webseite *csunplugged.org* der University of Canterbury zu finden.[3] Da die Unterlagen lediglich in englischer Sprache vorlagen, habe ich

sie übersetzt und darüber hinaus andere nötige Anpassungen vorgenommen. Beispielsweise änderte ich manche Inselnamen, um sicherzustellen, dass die Anfangsbuchstaben der Inseln sich nicht doppeln, so dass sie sich zur Bezeichnung der entsprechenden Zustände im Automaten eignen. Dieses Vorgehen halte ich für wichtig, um den Lernenden die Abstraktion von der Schatzkarte hin zum Zustandsdiagramm des DEAs zu erleichtern. Ich ergänzte die Karte außerdem um Routen, die von der Schatzinsel wegführen. Erst so wurde die Vervollständigung der Karte überhaupt möglich bzw. so wurde das Endprodukt einem DEA in genügendem Maße ähnlich.

Alternativ hätte ich die Schatzsuche auch am Computer über eine digitale Schatzkarte (z.B. über verlinkte Webseiten oder Verweise auf PowerPoint-Folien) vorbereiten können, habe mich letztlich aber für die analoge Variante entschieden und werde die Schüler an Kopien der stummen Schatzkarte arbeiten lassen. Meine Wahl begründet sich in der Überlegung, die Lernenden mittels einer motorischen Ansprache besser aktivieren zu können. Statt der statischen Arbeit am PC bewegen sich die Jugendlichen und interagieren persönlich miteinander. Ziel ist es, sie auf diese Weise unangestrengt an den Themenkomplex Automaten und Sprachen heranzuführen und nachhaltig in den Lernprozess zu integrieren.

Im Anschluss an das Rollenspiel folgt eine kurze Sicherungsphase. In einer Gruppendiskussion sollen die Schüler wesentliche Elemente jener Schatzkarte identifizieren, wie sie zur Anwendung kam. Die erarbeiteten Merkmale nutze ich dann, um im Lehrervortrag die Definition eines DEAs herzuleiten.

Um den Bezug zur Informatik nicht zu verlieren und ihn für die Klasse deutlich zu machen, veranschauliche ich kurz die Bedeutung von Automaten, indem ich die Wortherkunft des Begriffes „Informatik“ erläutere. Daraufhin ist eine Übungsphase vorgesehen, in der die Schüler ihre bisherigen Kenntnisse vertiefen sollen. Zunächst in Form von Tandem-Aufgaben, die von ihnen verlangen, zwischen den verschiedenen Darstellungsformen eines DEAs zu wechseln. Eine weitere Aufgabe sieht mittels der Ich-Du-Wir-Methode vor, dass die Lernenden die neuen Begriffe mit ihren Vorerfahrungen verknüpfen bzw. sie voneinander abgrenzen.

Zum Stundenende soll ein Ausblick auf die Anwendungsfelder von Automaten in der Informatik noch einmal die Relevanz der theoretischen Objekte hervorheben und gleichzeitig neugierig auf den Folgeunterricht machen.

3.4 Geplanter Stundenverlauf

Die Doppelstunde zur Einführung deterministischer endlicher Automaten ist für den 06. Januar 2020 geplant. Wie gewohnt können die kompletten 90 Minuten im Informatikraum S15 stattfinden. Ziel der Stunde ist es, den Schülern mittels der Vermittlung grundlegender Einsichten die Erweiterung ihrer Modellierungskompetenzen zu ermöglichen. Langfristiges Kompetenzziel ist das Verwenden von Automaten für eine zustandsorientierte Modellierung. Als Lernziele können die folgenden Punkte festgehalten werden:
Die S können:

L1. die Bestandteile eines DEAs benennen und erklären,

L2. DEAs in Tupelschreibweise und grafisch als Zustandsdiagramm darstellen und zwischen den Darstellungen wechseln,

L3. Automaten der Informatik von Automaten im Alltag abgrenzen.

Tabelle 2 zeigt den geplanten Unterrichtsverlauf. Das Unterrichtsmaterial ist in Anhang B zu finden.

3.5 Beschreibung der Unterrichtsdurchführung

Nach einer kurzen Begrüßung habe ich wie geplant mit dem Rollenspiel „Schatzsuche“ begonnen. Die Spielregeln wurden über den Beamer projiziert und von mir mündlich vorgetragen, während ich zeitgleich die Wegweiserkarten (siehe Anhang B.3) entsprechend der Inselpositionen auf der Schatzkarte im Klassenraum verteilte. Auf die Frage, wer Schatzsucher sein möchte, meldeten sich einige Schüler. Ich wählte drei von ihnen aus, darunter zwei Schüler, die sich im regulären Unterrichtsgeschehen eher zurückhalten. Meine Absicht war es, auf diese Weise ihre Motivation zu würdigen.

Die freiwilligen Sucher wurden anschließend zur Tür gebeten und erhielten eine leere Karte. Die restlichen Schüler verteilten sich auf die „Inseln“. Um ihnen Orientierung zu bieten, wurde die Schatzkarte über den Beamer an die Wand projiziert. Kopien der leeren Karte legte ich auf ihre gewöhnlichen Sitzplätze. Nacheinander und mit zeitlichem Abstand machte sich nun jeder der drei Sucher auf den Weg zur Startinsel – mit zeitlichem Abstand, damit die Schüler nicht hörten, welche Routen ihre Mitschüler wählten. Nach den geplanten 15 Minuten hatten alle Sucher ihre Karte vervollständigen können.

Phase Zeit	**Unterrichtsinhalte und -verhalten**	**Aktions- und Sozialform**	**Material, Medien**
Begrüßung 5 min	(erster Tag im neuen Jahr) L platziert die Routenkarten im Raum.		Routenkarten
Einstieg 5 min	L stellt die Schatzsuche vor: 7 Schüler sind Inseln, der Rest Piraten auf Schatzsuche Aufgaben für die Piraten: a) Finde einen Weg zur Schatzinsel. Notiere auf der Schatzkarte, wie man von Insel zu Insel kommt. b) Vervollständige die Schatzkarte.	Lehrervortrag, Frontalunterricht	Folie
Erarbeitung 15 min	S führen die Schatzsuche durch und vervollständigen die Schatzkarte.	Einzelarbeit, Unterrichtsspiel	Folie, Routenkarten Schatzkarten
Erarbeitung 6 min	Was sind die wesentlichen Bestandteile einer Schatzkarte? -> Inseln, Schiffe, Schiffsrouten, Startinsel, Ziel(e) Wie können wir Start- und Zielinsel markieren?	Frontalunterricht, Unterrichts-gespräch	Tafel
Sicherung 7 min	Ein S zeichnet seine Schatzkarte an die Tafel. Die anderen S ergänzen ihre Karten. Welche Wege führen zur Schatzinsel? Welche nicht?	Schülervortrag, Gruppendiskussi on	Tafel
Überleitung 7 min	Was hat diese Schatzsuche mit Informatik zu tun? Was bedeutet das Wort Informatik? -> "automatische Informationsverarbeitung" bzw. Information+Automatik	Lehrervortrag, Frontalunterricht	Folie
Erarbeitung 10 min	Definition: deterministischer endlicher Automat (55min) S schreiben Defintion ab.	Lehrervortrag, Frontalunterricht	Tafel
Vertiefen 20 min	S bearbeiten Aufgabe 1 im Tandem und Aufgabe 2 in Modus Ich-Du-Wir.	Tandem, Ich-Du-Wir	Arbeitsblatt
Vergleichen 10 min	Im Plenum werden Schwierigkeiten bei Aufgabe 1 besprochen. S präsentieren ihre Lösungen zu Aufgabe 2. *Welche Richtung ist schwieriger?* *Tupel -> Grafik, Grafik -> Tupel*	Frontalunterricht, Unterrichts-gespräch, Schülervortrag	Tafel, Dokumenten-kamera
Ausblick 5 min	Wofür werden Automaten in der Informatik benötigt? *hier ist vermutlich schon Schluss*	Lehrervortrag, Frontalunterricht	Folie
Erarbeitung 10 min	Definition: Wort, leeres Wort, Sprache, akzeptierte Sprache, reguläre Sprache	Lehrervortrag, Frontalunterricht	Folie
	weitere Fragen: • Gibt es einen DEA, der kein Wort akzeptiert? • Gibt es einen DEA, der alle Wörter über einem gegebenen Alphabet akzeptiert? • Gibt es Sprachen, die nicht von einem DEA akzeptiert werden?		

Tabelle 2: Unterrichtsentwurf zur Einführung deterministischer endlicher Automaten

Nachdem nun das Rollenspiel abgeschlossen war, wurden im Unterrichtsgespräch die wesentlichen Komponenten der Schatzkarte gesammelt und gemäß dem geplanten Tafelbild (siehe Anhang B.1) mit verschiedenen Farben am Whiteboard notiert. Zusätzlich zeichnete ich die Inseln als Zustände an, wobei ich diese mit den entsprechenden Anfangsbuchstaben kennzeichnete. Einer der Schatzsucher wurde von mir aufgefordert, die Zustandsübergänge entsprechend seiner Karte einzutragen. Die übrigen Schüler waren angehalten ihre Aufzeichnungen zu korrigieren bzw. das Diagramm auf ihre noch leere Schatzkarte zu übertragen. Daraufhin konfrontierte ich den Kurs mit der abschließenden Frage, welche Wege (im Sinne von A-B-Folgen) von der Startinsel zum Schatz führen und welche nicht. Auf beide Ansätze erhielt ich korrekte Antworten.

Die Arbeit an der Schatzkarte war hiermit endgültig abgeschlossen und ich wandte mich der Frage zu, was nun diese Schatzsuche mit *Informatik* zu tun hätte. Über die Herkunft des Begriffes Informatik als Kofferwort aus Information und Automatik leitete ich zur Definition des deterministischen endlichen Automaten über und ergänzte dementsprechend das Tafelbild. Als erstes Beispiel verwies ich auf die Abbildung des Zustandsdiagramms am Whiteboard, das die Daten der Schatzsuche hervorgebracht hatten. Dabei setzte ich stets die Bestandteile des Automaten mit denen der Karte in Bezug.

Es folgte die Ausgabe des Arbeitsblattes, welches in Anhang B.4 zu finden ist. Ich stellte die Aufgaben und Methoden zur Bearbeitung kurz vor und ließ die Schüler arbeiten. Dabei ging ich regelmäßig durch die Reihen, um mir einen Überblick über den Fortschritt zu verschaffen, ggf. zu helfen und um sicherzustellen, dass die Tandem-Aufgaben zunächst in Einzelarbeit gelöst wurden. Nach der Bearbeitungszeit trugen wir im Unterrichtsgespräch lediglich die Lösungen zu Aufgabe 2 zusammen, da die Lösungen der Tandem-Aufgaben über den Tandem-Partner gegeben waren und es bezüglich Aufgabe 1 auch keinerlei offene Fragen gab. Ein Vergleichen erübrigte sich demzufolge. Während der Gruppendiskussion provozierte ich die Schüler mittels gezielter Nachfragen immer wieder zur Präzisierung ihrer Antworten und bestand auf die Verwendung der neu gelernten Fachbegriffe.

Im Anschluss hielt ich einen kurzen Lehrervortrag, mit dem Ziel einen kleinen Überblick zur Nutzung von Automaten in der Informatik zu geben. Da noch einige Minuten Unterrichtszeit zur Verfügung standen, entschied ich mich in Vorbereitung auf die Folgestunden die Begriffe *Alphabet*, *Wort*, *leeres Wort*, *Sprache* und *akzeptierte Sprache* zu definieren. An dieser Stelle endete der Unterricht.

3.6 Reflexion der Stunde

Die Doppelstunde zum Einstieg in den Themenkomplex Automaten und Sprachen kann insgesamt als Erfolg gewertet werden. Anhand der korrekt vorgebrachten Schülerlösungen sowie der Sicherungsphasen ließ sich feststellen, dass alle Lernziele erreicht wurden. Dank der unterstützenden Beamerfolien wussten die Schüler zu jedem Zeitpunkt, was ich von ihnen erwartete. Es kam ihrerseits daher zu keinen Leerläufen oder Unsicherheiten. Auch von dem Tafelbild, dessen Übersichtlichkeit u.a. durch den gezielten Einsatz von Farben erreicht wurde, scheinen die Lernenden profitiert zu haben; zum Beispiel, um von der Schatzkarte auf die Definition eines DEAs zu abstrahieren. In den Unterrichtsdiskussionen konnte ich zudem bemerken, wie der Kurs sich die Schatzkarte als sinnvolle Gedankenstütze zu eigen machte; etwa um theoretische Überlegungen auszuformulieren. Mein ursprünglicher Plan ging also auch hier auf. Darüber hinaus hatte ich den Eindruck, dass die Schüler sich mit Freude am Unterricht beteiligten. Sie nahmen das Rollenspiel sehr gut an, waren motiviert und im Anschluss offen auch für die abstrakten Erklärungen. Meine Befürchtung, die Jugendlichen könnten die Schatzsuche im schlimmsten Fall für zu albern oder gar kindlich empfinden, bestätigte sich nicht.

Zukünftig allerdings würde auch ich als Lehrer an dem Rollenspiel partizipieren und die Aufgabe eines Wegweisers übernehmen. Auf diese Weise können noch mehr Schüler aktiv an der Schatzsuche teilnehmen.

Da am Ende der Doppelstunde noch einige Minuten Zeit blieben, führte ich neue Definitionen bis hin zur akzeptierten Sprache ein. Hier aber hatte ich den Aufwand unterschätzt, so dass diese Unterrichtsphase relativ gehetzt verlief. Zwar war mittels der ergänzenden Definitionen ein rundes Ende hinsichtlich der in der Schatzsuche gültigen Wege gelungen. Ein Ausstieg über die Relevanz der Automaten in der Informatik hätte die Stunde jedoch gleichermaßen sinnvoll schließen können. Hier mangelt es mir noch an dem nötigen Fingerspitzengefühl, das Unterrichtstempo an ausgewählten Stellen situationsbedingt zu erhöhen bzw. zu verlangsamen.

Die Definition eines Alphabets als geordnete, endliche Menge hatte ich aus dem Skript meiner Hochschulvorlesung übernommen. Dabei war mir nicht bewusst, dass es auch Definitionen gibt, in denen auf die Ordnung verzichtet wird. Da für die Schulzwecke eine Ordnung irrelevant ist, werde ich zukünftig das Alphabet ohne Ordnung definieren.

Ungeachtet dessen war mein Mentor sehr zufrieden mit der Stunde. Er lobte mein sicheres

Auftreten und die authentische Interaktion mit den Schülern. Auch das Rollenspiel wusste ihn zu begeistern. Seine ursprüngliche Planung hatte vorgesehen, die Unterrichtsreihe nur Automaten und Sprachen mit den Mealy-Automaten zu beginnen, um sofort interessante, motivierende Beispiele zur Hand zu haben. Meine andere Herangehensweise allerdings vermochte ihn derart zu überzeugen, dass er angab, die Thematik zukünftig auf dieselbe Weise einführen zu wollen.

Insgesamt bin ich mit der Doppelstunde im Informatik-Leistungskurs Klasse 12 zufrieden.

4 Kollegiale Hospitation

Am 05. Dezember 2019 hatte ich Gelegenheit, bei einem Kommilitonen zu hospitieren. Es handelte sich um einen Block von 90 Minuten im Wahlpflichtunterricht Informatik Klasse 10. Insgesamt waren 15 Schüler anwesend. Ein Schüler fehlte krankheitsbedingt. Aktuelles Thema war das objektorientierte Programmieren mittels *BlueJ*. Die Schüler besaßen bereits Grundkenntnisse im Programmieren mit Java. Das Stundenziel bestand darin, Konstruktoren sowie getter- und setter-Methoden einzuführen. Hierzu wurde als Programmieraufgabe ein naiver Ticketautomat zum Kaufen und Drucken von Fahrkarten herangezogen.

4.1 Vorüberlegungen zur Auswertung

Bekannterweise lässt sich eine Unterrichtsstunde unter zahlreichen Aspekte betrachten. Für eine Hospitation ist es daher unabdingbar, sich bereits im Vorfeld für einen spezifischen Gesichtspunkt zu entscheiden, der infolge das Objekt näherer Beobachtung und eingehender Reflexion wird. Auf Wunsch meines Kommilitonen wählte ich den Aspekt der inhaltlichen Klarheit zum theoretischen Gegenstand meiner Hospitation.

In Anlehnung an den entsprechenden Hospitationsbogen von Meyer (siehe [7]) legte ich für diese Beobachtungsaufgabe besonderes Augenmerk auf die folgenden Punkte und fertigte hierzu im Verlauf des Unterrichts Notizen an:

- Ist ein roter Faden erkennbar?
- Lässt sich die Lehrkraft leicht ablenken oder schweift vom Thema ab?
- Spricht die Lehrkraft klar und verständlich?
- Ist das Tafelbild klar und verständlich?
- Sind die Arbeitsaufträge verständlich formuliert?
- Werden Verständnisfragen umfänglich geklärt?

Der konkrete Hospitationsbogen fand dabei jedoch keine Verwendung.

4.2 Reflexion

In ihrer Gesamtheit verlief die von mir hospitierte Unterrichtsstunde, gerade auch im Hinblick auf die inhaltliche Klarheit, gut. Ein roter Faden ließ sich bereits früh im Unterrichtsgeschehen deutlich erkennen. Der Ablauf des 90-Minuten-Blocks war klar strukturiert und

wurde den Schülern zu Beginn der Stunde kommuniziert. Während des Unterrichts war mein Kommilitone auf die Inhalte fokussiert und ließ sich nicht ablenken. Auch gelang es ihm, den im Vorfeld anberaumten Zeitplan weitestgehend einzuhalten. Seine Aussprache war stets angemessen laut und deutlich. Sein Auftreten wirkte prinzipiell sicher.

Meine insgesamt positive Einschätzung würde ich allein dadurch einschränken, dass häufig Füllwörter und Verzögerungslaute genutzt wurden, welche ich als leicht störend empfand. Für mich selbst ziehe ich aus dieser Beobachtung, selbst mehr auf meine Sprache zu achten, um in meiner zukünftigen Tätigkeit als Lehrkraft derartige Wörter und Laute auf ein Minimum zu reduzieren.

Anstatt eines Tafelbildes wurden Folien und Arbeitsblätter verwendet. Auch diese waren übersichtlich und klar strukturiert. Ansprechende Bilder lockerten das Design auf. Von Seiten der Schülerschaft kam es dahingehend zu keinerlei Verständnisfragen.

Die Arbeitsaufträge waren durchaus verständlich formuliert, lediglich eine Beschreibung zur Funktionsweise des Automaten fehlte. Die Kommentare im vorgegebenen Code genügten hier nicht. Einer Reihe von Schülern und auch mir war beispielsweise nicht klar, was konkret der Befehl „Ticket drucken“ bewirken sollte. Die Unklarheiten wurden in Einzelgesprächen geklärt. Besser wäre es gewesen, nach Feststellung des Problems die Arbeitsphase kurz zu unterbrechen, um die vorgesehene Funktionsweise des fraglichen Befehls für die gesamte Klasse zu erläutern. In einem späteren Gespräch erklärte mein Kommilitone sein Vorgehen damit, dass es auf ihn den Eindruck gehabt hätte, als hätte es nur vereinzelte Unsicherheiten gegeben, so dass er den laufenden Arbeitsprozess nicht hatte unterbrechen wollen. Hieraus nehme ich mit, dass es derlei Verständnisschwierigkeiten stets genau zu beobachten gilt; sprich: wie viele Schüler haben Verständnisprobleme und wie groß sind diese? Ein oberflächliches Scannen reicht hier nicht aus. Vielmehr gilt es, ein gewisses Gespür und Fingerspitzengefühl zu entwickeln, um eine solche Situation korrekt einschätzen und die angemessene Entscheidung zwischen der Klärung im Einzel- oder Gruppengespräch treffen zu können. Hierfür braucht es Erfahrung, so dass ich kommende Unterrichtsgelegenheiten nutzen will, um meine Fähigkeit zur Einschätzung von Quantität und Qualität etwaiger Verständnisprobleme auszubauen und zu trainieren.

Ergänzend zu den obigen Gesichtspunkten habe ich mir noch weitere Eindrücke notiert, wenn mir Aspekte des Lehr- und Lernverhaltens besonders aufgefallen sind. So herrschte durchweg eine positive Arbeitsatmosphäre, begünstigt durch die offene Art und Ausstrahlung meines Kommilitonen. Es gelang ihm, den Spaß, der er augenscheinlich selber am Un-

terrichten hatte, auf die Schüler zu übertragen. Die Klasse war motiviert, sich einzubringen. Die Schüler gaben sich untereinander konstruktive Kritik bzw. korrigierten sich gegenseitig in wertschätzender Weise. Auch ich selbst strebe in meiner Rolle als Lehrkraft eine solche Atmosphäre an und dieses Unterrichtsbeispiel hat mir gezeigt, mit dieser Zielvorstellung auf dem richtigen Weg zu sein.

Im Folgenden einige Worte zum Verlauf des Unterrichts:
Zum Einstieg erfolgte die Aufforderung an die Schüler, den Stoff der vorherigen Unterrichtsstunde zu rekapitulieren. Im Anschluss wurden mittels eines Lehrervortrag die Konstruktoren eingeführt. Den Unterricht auf diese Weise zu eröffnen, empfand ich als wenig motivierend. Dennoch gelang der Einstieg und wurde weder durch Störungen und Unaufmerksamkeit sabotiert. Nicht zuletzt mache ich hierfür das gute Schüler-Lehrer-Verhältnis verantwortlich. Besser wäre es dennoch gewesen, die Schüler den Unterrichtsgegenstand selbst entdecken zu lassen; sprich wie sich der Code eines Konstruktors von den bereits bekannten Methoden unterscheidet. Die Gemeinsamkeiten und Unterschiede hätte man anschließend im Gruppengespräch diskutieren können.

Auch die Gesprächsführung betreffend, konnte ich noch einige Schwächen ausmachen. So stellte mein Kommilitone mehrfach Fragen, die von den Schülern im ersten Versuch nicht erschöpfend beantwortet wurden. Hierauf entgegnete er eher unkonkret: „Noch nicht so ganz. Wer kann es noch anders formulieren?“ Ein zweiter Antwortversuch von Seiten der Schüler behob dann zwar die Mängel des ersten, blieb jedoch weiterhin unvollständig. Dennoch wurde diese zwar bessere, aber noch immer unzureichende Antwort akzeptiert. Auf diese Weise blieb mehrfach unklar, wie nun die korrekte, erwartete Antwort gelautet hätte: Inwiefern war die erste Meldung richtig, was fehlte? Zudem wurde es vernachlässigt, die erste Antwort entsprechend zu würdigen. Ich könnte mir vorstellen, dass sich ein solches Lehrerverhalten auf Dauer negativ auf die mündliche Beteiligung der Klasse auswirkt.

Darüber hinaus gab es kleine fachliche Ungenauigkeiten seitens der Lehrkraft. Als etwa die Zugriffsmethoden *get* und *set* besprochen wurden, lag der Fokus bezüglich der Unterscheidung der Zugriffsmodifikatoren *private* und *public* stets auf dem Aspekt der Sichtbarkeit im Sinne der Geheimhaltung. Als Beispiel wurden Passwörter genannt, die allerdings der Thematik nicht vollkommen gerecht werden. Neben der Möglichkeit zum Auslesen von Variablenwerten ist auch das Verändern derselben relevant. Beides soll gegebenenfalls über die Zugriffsmethoden ermöglicht, gesteuert und kontrolliert werden. So wie das Beispiel besprochen wurde, lag der Fokus aber lediglich auf dem Auslesen.

Mir ist darüber hinaus aufgefallen, dass sich die Schüler kaum Notizen gemacht haben und auch selten ihre schriftlichen Lösungen bei oder nach der Sicherungsphase angepasst haben. Die Möglichkeit der unabhängigen Hospitation hat mir hierfür die Augen geöffnet. Denn auch ich selbst habe es bisher vernachlässigt, die selbständige schriftliche Mitarbeit der Schüler im Blick zu behalten und nehme mir daher vor, das Unterrichtsgeschehen diesbezüglich genauer zu beobachten.

Als letzten Punkt möchte ich noch eine Situation beschreiben, die mir einmal mehr verdeutlicht hat, dass man im Schulalltag immer wieder mit unvorhergesehenen Gegebenheiten konfrontiert wird, und man sich dementsprechend eine gewisse Flexibilität aneignen muss, um erfolgreich mit ihnen umgehen können muss. Im Falle des von mir hospitierten Unterrichts war ein neues Smartboard im Informatikraum installiert worden, welches sich in seiner Bedienung maßgeblich von vergleichbaren Geräten der Schule unterschied. Ich konnte die dadurch ausgelöste Verunsicherung bei meinem Kommilitonen spüren und bot, da der betreuende Lehrer noch nicht eingetroffen war, meine Hilfe an. Gemeinsam gelang es uns, die richtige Quelle einzustellen, so dass der Unterricht pünktlich starten konnte. Aus diesem Erlebnis habe ich gelernt, dass es trotz des Zeitdrucks, den die Unterrichtsplanung mit sich bringt, nicht angeraten ist, übereilt und hektisch zu agieren. Stattdessen empfiehlt es sich, der unvorhergesehenen Situationen ruhig begegnen und ggf. einen Kollegen/ evtl. auch einen Schüler um Hilfe zu bitten.

5 Reflexion des Praktikums

Das Praxissemester am Leibniz-Gymnasium war nicht nur eine Erfahrung, die mir sehr viel Spaß gebracht hat, sie bot mir darüber hinaus die Möglichkeit, mich auf meinem Weg zum Lehrerberuf weiterzuentwickeln. Ich habe mich selbst ausprobieren, vom Kollegium lernen und neue Ideen und Methoden erproben können. Gerade auch die angenehme Arbeitsatmosphäre an der Schule hat dazu beigetragen; maßgeblich begünstigt durch das nette und hilfsbereite Lehrpersonal. Ich konnte mich bei Unsicherheiten jederzeit mit meinen Mentoren und anderen Lehrkräften austauschen. Die fachlichen wie didaktischen Gespräche verliefen stets auf Augenhöhe. Ich habe mich willkommen gefühlt, mehr als Lehrkraft denn als Praktikant. Ich möchte es daher zu einer persönlichen Priorität machen, meiner späteren Lehrtätigkeit als Teil eines aufgeschlossenen und engagierten Kollegiums nachgehen zu können.

Die Zeit am Leibniz-Gymnasium konnte mich in einigen meiner Ansichten bestärken. So habe ich beispielsweise erleben können, dass ich in der Lage bin, mich vor einer Klasse durchzusetzen. Sowohl mein Mentor als auch andere Lehrkräfte lobten mein sicheres Auftreten, das gewiss auch auf meine mehrjährige Lehrerfahrung in Mathematik- und Informatik-AGs zurückzuführen ist. Es hat mich gefreut, dass ich scheinbar in der Lage war, die dort gewonnenen Erkenntnisse auch auf den regulären Unterricht zu übertragen.

Durch meine Zeit an der Schule fand ich mich zudem darin bestätigt, dass ich es vorziehe, in höheren Klassen zu unterrichten. Ich brenne für meine Fächer und habe Freude daran, Wissen und Begeisterung weitergeben zu können. Den Fokus meiner Tätigkeit lege ich dementsprechend bevorzugt auf den fachlichen und weniger den pädagogischen Bereich. Daraus ergibt sich, dass ich, falls möglich, später in einem Gymnasium mit MINT-Profil unterrichten möchte. Freuen konnte ich mich zudem über positives Schülerfeedback. Dies erhielt ich sowohl im direkten Austausch mit den Jugendlichen als auch über Berichte meines Mentors. Ihm zufolge erkundigten sich die Schüler nach mir und äußerten Enttäuschung, wenn ich während meiner Präsenztage an der Universität nicht in der Schule zugegen war.

Im Sinne einer umfänglichen Reflexion möchte ich nun auf drei persönliche Ziele eingehen, die ich mir zu Beginn des Praxissemesters gesetzt hatte.
So wollte ich zum einen meinen Methodenkoffer ausbauen (1): Die Planung des Unterrichts stellt jede junge Lehrkraft vor ein Dilemma. Zwar möchte man zielsicher jene Methoden wählen, die der Vermittlung des anstehenden Unterrichtsstoffes gerecht werden, besitzt jedoch zu wenig Erfahrung, um dies kompetent einschätzen zu können. Aus dieser Zwickmühle

wollte ich ausbrechen, indem ich mich verpflichtete, mindestens drei neue Methoden zu erproben und ihre Nützlichkeit zu reflektieren. Dieses Ziel konnte ich erreichen. Ich nutzte Code-Puzzles, Tandem-Aufgaben, Kahoot-Quizze, ein Rollenspiel, Analogien, übte das Problemlösen mittels enaktiver Gruppenarbeit und förderte entdeckendes Lernen mittels Analysetools. Besonders überrascht hat mich, wie positiv die Schüler auf Quizze reagierten. Zwar hatte ich erwartet, dass die Schüler spielerisch aktiviert würden, nicht aber, dass diese Methode sich als derart motivierend präsentieren würde. Es wird sich noch zeigen müssen, ob diese Erfahrung sich in anderen Schulen/Klassen wiederholen lässt. Grundsätzlich möchte aber auch zukünftig Quizze in meinen Unterrichtsalltag integrieren, um den beschriebenen positiven Effekt auszunutzen.

Darüber hinaus hatte ich mir vorgenommen, eine Klausur- bzw. Testaufgabe im Fach Informatik zu entwerfen und anschließend zu bewerten (2). Aus organisatorischen Gründen ließ sich dieses Vorhaben nur in Teilen realisieren, so dass ich zwar keine Klausuraufgabe selbst entwerfen, in einem Informatik-Wahlpflichtkurs der Klasse 10 jedoch die Möglichkeit zum Korrigieren erhielt. Wie ich im Vorfeld bereits befürchtet hatte, war diese Tätigkeit herausfordernd und extrem zeitaufwendig. Auf diesem Gebiet muss ich mich deutlich verbessern. Trotz des Tatsache, dass ich mich mit meinem Mentor darüber verständigt hatte, auf welche Antworten es wie viele Punkte gibt, war die Korrektur mühsam. Im Detail war ich mir unsicher, wie einzelne Schülerantworten zu bewerten sind und habe mich dementsprechend strikt an die Ansagen meines Mentors gehalten. Dessen anschließendes Feedback lautete, dass ich sehr streng korrigiert hätte. Ich entnehme daraus, dass auch hier, wie so oft, Fingerspitzengefühl gefragt ist.

Mein letztes Ziel war es, die Unterrichtsvorbereitung effizienter zu gestalten (3). Bereits die Erfahrungen, die ich mit früheren AGs und Werkstätten gesammelt habe, haben mir gezeigt, dass ich im Durchschnitt viel Zeit für die Planung der Einheiten benötige. Der hohe Maßstab, den ich an mich selbst anlege, zwingt mich dazu, viele Materialien selbst bzw. neu zu gestalten. Auf diese Weise erhalte ich einerseits Lernmittel, mit denen ich rundum zufrieden bin, jedoch nur zum Preis eines unverhältnismäßig großen (Zeit-)Aufwands. Ich muss daher lernen, auch auf Fremdmaterial zurückzugreifen, selbst wenn mir dessen Inhalt und Gestaltung nicht vollends entspricht. Dieser Kompromiss ist mir im Praxissemester nur teilweise gelungen. Zwar habe ich mehr Routine in der Unterrichtsvorbereitung gewonnen und die Materialien anderer Lehrkräfte zur Grundlage meiner Unterrichtsplanung und -durchführung genutzt, dennoch halte ich mich weiterhin an Details auf und muss an meiner

Flexibilität arbeiten. Dies gilt auch für das Unterrichtsgeschehen. Es fällt mir noch schwer, auf Probleme, mögen sie von Schülern ausgehen oder technisch/organisatorischer Natur sein, spontan zu reagieren. Gerade auch bei Verständnisproblemen muss ich den Mut haben, von meiner eigenen Stundenplanung ggf. abzuweichen.

Eine weitere Herausforderung, die mir das Praxissemester nahegebracht hat, sind die zusätzlichen IT-Aufgaben, die auf mich als Informatiklehrer zukommen können. Auf diesem Gebiet habe ich nur wenig Kenntnisse und war schnell von den Erläuterungen meines Mentors überfordert. Sollte ich später für die IT-Betreuung zuständig sein, muss ich mir dieses zusätzliche Wissen erst aneignen.

Eine weitere Auffälligkeit, die der Schulalltag mir offenbart hat, waren die kurzen Pausen, die am Leibniz-Gymnasium zwischen den Blöcken liegen. Als Lehrkraft war man mit vielen organisatorischen Aufgaben beschäftigt, musste sich mit Kollegen abstimmen, eventuell zum Nachbarhaus gehen oder wollte auch nur etwas essen und trinken. Aufgrund des straffen Stundenplans blieb daher nur wenig Zeit, um mit den Schülern in Kontakt zu kommen und etwaige Probleme zu besprechen – etwas, dass ich gerne getan hätte.

Nichtsdestotrotz war und bin ich mit dem Verlauf meines Praxissemesters sehr zufrieden und sehe mich in meiner Berufswahl bestärkt.

6 Der Alternativentwurf

6.1 Problemanalyse

Am 13. September 2019 hielt ich den zweiten Teil einer Doppelstunde im Informatik-Wahlpflichtunterricht der Klasse 10 ab. In den ersten 45 Minuten besprach mein Mentor vertiefende Übungen zu `boolean`-Variablen, die im vorherigen Block eingeführt worden waren. Im Anschluss daran, war es meine Aufgabe, den Themenkomplex der Variablen um den Datentyp `int` zu erweitern. Konkret wurden die folgenden Lernziele formuliert:
Die Schüler können

L1: den Ablauf von einfachen Programmen, in denen `int`-Variablen genutzt werden (wie „fünf Körner fressen“), beschreiben und im Ablauf die Variablenwerte angeben,

L2: `int`-Variablen nutzen, um einfache Probleme (wie „eine Reihe von 6 Körnern legen“) zu lösen.

Im Fokus der Unterrichtsstunde stand die Förderung der Kompetenz, verschiedene Datentypen in unterschiedlichen Zusammenhängen sachgerecht verwenden zu können.

Im Folgenden möchte ich nun einen kurzen Überblick zum den Verlauf der Stunde geben. Der tabellarische Unterrichtsentwurf findet sich in Tabelle 3. Die entsprechenden Folien sind im Anhang C.3 beigefügt.

Um das Interesse der Schüler zu gewinnen, gestaltete ich den Einstieg mittels eines GIFs, das die Animation eines sich überfressenden Hamsters zeigte. Im Anschluss warf ich die Frage auf, wie es gelingen könne, das Tier exakt fünf Körner fressen zu lassen. Im Gruppengespräch wurden daraufhin erste Ideen gesammelt. Aus den geäußerten Überlegungen ging schnell hervor, dass der Hamster gezwungen wäre, sich die Anzahl der gefressenen Körner zu merken, und dass zu diesem Zweck `boolean`-Variablen nicht ausreichend wären. Aufbauend auf diese Motivation führte ich dann die `int`-Variablen ein. Dabei legte ich viel Wert darauf, sowohl Gemeinsamkeiten als auch Unterschiede der beiden Typen zu besprechen. Unterstützend nutzte ich Analogien. `Boolean`-Variablen waren im Vorfeld als Briefumschläge visualisiert worden, die einen Zettel mit dem Wert `true` oder `false` beinhalteten. An dieses Bild anknüpfend, verwendete ich zur Veranschaulichung der int-Variablen daher eine kleine Kiste, in der ein prismaförmiges Schild lag. Darauf vermerkt war der Wert der Variable. Auf diese Weise sollte verdeutlicht werden, dass die verschiedenen Variablentypen unterschiedliche Wertbereiche besitzen.

Phase Zeit	**Unterrichtsinhalte**	**Erwartetes Verhalten der Schüler**	**Aktions- und Sozialform**	**Material Medien**
Einstieg 2 min	L zeigt Animation als Motivation für die Stunde.	S hören zu und liefern erste Lösungsvorschläge.	FU, LV	Folie, Animation
Erarbeitung 7 min	L führt int-Variablen ein, vergleicht sie mit Variablen vom Typ boolean und nutzt die Analogie „Kiste“. L zeigt Kiste und tauscht den Wert aus. L stellt Vergleichsoperatoren vor.	S hören zu und stellen ggf. Fragen.	FU, LV	Folie, Brief und Kiste
Vertiefen 7 min	L stellt Aufgabe: sattFressen() vor. L hilft bei Fragen. L gibt einem S einen Klebestift zum Fixieren der Lösung.	S bringen die Code-Schnipsel in eine korrekte Reihenfolge.	EA, Puzzle	Folie, Code-Puzzle, Klebestift
Sicherung 10 min	L schaltet die Dokumenten-kamera an. L stellt Fragen bei Ungenauigkeiten.	Der ausgewählte S zeigt unter der Dokumenten-kamera seine Lösung und beschreibt, was die einzelnen Codezeilen bewirken. Andere S korrigieren ggf. [L1]	FU, SV	Dokumenten-kamera
Vertiefen 14 min	L stellt die Programmier-aufgabe zur Methode reiheLegen() vor.	S implementieren die Methode zusammen mit dem Banknachbarn.	Partnerarbeit	Schüler-PC, Folien
Sicherung 5 min	L stellt fragen bei Ungenauigkeiten.	Ein S stellt seine projizierte Lösung vor. [L2]	FU, SV	Beamer
	weitere Fragen: • Wie kann man sattFressen() verbessern? • Wie kann man reiheLegen() verbessern?	→ *nicht gegen Mauer laufen!* → *entweder weitere Bedingung im Schleifenkopf oder Variable auf Endwert setzen*		

Tabelle 3: Tabellarischer Unterrichtsentwurf der problembehafteten Stunde

Das neu erworbene Wissen sollte nun mittels eines Code-Puzzles vertieft werden. Hierfür hatte ich eine Methode `sattFressen()` geschrieben, bei der der Hamster (wenn möglich) fünf Körner in der Reihe vor ihm fraß. Ich druckte den Code und zerschnitt die einzelnen Zeilen horizontal in insgesamt zehn Teile. Dabei fasste ich die Zeilen so zusammen, dass ausschließlich eine Reihenfolge für die Lösung der Aufgabe sinnvoll war. Der in der Methode genutzte `++`-Operator wurde vorher nicht explizit eingeführt, sondern über einen Kommentar im Code erklärt.

Die Schüler arrangierten nun in Einzelarbeit die Papierstreifen. Nach kurzer Bearbeitungsphase gab ich einem ausgewählten Schüler, der die korrekte Lösung hatte, einen Klebestift zum Fixieren. Als er dann den Code unter der Dokumentenkamera vorstellte, bemerkte ich, dass der Schüler vor dem Kleben die Reihenfolge der Zeilen nochmals verändert hatte. So war der Code nicht nur inkorrekt, sondern erzeugte zudem eine Endlosschleife. Hier eine Gegenüberstellung der korrekten Lösung und der Schülerlösung:

Korrekte Lösung:

```
public void sattFressen(){
  int anzahlKoerner = 0;
  while(vornFrei()){
    vor();
    if(anzahlKoerner < 5){
      if(kornDa()){
        nimm();
        anzahlKoerner++;
      }
    }
  }
}
```

Schülerlösung:

```
public void sattFressen(){
  int anzahlKoerner = 0;
  while(vornFrei()){
    if(anzahlKoerner < 5){
      if(kornDa()){
        nimm();
        vor();
      }
    }
    anzahlKoerner++;
  }
}
```

Dieses Problem hatte zur Folge, dass die Sicherungsphase mehr Zeit in Anspruch nahm, als ich geplant hatte. Darüber hinaus wich ich an dieser Stelle kurz vom eigentlichen Unterrichtsplan ab, indem ich Endlosschleifen thematisierte. Die Stunde war zu diesem Zeitpunkt bereits so weit fortgeschritten, dass die Folgeaufgabe nicht mehr im Rahmen des Unterrichts beendet werden konnte. Ich gab diese daher als Hausaufgabe zur nächsten Informatikstunde auf.

Im Nachgang zu der beschriebenen Doppelstunde machte ich nun folgende Defizite in mei-

ner Unterrichtsvorbereitung aus: Ich hatte zwei Arten von Fehlerquellen nicht ausreichend antizipiert. Eine inhaltliche Fehlerquelle war das Erzeugen von Endlosschleifen, die aufgrund der Verschachtelung von Bedingungen nicht als solche von den Schülern identifiziert wurden. Eine methodische bestand darin, dass durch das Festkleben falsche Schülerergebnisse fixiert wurden (trotz meiner gezielten Auswahl eines Schülers).

6.2 Lösungsvorschläge

Im Folgenden möchte ich weiter auf die oben beschriebenen Probleme eingehen und vier mögliche Lösungsvorschläge vorstellen. Auch soll der Arbeitsauftrag über eine veränderte Grundannahme vereinfacht werden: Wir bestimmen, dass bis zur nächsten Mauer ausreichend Körner vor dem Hamster liegen. Der Code ist auf diese Weise leichter zu analysieren. Es kann zudem davon ausgegangen werden, dass Schüler weniger Endlosschleifen erzeugen. Die Lösungsvorschläge:

1. Die Schüler erhalten das Code-Puzzle in digitaler Form; mit der Aufforderung, die Zeilen in Partnerarbeit in die korrekte Reihenfolge zu bringen.

   Diese spielerische Methode ist ähnlich aktivierend wie das enaktive analoge Puzzle. Die Form der Partnerarbeit senkt zudem die Fehlerquote. Nachteilig ist, dass die Schüler kein Endprodukt erhalten.

2. Das digitale Code-Puzzle wird für die Klasse über den Lehrer-PC projiziert, sodass in der Gruppendiskussion Reihenfolgen bzw. einzelne Zeilenpositionen vorgeschlagen werden können.

   Dieses Vorgehen eignet sich dahingehend, dass eventuelle Fehlvorstellungen schnell und unter Einbeziehung der gesamten Klasse behoben werden können. Ein typischer Nachteil der genannten Methode besteht allerdings darin, dass sich einzelne Schüler der Gruppendiskussion und damit dem Denkprozess entziehen können.

3. Die Schüler erhalten die zufällig angeordneten Codezeilen in einem Textdokument und nutzen sie, um die korrekte Methode in *Greenfoot* zu implementieren.

   Vorteil dieser Methode ist, dass die Schüler sofort Feedback darüber erhalten können, ob ihr Ansatz das Problem löst. Zugleich aber untergräbt sie die Erforderlichkeit, den Code korrekt zu analysieren und zu verstehen, ehe die Schüler sich wohlüberlegt auf

eine spezifische Lösung festlegen. Die Vermittlung der Kompetenz, Aufgaben zielgerichtet algorithmisch lösen und Fehler im eigenen Programm selbstständig finden zu können, tritt hierbei folglich in den Hintergrund.

4. Die Schüler erhalten den Code jener Methode, mit der der Hamster genau fünf Körner frisst. Der Methodenname `macheWas` gibt jedoch keine Hinweise darauf, was in der Methode passiert. Arbeitsauftrag der Schüler ist es nun, zu beschreiben, was die einzelnen Zeilen bewirken, und einen geeigneten Namen für die Methode zu finden.

   Dieser Ansatz bietet einen diagnostischen Vorteil. Nur wer versteht, was die einzelnen Zeilen bewirken, kann eine entsprechende Abstraktion vollziehen und einen passenden Namen finden. So können Verständnisprobleme identifiziert werden. Diese Herangehensweise würde allerdings einen veränderten Unterrichtseinstiegs voraussetzen, welcher ähnlich ansprechend und aktivierend sein sollte, wie der von mir verwendete.

6.3 Der überarbeitete Entwurf

Ich möchte nun einen überarbeiteten Unterrichtsentwurf präsentieren und werde mich dabei an dem ersten der oben skizzierten Lösungsvorschläge orientieren. Grundlage dessen wird die ebenfalls oben beschriebene vereinfachte Aufgabenstellung sein. Wir gehen folglich davon aus, dass zwischen dem Hamster und der nächsten Mauer in dessen Blickrichtung ausreichend Körner vorhanden sind. Zudem wird das analoge Code-Puzzle durch ein digitales ersetzt. Dies gilt es gemeinsam mit dem Banknachbarn zu bearbeiten. In der Sicherungsphase wird ein Schüler bestimmt, der das Puzzle am Lehrer-PC vorführen soll und darüber hinaus beschreibt, welche Auswirkung die einzelnen Codezeilen im Programmablauf haben. Der Rest der Klasse soll die Erläuterungen des Vortragenden ggf. hinterfragen oder korrigieren. Es soll außerdem Raum für Verständnisfragen geben. Da das Puzzle in Partnerarbeit gelöst werden soll, wird die Sozialform für die nächste Aufgabe zur Einzelarbeit abgeändert. Hiermit soll jedem Schüler die Möglichkeit gegeben werden, sich eigenständig mit der Problemstellung zu befassen um zeitgleich auch Lernfortschritte oder -defizite festzustellen.

Das digitale Puzzle ist über http://learningapps.org/watch?v=pcynfp3na20 abrufbar. Tabelle 4 zeigt den tabellarischen Unterrichtsentwurf und die entsprechenden Folien befinden sich im Anhang C.4.

Zum Abschluss dieses Gliederungspunktes möchte ich nun kurz die Gründe erläutern, aus denen ich mich für den ersten der vier Lösungsvorschläge entschieden habe.

Phase Zeit	**Unterrichtsinhalte**	**Erwartetes Verhalten der Schüler**	**Aktions- und Sozialform**	**Material, Medien**
Einstieg 2 min	L zeigt Animation als Motivation für die Stunde.	S hören zu und liefern erste Lösungsvorschläge.	FU, LV	Folie, Animation
Erarbeitung 7 min	L führt int-Variablen ein, vergleicht sie mit Variablen vom Typ boolean und nutzt die Analogie „Kiste“. L zeigt Kiste und tauscht den Wert aus. L stellt Vergleichsoperatoren vor.	S hören zu und stellen ggf. Fragen.	FU, LV	Folie, Brief und Kiste
Vertiefen 7 min	L stellt Aufgabe: sattFressen() vor. Zeigt hierzu die App http://learningapps.org/watch?v=pcynfp3na20 L hilft bei Fragen.	S gehen auf die Webseite und bringen die Code-Streifen in eine korrekte Reihenfolge.	Partnerarbeit, Puzzle	Folie, Code-Puzzle
Sicherung 10 min	L öffnet das Puzzle. L stellt Fragen bei Ungenauigkeiten. *Evtl. bewusst Endlosschleife kreieren und besprechen.*	Ein S führt das Puzzle am Lehrer-PC vor und beschreibt, was die einzelnen Codezeilen bewirken. Andere S korrigieren ggf. [L1]	FU, SV	Code-Puzzle
Vertiefen 14 min	L stellt die Programmieraufgabe zur Methode reiheLegen() vor.	S implementieren die Methode.	LV → EA	Schüler-PC, Folien
Sicherung 5 min	L stellt fragen bei Ungenauigkeiten.	Ein S stellt seine projizierte Lösung vor. [L2]	FU, SV	Beamer
Ausblick	Evtl. kurz Methoden mit Rückgabewert vom Typ int motivieren.		FU, LV	Folie
	weitere Fragen: • Wie kann man sattFressen() verbessern? • Wie kann man reiheLegen() verbessern?	*→ nicht gegen Mauer laufen!* *→ entweder weitere Bedingung im Schleifenkopf oder Variable auf Endwert setzen*		

Tabelle 4: Tabellarischer Verlauf des Alternativentwurfs

6.4 Didaktische Begründung

Prinzipiell halte ich das Code-Puzzle für eine sehr geeignete Methode in diesem Abschnitt der Stoffvermittlung. Die spielerische Form aktivierte und motivierte die Schüler. Auch der Umstand, dass die Klasse mit der Methode bis dato nicht vertraut war, tat dem Interesse der Schüler keinen Abbruch, sondern führte eher zu einer umso regeren Beteiligung.

Das operative Üben ist darüber hinaus ein wichtiger Aspekt im Verstehensprozess komplexer Sachverhalte.[10] In diesem Sinne schult die Methode des Code-Puzzles sowohl die Reversibilität als auch die Kompositionsfähigkeit der Schüler und besitzt daher das Potential, die bereits vorhandenen Kenntnisse zu Variablen weiter zu schärfen.

Die gewählte Sozialform der Partnerarbeit verringert die Anfälligkeit für fehlerhafte Lösungen. Auch sind die Schüler durch das kooperative Arbeiten dazu angehalten, ihr Verständnis eines komplexen Variablenbegriffs weiter auszubilden. Beide Aspekte werden noch einmal dadurch unterstützt, dass das Puzzle den Fokus auf die Semantik und nicht auf die Syntax legt. Diese didaktische Reduzierung erleichtert es den Schülern den neuen Datentyp der Integer in ihr bestehendes Konzept von Variablen einzugliedern und zu begreifen. Von einem enaktiven Zugang, wie ihn die analogen Code-Streifen bieten, kann bei der Lernapp nur eingeschränkt die Rede sein. Dafür bietet die digitale Variante die Möglichkeit eines sofortigen Feedback. Auf Wunsch können die Schüler jederzeit feststellen, ob ihre aktuelle Lösung korrekt oder fehlerhaft ist.

Als weiterer Mangel meines alternativen Unterrichtsentwurfs kann festgehalten werden, dass durch ihn kein „Schülerprodukt“ entsteht; es ist also nicht möglich, die erarbeitete Lösung klassisch abzuheften, um zum Lernen oder Nachschlagen auf die zurückgreifen zu können. Andererseits lässt sich das digitale Puzzle wiederholend durchführen und eignet sich dadurch beispielsweise für die vertiefende Vorbereitung auf eine Prüfung. Darüber hinaus ist es dem einzelnen Schüler jederzeit freigestellt, die korrekte Lösung auch ohne Aufforderung zu notieren, um sie in den eigenen Unterlagen parat zu haben.

Literatur

[1] *Rahmenlehrplan für die gymnasiale Oberstufe – Informatik.* Senatsverwaltung für Bildung, Jugend und Sport Berlin. https://www.berlin.de/sen/bildung/unterricht/faecher-rahmenlehrplaene/rahmenlehrplaene/mdb-sen-bildung-unterricht-lehrplaene-sek2_informatik.pdf (Stand 14.03.2020), 2015.

[2] Baier, Christel, Manuela Berg und Walter Nauber: *Formale Systeme WS 2011/2012 Skript zur Vorlesung.* TU Dresden. https://www.researchgate.net/profile/Christel_Baier/publication/267409804_Formale_Systeme_WS_20112012_Skript_zur_Vorlesung/links/54e24b020cf2966637965e77/Formale-Systeme-WS-2011-2012-Skript-zur-Vorlesung.pdf (Stand 15.03.2020).

[3] CS Unplugged: *Finite State Automata.* University of Canterbury, NZ. https://classic.csunplugged.org/finite-state-automata/ (Stand 14.03.2020).

[4] Köbler, Johannes: *Einführung in die Theoretische Informatik WS 2015/16 Skript zur Vorlesung.* Humboldt-Universität zu Berlin. https://www.informatik.hu-berlin.de/de/forschung/gebiete/algorithmenII/Lehre/ws15/einftheo/skript/ethi-skript.pdf/at_download/file (Stand 15.03.2020).

[5] Klafki, Wolfgang: *Die didaktische Analyse als Kern der Unterrichtsvorbereitung.* In: *Studien zur Bildungstheorie und Didaktik*, Seiten 135–142. Beltz, Weinheim, 1963.

[6] Leibniz-Gymnasium: *Schulcurriculum Informatik.* Berlin-Kreuzberg, 2018.

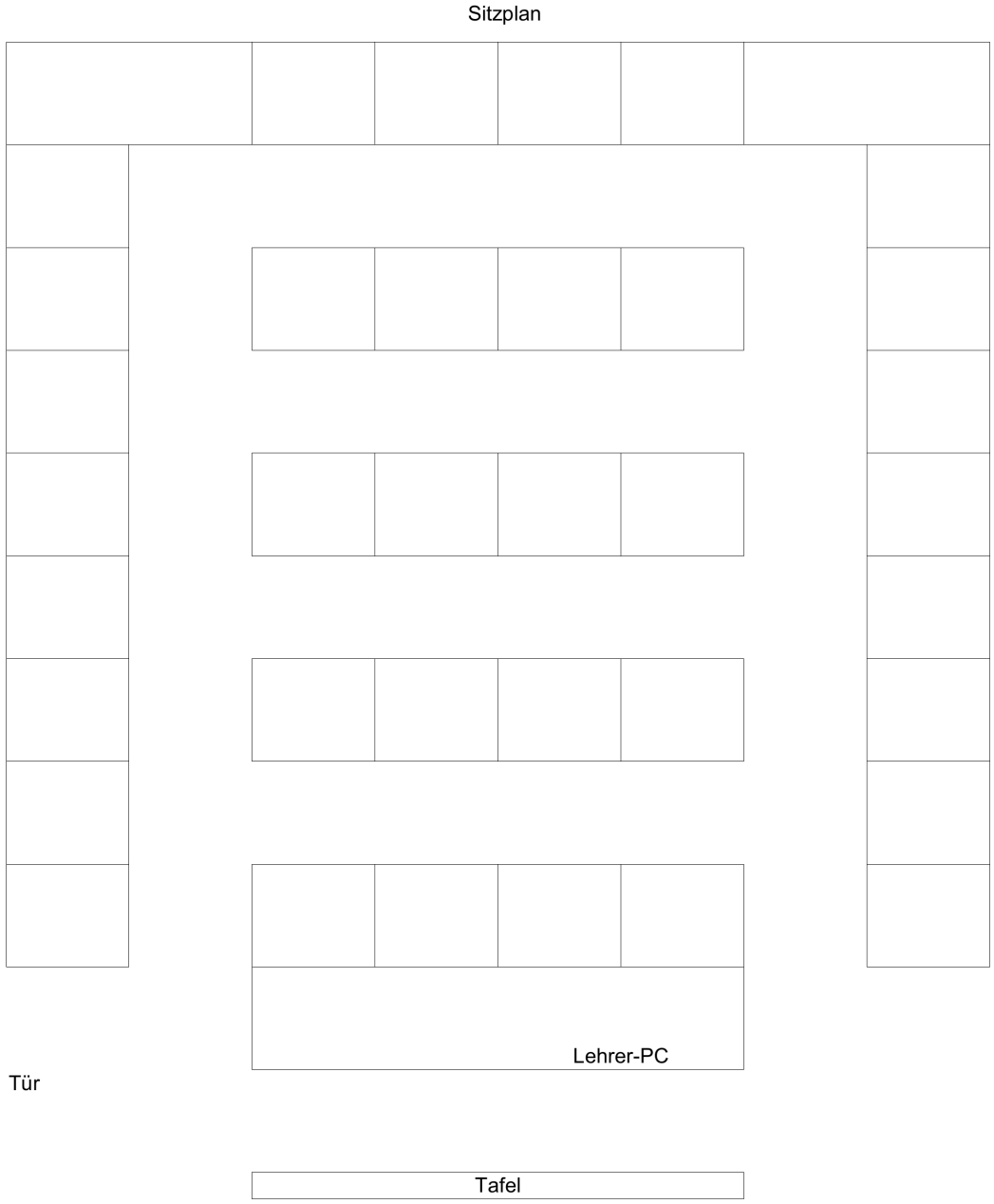
[7] Meyer, Hilbert: *Ergänzung zum Leitfaden – Zehn Beobachtungsbogen zu Einzelmerkmalen.* In: *Leitfaden Unterrichtsvorbereitung*, Seite 227 ff. Cornelsen Scriptor, Berlin, 8. Auflage, 2007.

[8] Meyer, Hilbert: *Leitfaden Unterrichtsvorbereitung.* Cornelsen Scriptor, Berlin, 9. Auflage, 2007.

[9] Röhner, Gerhard *et al.*: *Bildungsstandards Informatik für die Sekundarstufe II - Empfehlungen der Gesellschaft für Informatik e. V. erarbeitet vom Arbeitskreis Bildungsstandards SII.* LOG IN, 36(183/184), Februar 2016.

[10] Wittmann, Erich Ch.: *Objekte-Operationen-Wirkungen: Das operative Prinzip in der Mathematikdidaktik.* Mathematik lehren, 11:7–11, 1985.

A Raumplan S15



B Material zur Doppelstunde Einführung deterministischer endlicher Automaten

B.1 Tafelbilder

Ein deterministischer endlicher Automat (kurz DEA) wird durch ein Tupel M = (Z, Σ, δ, q_0, E) beschrieben, mit

Z : endliche Menge der Zustände — Inseln {P,F,T,S,V,M,K}

Σ : endliches Eingabealphabet — Schiffe {A,B}

δ : Überführungsfunktion — Schiffsrouten δ(P,A) = F, δ(P,B) = K, ...

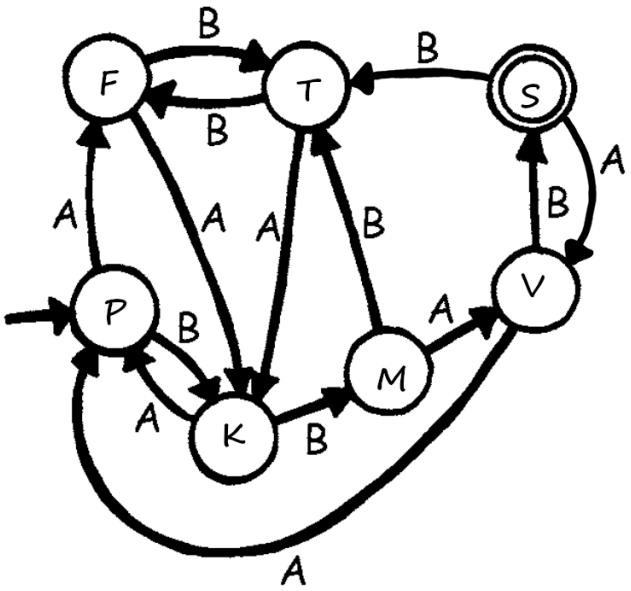
$Z \times \Sigma \rightarrow Z$

oder

δ	P	F	T	S	V	M	K
A	F	K	K	V	P	V	P
B	K	T	F	T	S	T	M

q_0 : Startzustand — Startinsel q_0 = P

E : Menge der Endzustände — Ziel(e) E = {S}

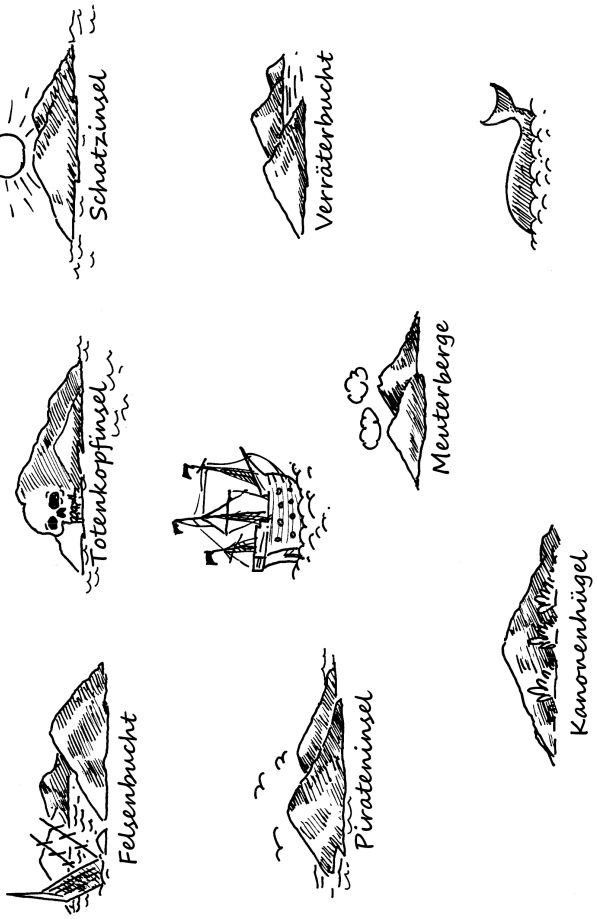


B.2 Schatzkarte

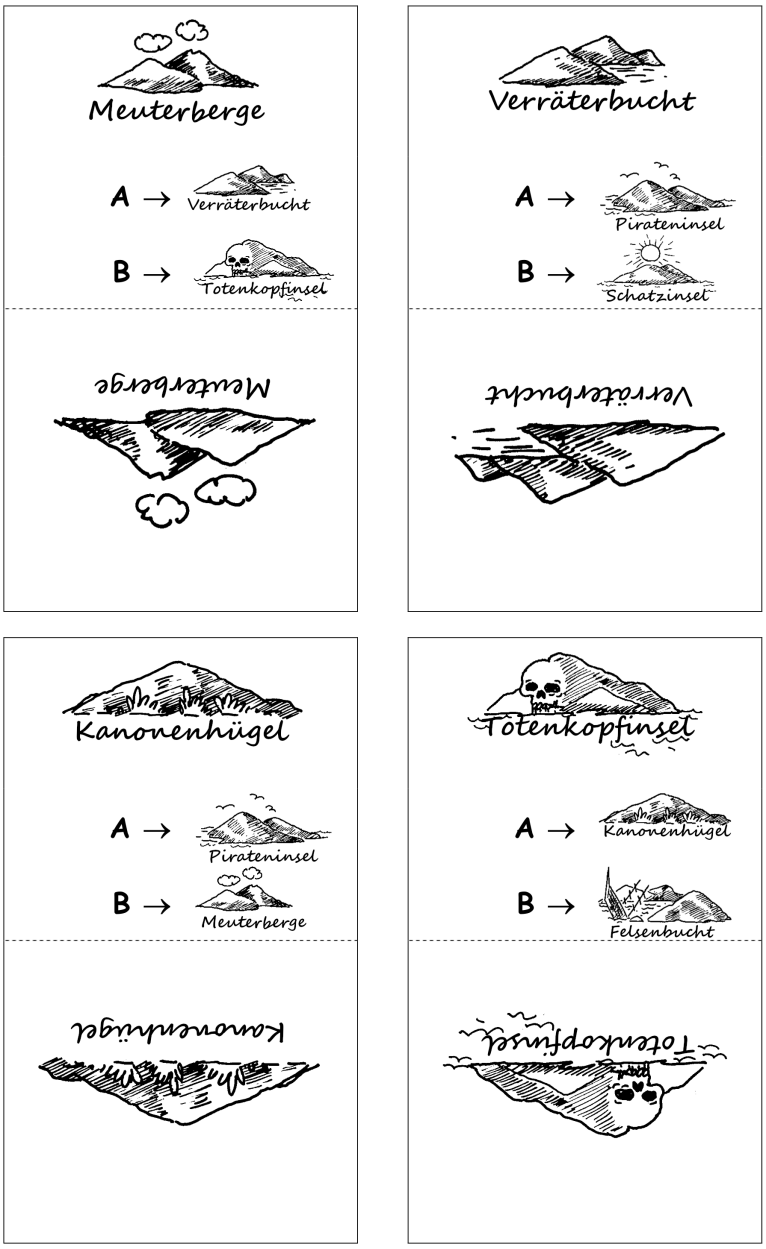
Aufgabe:

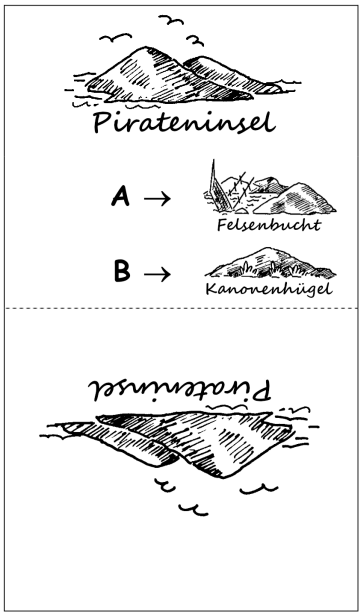
(a) Finde einen Weg zur Schatzinsel. Notiere auf der Schatzkarte, wie man von Insel zu Insel kommt.

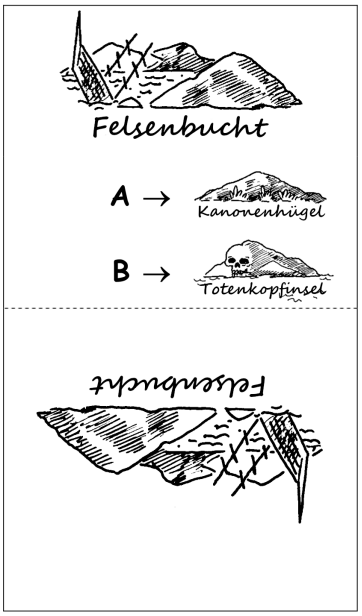
(b) Vervollständige die Schatzkarte.

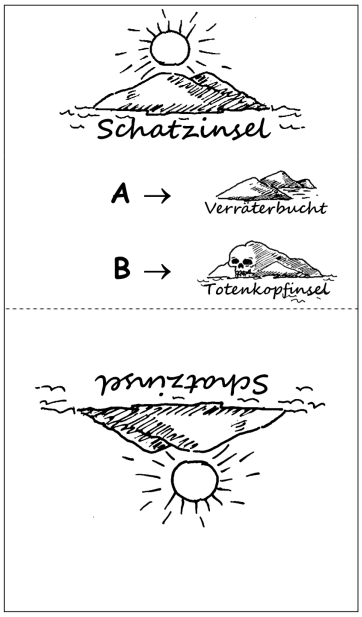


B.3 Wegweiserkarten




Pirateninsel
A →
Felsenbucht
B →
Kanonenhügel
Pirateninsel

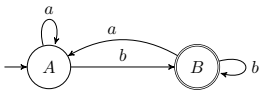

Felsenbucht
A →
Kanonenhügel
B →
Totenkopfinsel
Felsenbucht


Schatzinsel
A →
Verräterbucht
B →
Totenkopfinsel
Schatzinsel

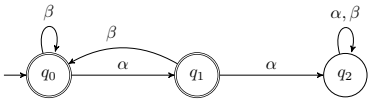
B.4 Arbeitsblätter

Aufgabe 1 (Tandem)**:**
Bearbeite die Aufgaben allein. Überprüfe deine Lösungen dann mit deinem Tandempartner.

(a) Schreibe den Automaten in Tupelschreibweise.



(b) Schreibe den Automaten in Tupelschreibweise.



(c) Stelle den Automaten M in grafischer Schreibweise dar.
$M = (\{A, B\}, \{a, b\}, \delta, A, \{A\})$ mit

δ	A	B
a	B	A
b	A	B

(d) Stelle den Automaten M in grafischer Schreibweise dar.
$M = (\{S_0, S_1, S_2, S_3\}, \{0, 1\}, \delta, S_0, \{S_2\})$ mit

δ	S_0	S_1	S_2	S_3
0	S_3	S_2	S_3	S_3
1	S_1	S_3	S_3	S_3

Aufgabe 2 (Ich-Du-Wir)**:**
Bearbeite die Aufgaben zunächst allein. Tausche dich dann mit deinem Banknachbarn aus.

(a) Erkläre, warum die Adjektive „deterministisch“ und „endlich“ zur Bezeichnung der DEAs verwendet werden.

(b) Gib Unterschiede und Gemeinsamkeiten von DEAs und Automaten aus dem Alltag (z.B. Geldautomat) an.

Aufgabe 1 (Tandem)**:**
Bearbeite die Aufgaben allein. Überprüfe deine Lösungen dann mit deinem Tandempartner.

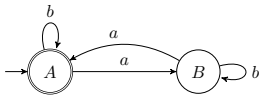
(a) Stelle den Automaten M in grafischer Schreibweise dar.
$M = (\{A, B\}, \{a, b\}, \delta, A, \{B\})$ mit

δ	A	B
a	A	A
b	B	B

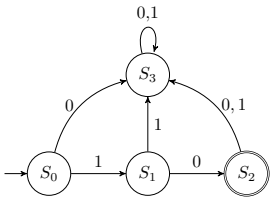
(b) Stelle den Automaten M in grafischer Schreibweise dar.
$M = (\{q_0, q_1, q_2\}, \{\alpha, \beta\}, \delta, q_0, \{q_0, q_1\})$ mit

δ	q_0	q_1	q_2
α	q_1	q_2	q_2
β	q_0	q_0	q_2

(c) Schreibe den Automaten in Tupelschreibweise.



(d) Schreibe den Automaten in Tupelschreibweise.



Aufgabe 2 (Ich-Du-Wir)**:**
Bearbeite die Aufgaben zunächst allein. Tausche dich dann mit deinem Banknachbarn aus.

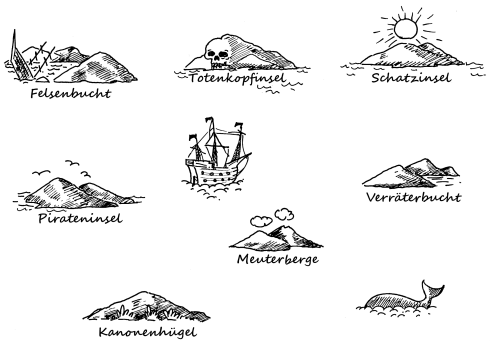
(a) Erkläre, warum die Adjektive „deterministisch“ und „endlich“ zur Bezeichnung der DEAs verwendet werden.

(b) Gib Unterschiede und Gemeinsamkeiten von DEAs und Automaten aus dem Alltag (z.B. Geldautomat) an.

B.5 Folien

Willkommen in 2020

Die Schatzsuche



Die Schatzsuche

Teilnehmer:

- 7 Personen als Wegweiser
- alle anderen sind Schatzsucher

Regeln:

- Die Schatzsucher starten bei der Pirateninsel.
- Auf jeder Insel können sie sich für Schiff A oder Schiff B entscheiden.
- Der Wegweiser sagt das Ziel des gewählten Schiffes.
- Der Schatzsucher notiert die Information auf seiner Schatzkarte und begibt sich zu dem genannten Ziel.

Aufgabe:

(a) Finde einen Weg zur Schatzinsel. Notiere auf der Schatzkarte, wie man von Insel zu Insel kommt.

(b) Vervollständige die Schatzkarte.

Die Schatzsuche: Auswertung

- Was sind die wesentlichen Bestandteile einer Schatzkarte?

Die Schatzsuche: Auswertung

- Was sind die wesentlichen Bestandteile einer Schatzkarte?
- Wie können wir Start- und Zielinsel markieren?

Die Schatzsuche: Auswertung

- Was sind die wesentlichen Bestandteile einer Schatzkarte?
- Wie können wir Start- und Zielinsel markieren?

Die Schatzsuche: Auswertung

- Was sind die wesentlichen Bestandteile einer Schatzkarte?
- Wie können wir Start- und Zielinsel markieren?
- Was hat das mit Informatik zu tun?

Die Schatzsuche: Auswertung

- Was sind die wesentlichen Bestandteile einer Schatzkarte?
- Wie können wir Start- und Zielinsel markieren?
- Was hat das mit Informatik zu tun? Was bedeutet das Wort Informatik?

Die Schatzsuche: Auswertung

- Was sind die wesentlichen Bestandteile einer Schatzkarte?
- Wie können wir Start- und Zielinsel markieren?
- Was hat das mit Informatik zu tun? Was bedeutet das Wort Informatik?
  „automatische Informationsverarbeitung“

Die Schatzsuche: Auswertung

- Was sind die wesentlichen Bestandteile einer Schatzkarte?
- Wie können wir Start- und Zielinsel markieren?
- Was hat das mit Informatik zu tun? Was bedeutet das Wort Informatik?
  „automatische Informationsverarbeitung“ bzw. Information und Automatik

Deterministische endliche Automaten

Herr Röse

06.01.2020

Aufgabe 1 (Tandemaufgabe): Bearbeite die Aufgaben allein. Überprüfe deine Lösungen dann mit deinem Tandempartner.

Aufgabe 2 (Ich-Du-Wir): Bearbeite die Aufgaben zunächst allein. Tausche dich dann mit deinem Banknachbarn aus.

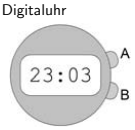
Wofür benötigen wir Automaten?

- für Modellierungsaufgaben, um Algorithmen zu entwickeln und zu implementieren

Wofür benötigen wir Automaten?

- für Modellierungsaufgaben, um Algorithmen zu entwickeln und zu implementieren

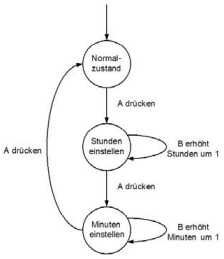
  Bsp.: Digitaluhr



Wofür benötigen wir Automaten?

- für Modellierungsaufgaben, um Algorithmen zu entwickeln und zu implementieren

  Bsp.: Digitaluhr



Wofür benötigen wir Automaten?

- um Algorithmen zu analysieren

Wofür benötigen wir Automaten?

- um Algorithmen zu analysieren
- Rechnermodelle - Berechenbarkeit, Komplexität, Laufzeitanalysen

Wofür benötigen wir Automaten?

- um Algorithmen zu analysieren
- Rechnermodelle - Berechenbarkeit, Komplexität, Laufzeitanalysen
- Halteproblem:
  Gibt es einen Algorithmus (d.h. einen Automaten), der für jeden Algorithmus entscheiden kann, ob dieser enden wird?

Definition:

- Ein *Alphabet* $\Sigma = \{a_1, \ldots, a_m\}$ ist eine geordnete Menge von endlich vielen *Zeichen*
  Bsp.: $\{A, B\}, \{0, 1\}, \{a, b, \ldots, z\}$, {aufheben, ablegen}

Definition:

- Ein *Alphabet* $\Sigma = \{a_1, \ldots, a_m\}$ ist eine geordnete Menge von endlich vielen *Zeichen*
  Bsp.: $\{A, B\}, \{0, 1\}, \{a, b, \ldots, z\}$, {aufheben, ablegen}
- Eine Folge $x = x_1 \ldots x_n$ von n Zeichen heißt *Wort*.

Definition:

- Ein *Alphabet* $\Sigma = \{a_1, \ldots, a_m\}$ ist eine geordnete Menge von endlich vielen *Zeichen*
  Bsp.: $\{A, B\}, \{0, 1\}, \{a, b, \ldots, z\}$, {aufheben, ablegen}
- Eine Folge $x = x_1 \ldots x_n$ von n Zeichen heißt *Wort*.
- Spezialfall: Für $n = 0$ bezeichnen wir das *leere Wort* mit ε.

Definition:

- Ein *Alphabet* $\Sigma = \{a_1, \ldots, a_m\}$ ist eine geordnete Menge von endlich vielen *Zeichen*
  Bsp.: $\{A, B\}, \{0, 1\}, \{a, b, \ldots, z\}$, {aufheben, ablegen}
- Eine Folge $x = x_1 \ldots x_n$ von n Zeichen heißt *Wort*.
- Spezialfall: Für $n = 0$ bezeichnen wir das *leere Wort* mit ε.
- Eine Menge von Wörtern über einem Alphabet Σ heißt *Sprache über* Σ.
  Bsp.: $\{\varepsilon, A, B, AA, AB, BA, BB, AAA, \ldots\}$

- Erhält ein DEA ein Wort x als Eingabe, so erarbeitet er x von links nach rechts ab.

- Erhält ein DEA ein Wort x als Eingabe, so erarbeitet er x von links nach rechts ab.
- Der DEA *akzeptiert* ein Wort x, falls er nach dem Lesen des letzten Zeichens auf einem Endzustand endet.

- Erhält ein DEA ein Wort x als Eingabe, so erarbeitet er x von links nach rechts ab.
- Der DEA *akzeptiert* ein Wort x, falls er nach dem Lesen des letzten Zeichens auf einem Endzustand endet.
- Die von einem DEA *akzeptierte Sprache* ist die Menge aller Wörter, die er akzeptiert.

- Erhält ein DEA ein Wort x als Eingabe, so erarbeitet er x von links nach rechts ab.
- Der DEA *akzeptiert* ein Wort x, falls er nach dem Lesen des letzten Zeichens auf einem Endzustand endet.
- Die von einem DEA *akzeptierte Sprache* ist die Menge aller Wörter, die er akzeptiert.

Aufgabe:
Gib drei Wörter an, die vom Schatzkartenautomat:

(a) akzeptiert werden,

(b) nicht akzeptiert werden.

C Material zum Alternativentwurf

C.1 Startbild der Animation

C.2 Code-Puzzle

```
public void sattFressen(){

int anzahlKoerner = 0;

while(vornFrei()){

vor();

if(anzahlKoerner < 5){

if(kornDa()){

nimm();

anzahlKoerner++; //erhöht den Wert der Variable um 1

      }
   }

      }
   }
```

C.3 Alte Folien

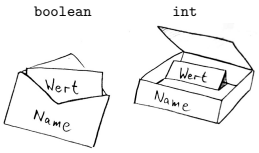
Hamster sind auch mal satt

Hamster sind auch mal satt

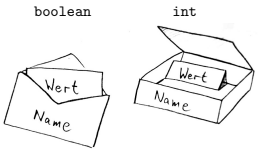
Situation: Der Hamster soll genau 5 Körner fressen.

Wie können wir ihm beibringen nicht mehr Körner aufzunehmen?

Ganzzahlige Variablen — Integer



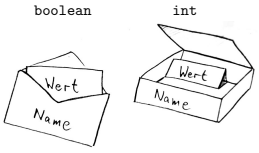
Ganzzahlige Variablen — Integer



Deklaration:

Typ	Name		Wert
`boolean`	`merke`	`=`	`false;`

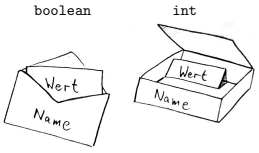
Ganzzahlige Variablen — Integer



Deklaration:

Typ	Name		Wert
`boolean`	`merke`	`=`	`false;`
`int`	`anzahl`	`=`	`1337;`

Ganzzahlige Variablen — Integer



Deklaration:

Typ	Name		Wert
`boolean`	`merke`	`=`	`false;`
`int`	`anzahl`	`=`	`1337;`

Vergleichsoperatoren:

Code:	`<=`	`>=`	`<`	`>`	`==`	`!=`
Bedeutung:	$\leq$	$\geq$	$<$	$>$	$=$	$\neq$

Aufgabe: sattFressen()

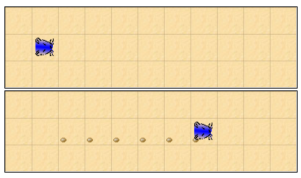
Ordnet die Zeilen so an, dass der Hamster genau 5 Körner isst.

Lösungsvorschlag

```
public void sattFressen(){
   int anzahlKoerner = 0;
   while(vornFrei()){
      vor();
      if(anzahlKoerner < 5){
         if(kornDa()){
            nimm();
            anzahlKoerner++; //erhöht den Wert der
                             //Variable um 1
         }
      }
   }
}
```


Aufgabe: reiheLegen()

Der Hamster möchte auch im nächsten Jahr Körner haben. Also muss er sich ein Beet anlegen.
Schreibt eine Methode `reiheLegen()` in der der Hamster 6 Körner hintereinander legt.



Führe die Reihe fort

Wir kennen

- Variablen vom Typ `boolean`

Führe die Reihe fort

Wir kennen

- Variablen vom Typ `boolean`
- Methoden mit einem Rückgabewert vom Typ boolean

Führe die Reihe fort

Wir kennen

- Variablen vom Typ `boolean`
- Methoden mit einem Rückgabewert vom Typ boolean
- Variablen vom Typ `int`

Führe die Reihe fort

Wir kennen

- Variablen vom Typ `boolean`
- Methoden mit einem Rückgabewert vom Typ boolean
- Variablen vom Typ `int`
- ...

Aufgabe: koernerAufFeld()

Schreibt eine Methode `koernerAufFeld()` in der der Hamster die Körner auf seinem Feld zählt und die Anzahl zurückgibt.

Die Körner sollen danach wieder auf dem Feld liegen!

Aufgabe: koernerAufFeld()

Schreibt eine Methode `koernerAufFeld()` in der der Hamster die Körner auf seinem Feld zählt und die Anzahl zurückgibt.

Die Körner sollen danach wieder auf dem Feld liegen!

Tipp: `n--` verringert den Wert der Variable `n` um 1.

C.4 Neue Folien

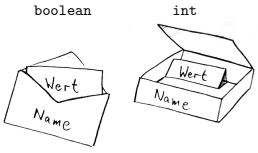
Hamster sind auch mal satt

Hamster sind auch mal satt

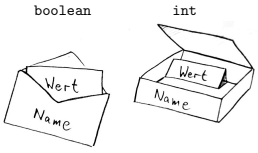
Situation: Der Hamster soll genau 5 Körner fressen.

Wie können wir ihm beibringen nicht mehr Körner aufzunehmen?

Ganzzahlige Variablen — Integer



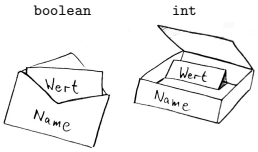
Ganzzahlige Variablen — Integer



Deklaration:

Typ	Name		Wert
`boolean`	`merke`	`=`	`false;`

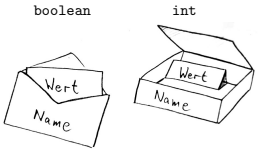
Ganzzahlige Variablen — Integer



Deklaration:

Typ	Name		Wert
`boolean`	`merke`	=	`false;`
`int`	`anzahl`	=	`1337;`

Ganzzahlige Variablen — Integer



Deklaration:

Typ	Name		Wert
`boolean`	`merke`	=	`false;`
`int`	`anzahl`	=	`1337;`

Vergleichsoperatoren:

Code:	`<=`	`>=`	`<`	`>`	`==`	`!=`
Bedeutung:	$\leq$	$\geq$	$<$	$>$	$=$	$\neq$

Aufgabe: sattFressen()

1. Geht auf `learningapps.org/watch?v=pcynfp3na20`.
2. Ordnet die Zeilen so an, dass der Hamster genau 5 Körner isst.

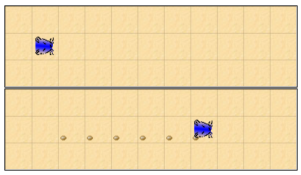
   Hinweis: Wir gehen davon aus, dass in der Reihe vor dem Hamster ausreichend Körner liegen.
3. Beschreibt anschließend zeilenweise, was beim Programmablauf passiert.

Lösungsvorschlag

```
public void sattFressen(){
   int anzahlKoerner = 0;
   while(anzahlKoerner < 5){
      if(kornDa()){
         nimm();
         anzahlKoerner++; //erhöht den Wert der Variable um 1
      }
      vor();
   }
}
```


Aufgabe: reiheLegen()

Der Hamster möchte auch im nächsten Jahr Körner haben. Also muss er sich ein Beet anlegen.
Schreibt eine Methode `reiheLegen()` in der der Hamster 6 Körner hintereinander legt.



Führe die Reihe fort

Wir kennen

- Variablen vom Typ `boolean`

Führe die Reihe fort

Wir kennen

- Variablen vom Typ `boolean`
- Methoden mit einem Rückgabewert vom Typ boolean

Führe die Reihe fort

Wir kennen

- Variablen vom Typ `boolean`
- Methoden mit einem Rückgabewert vom Typ boolean
- Variablen vom Typ `int`

Führe die Reihe fort

Wir kennen

- Variablen vom Typ `boolean`
- Methoden mit einem Rückgabewert vom Typ boolean
- Variablen vom Typ `int`
- ...

Aufgabe: koernerAufFeld()

Schreibt eine Methode `koernerAufFeld()` in der der Hamster die Körner auf seinem Feld zählt und die Anzahl zurückgibt.

Die Körner sollen danach wieder auf dem Feld liegen!

Aufgabe: koernerAufFeld()

Schreibt eine Methode `koernerAufFeld()` in der der Hamster die Körner auf seinem Feld zählt und die Anzahl zurückgibt.

Die Körner sollen danach wieder auf dem Feld liegen!

Tipp: `n--` verringert den Wert der Variable `n` um 1.