2 Vom Problem zum Algorithmus

2.1 Grobe Klassifikation von Problemen

Unser Ziel ist es, Probleme mit Hilfe von Computern zu lösen. Dazu müssen wir dem Computer mitteilen, was er tun soll, um Lösungen zu finden. Da wir bisher nur eine recht vage Vorstellung von dem haben, was Probleme sind, was Computer leisten und wie sie arbeiten, wollen wir zunächst anhand einiger Beispiele Probleme grob klassifizieren und uns überlegen, wie wir selbst Probleme bearbeiten. Im Anschluß daran werden wir versuchen, ein komplexeres Problem mit Hilfe eines Computers zu lösen.

Typ-1-Probleme: Lösung bekannt.
Manche Probleme sind sehr einfach, und man kann die Lösung sofort angeben, weil man sie auswendig weiß.
Beispiel: Ein Pfund Äpfel kostet 1 DM. Was kosten 2 Pfund?

Typ-2-Probleme: Lösung einfach zu berechnen.
Andere Probleme sind etwas schwieriger. Hier wissen wir die Lösung nicht mehr auswendig, aber wir kennen immerhin ein Verfahren oder haben ein Hilfsmittel (z.B. Taschenrechner), mit dem wir die Lösung bestimmen können.
Beispiel: Mein Auto hat für 452 km 38,6 Liter Benzin verbraucht. Wieviel hat es für 100 km verbraucht?

Typ-3-Probleme: Lösung aufwendig zu berechnen.
Eine dritte Kategorie bilden die Probleme, die wir mit einiger Geduld lösen können, weil wir ein allgemeingültiges Lösungsverfahren kennen, z.B. das sture Durchprobieren aller Möglichkeiten.
Beispiel: Matt in drei Zügen (Abb. 1).

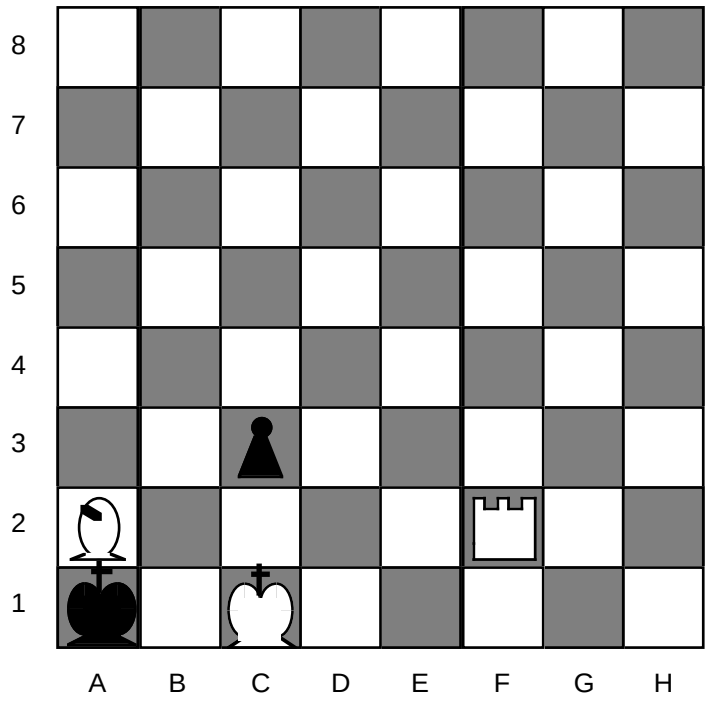


Abb. 1: Weiß zieht und setzt in drei Zügen matt.

Typ-4-Probleme: Theoretisch lösbare, praktisch unlösbare Probleme.
In eine weitere Gruppe fallen Probleme, die wir wohl theoretisch lösen können, weil wir ebenfalls ein Lösungsverfahren kennen, die aber praktisch unlösbar sind, weil das Lösungsverfahren zu aufwendig ist.
Beispiel: Wieviele Sandkörner liegen am Strand der Nordsee?

Typ-5-Probleme: Theoretisch unlösbare Probleme.
Schließlich gibt es noch die Probleme, für die man sicher weiß, daß man sie nicht lösen kann.
Beispiel: Man teile einen Winkel mit Zirkel und Lineal in drei gleichgroße Winkel.

Im folgenden wollen wir ein größeres Problem zwischen Typ-2 und Typ-3 mit Hilfe eines Computers lösen und dabei die hierzu notwendigen Voraussetzungen erarbeiten und analysieren.

2.2 Beispielproblem: Karl-May-Festspiele

Situation.
Jedes Jahr während der Sommermonate finden die Karl-May-Festspiele statt. Die hohe Attraktivität der Spiele sorgt dafür, daß jede Vorstellung ausverkauft ist und daß häufig

Besucher abgewiesen werden müssen. Viele Interessenten lassen sich daher bereits mehrere Wochen vor Vorstellungsbeginn Eintrittskarten reservieren.
Nach einiger Zeit stellten die Festspielorganisatoren aber zunehmend fest, daß vorbestellte Karten (größenordnungsmäßig 300) am Vorstellungstermin nicht abgeholt wurden. Diese Karten wurden dann kurz vor Vorstellungsbeginn unter die anwesenden Besucher verteilt, die es (meist aus Unkenntnis über den Zuschauerandrang) versäumt hatten, sich Karten vorweg reservieren zu lassen.
Dieses Verteilungsverfahren entwickelte sich jedoch im allgemeinen chaotisch. Beim Ausruf "Es sind noch Karten da" setzte meist ein solches Gedränge ein, daß die Kassierer um ihre Gesundheit fürchten mußten. Regelmäßig siegten in dem Tumult die Stärkeren, Kinder wurden zurückgedrängt. Viele verloren angesichts des Andrangs die Hoffnung, eine Eintrittskarte abzubekommen, und fuhren resigniert nach Hause, obwohl bei geregelter Verteilung auch für sie noch Karten vorhanden gewesen wären. Ergebnis: Die Veranstalter behielten Karten übrig und büßten Eintrittsgelder ein. Häufig konnte die Verteilung nicht bis zum Ende der Vorstellung abgewickelt werden. Die Nachzügler störten dann beim Aufsuchen ihrer Sitzplätze die Aufführung. Wegen dieser und einer Reihe weiterer Schwierigkeiten will man daher das Verteilungsverfahren für die kommende Saison grundlegend ändern.

Problem: Entwurf eines Verteilungsverfahrens für Eintrittskarten.
Voraussetzungen: Jedem Interessenten, der eine Eintrittskarte reserviert, wird mitgeteilt, daß die Karten bis spätestens eine halbe Stunde vor Spielbeginn abgeholt werden müssen. Danach wird die Reservierung ungültig. Die nicht abgeholten Karten kommen dann in einen Topf und werden unter den anwesenden Besuchern verteilt. Das Verteilungsverfahren darf höchstens eine halbe Stunde dauern, denn die bisher üblichen Störungen der Vorstellung möchte man zukünftig vermeiden. Wie wählt man nun der Reihe nach die Personen aus der wartenden Menge aus, die Karten erhalten? Man kommt überein, daß Familien bzw. Gruppen mit vielen Kindern bevorzugt werden sollen. Sollte es mehrere Familien/Gruppen mit jeweils gleichvielen Kindern geben, so soll zunächst die Gruppe/Familie Karten erhalten, die die weitere Anreise zu den Festspielen hatte. Der Einfachheit halber will man bei der Bemessung des Anreiseweges nur unterscheiden, ob die Gruppe/Familie aus der Nähe des Veranstaltungsortes oder von auswärts kommt. Sofern auch dies noch nicht zu einer eindeutigen Reihenfolge führt, soll schließlich der früher Eingetroffene zuerst seine Karte erhalten (Motto: "Wer zuerst kommt, mahlt zuerst").

Zusammengefaßt bekommt also zuerst die Familie/Gruppe mit den meisten Kindern ihre Eintrittskarten, bei gleicher Kinderzahl zuerst die Familie/Gruppe, die von auswärts kommt, und bei Familien/Gruppen gleicher Kinderzahl, die alle von auswärts angereist sind, erhält die ihre Karten, die zuerst da war.

2.2.1 Aufgabe

Man plane den organisatorischen Ablauf des Verteilungsverfahrens unter den geschilderten Nebenbedingungen.

Präzise Formulierung des Verfahrens.
Da die meisten Organisationsfunktionen (z.B. Kassieren, Karten kontrollieren etc.) bei den Festspielen von ständig wechselnden Aushilfskräften wahrgenommen werden, soll der Gesamtablauf des Verteilungsverfahrens so aufgeschrieben und gegliedert werden, daß er *von ungeübten Hilfskräften ohne Vorkenntnisse und ohne Zusatzinformationen korrekt* durchgeführt werden kann.

2.2.2 Lösungsansatz 1: Probieren

Grobstruktur.
An einem bestimmten Stand werden Formblätter der Gestalt aus Abb. 2 in genügend großer Stückzahl deponiert.

Name	von außerhalb	Kinderzahl	Ankunftszeit

Abb. 2: Formblätter

Immer wenn Besucher eintreffen, die keine Karte vorbestellt haben, wird ihr Name (bei Gruppen der Name des Leiters, bei Familien der Familienname), sowie Kinderzahl und Ankunftszeit notiert. Sind die Besucher von auswärts gekommen, so schreibt man in die zugehörige Spalte "ja" sonst "nein". Eine halbe Stunde vor Vorstellungsbeginn wird der Stand geschlossen und von zwei Helfern - man schätzte, daß zwei Personen reichen würden - nach den angegebenen Kriterien (Kinderzahl/von auswärts/Ankunftszeit) die Reihenfolge festlegt, nach der die Besucher ihre Karten erhalten.

Verfeinerung des Ablaufs.

Wie müssen die beiden Helfer (Nennen wir sie fortan M1 und M2) bei der Verteilung nun genau vorgehen? Zum Beispiel so: M1 und M2 teilen die Formblätter unter sich auf, so daß jeder etwa gleichviele Kartenvormerkungen hat. Anschließend beginnt jeder eine neue Liste, auf der er die Namen, die Kinderzahlen, die Antworten auf die Frage "von auswärts" und die Ankunftszeit nach den Kriterien geordnet einträgt. An oberster Stelle jeder Liste stehen dann die Gruppen/Familien, die die meisten Kinder haben, von auswärts kommen und am frühesten eingetroffen sind. Am Schluß jeder Liste befinden sich die Gruppen/Familien, die die wenigsten Kinder haben, nicht von auswärts kommen und am spätesten eingetroffen sind. Beide Listen mischen M1 und M2 danach zusammen und schreiben eine neue Liste, die die endgültige Reihenfolge enthält, in der die Karten auf die Wartenden verteilt werden. M1 oder M2 läuft dann mit der endgültigen Liste zur Kasse, von wo aus die Namen der Reihe nach aufgerufen und die gewünschten Karten verkauft werden, bis entweder alle Karten verteilt sind oder alle Personen auf der Liste Karten erhalten haben. Soweit das Verfahren, das in Zukunft durchgeführt werden soll.

Testphase.

Um das Verfahren nicht gleich einem Härtetest an einem der Wochenendspieltage zu unterziehen, startet man es bei einer der werktäglichen Abendvorstellungen.

Eine "Manöverkritik" im Anschluß an diese Vorstellung zeigt, daß das beschriebene Verfahren bereits bei der an diesem Abend relativ kleinen Zahl von 50 Wartenden erhebliche Schwächen aufweist:

- Das Aufnehmen von Besuchern in die Warteliste ist problematischer als man angenommen hat: Viele Wartende rufen gleichzeitig ihre Namen; es gibt Verständigungsschwierigkeiten beim Aufnehmen des Namens ("Buchstabieren Sie bitte!"). Bei der Frage "Kommen Sie von auswärts?" kommen Gegenfragen "Wozu wollen Sie denn das wissen?" usw.
- Die Herstellung einer Reihenfolge der wartenden Besucher nach den angegebenen Kriterien durch die beiden Verteiler M1 und M2 nimmt viel zu viel Zeit in Anspruch. Die Verteilung konnte so gerade in einer halben Stunde abgewickelt werden. Unter harten Bedingungen bei Wochenendvorstellungen mit 200 bis 300 Wartenden ist eine halbe Stunde erheblich zu wenig. Zudem war das Verfahren so umständlich, daß M1 und M2 gelegentlich durcheinander kamen und nicht mehr genau wußten, welcher Schritt als nächstes zu tun war. Eine Überprüfung der endgültigen Liste ergab sogar, daß die Verteiler sechs Wartende falsch eingeordnet hatten.

- Bei der Übertragung der Namen auf neue Listen kam es gelegentlich zu Schreibfehlern und Unleserlichkeiten. Bei der Kartenvergabe fühlte sich dann bei Aufruf des Namens plötzlich niemand mehr angesprochen.
- Unter den Wartenden waren zwei Schmidt's und drei Müller's. Beim Aufruf "Müller" mußte die Familie/Gruppe, die gemeint war, erst identifiziert werden. Man hatte bei der Planung übersehen, daß der Name allein zur Unterscheidung nicht ausreicht.

2.2.3 Lösungsansatz 2: Präzisieren, Spezifizieren, Strukturieren

Wir haben in der Testphase festgestellt, daß nahezu alles verbessert werden muß, damit das Verfahren praktisch einsetzbar wird. Hierzu ist es allerdings notwendig, den gesamten Vorgang *klarer zu strukturieren* und die einzelnen Abläufe *genauer zu analysieren und zu präzisieren*.

Weitere Strukturierung und Präzisierung des Verfahrens.
Offensichtlich unterteilt sich das Verfahren in *drei große Teile*, die in sich relativ abgeschlossen sind. Der *erste* Teil besteht in der Aufnahme der wartenden Besucher in eine Liste. Nach Abschluß dieser Phase beginnt eine halbe Stunde vor Vorstellungsbeginn der *zweite* Teil, bei dem die Wartenden nach den angegebenen Kriterien Kinderzahl/von auswärts?/Ankunftszeit geordnet werden. Die *dritte* Phase schließlich besteht im Ausrufen der Wartenden in der Reihenfolge, wie sie von der 2. Phase hergestellt worden ist, und in der Vergabe der übriggebliebenen Eintrittskarten.

Phase 1: Aufnahme der wartenden Besucher.
Eine Präzisierung in dieser Phase besteht aus der Festlegung, welche Informationen man von den wartenden Besuchern benötigt und wie diese festgehalten werden. In der Fachsprache der Informatik sucht man eine geeignete *Modellierung* der Besucher, also eine Darstellung eines Menschen durch eine handhabbare *Repräsentation*.

Präzisierung der Daten.
Im ersten Lösungsansatz hat man eine ungeeignete Modellierung der Besucher gewählt und ohne große Überlegung den Namen, die Kinderzahl, ja/nein auf die Frage "von auswärts?" und die Ankunftszeit notiert.
Der kurze Test hat bereits gezeigt, daß sich das Aufnehmen des Namens nicht bewährt hat (mehrere "Müller").
Die Information "Ankunftszeit" ist bei genauem Hinsehen exakter, als sie benötigt wird: Es interessiert ja in Wirklichkeit gar nicht die genaue Zeit, sondern nur die Reihenfolge, in der die Besucher eintreffen und sich registrieren lassen. Eine einfache Numerierung

der Ankommenden mit 1,2,3,... erfüllt den gleichen Zweck. Der Erstangekommene erhält die Nummer 1, der Zweite die 2 usw. Gleichzeitig wird durch diese Festlegung auch das Identifizierungsproblem von oben gelöst: Anstelle der Namen werden jetzt die Ordnungsnummern verwendet. Jede Familie/Gruppe muß sich nun allerdings ihre Nummer merken. Betrachten wir noch kurz die Informationen "Kinderzahl" und "von auswärts". Anscheinend läßt sich hier nichts verbessern. Die *endgültige Präzisierung* der Daten (die *Repräsentation* eines Besuchers) lautet also:

Kinderzahl/von auswärts bzw. nicht von auswärts/Ordnungsnummer.

Präzisierung der Datenaufnahme.
Wir erinnern uns, daß es beim Festhalten der Daten Verständigungsschwierigkeiten und Drängeleien gab. Nach längeren Überlegungen verfällt man auf folgende Lösung, welche zusätzlich noch den Vorteil hat, aus Sicht der Veranstalter also gewissermaßen *automatisch*, d.h. ohne den Einsatz eines Helfers, ablaufen zu können.
Man besorgt sich einen "Spender" für Wartenummern, wie er häufig bei Ärzten oder in Ämtern verwendet wird. Diesen Spender füllt man mit Zetteln der Art aus Abb. 3.

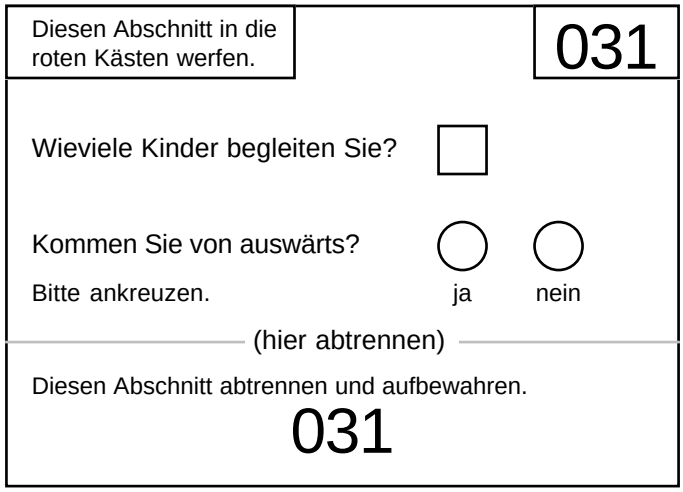


Abb. 3: Muster für einen Zettel

Eintreffende Besucher/Gruppen/Familien, die keine Karte vorbestellt haben, nehmen einen Zettel aus dem Spender, füllen die obere Hälfte aus und trennen sie ab. Den unteren Teil verwahren sie, den oberen Teil werfen sie in einen Briefkasten. Auf beiden Teilen des Zettels ist die Ordnungsnummer bereits aufgedruckt (hier die Zahl 31). Wir nehmen im folgenden an, daß man mit 500 Ordnungsnummern auskommt.
Die Präzisierung der ersten Phase (Daten und Ablauf) ist damit abgeschlossen.

Phase 2: Sortieren der Besucher.
In dieser Phase müssen wir vorwiegend das Verfahren *beschleunigen* und weniger fehleranfällig realisieren. Eine Präzisierung besteht daher im wesentlichen aus der Zerlegung des Sortiervorgangs in *klar verständliche Einzelschritte*, die von M1 und M2 fehlerfrei nachvollzogen werden können.

Präzisierung und Strukturierung des Sortierverfahrens.
Es fällt auf, daß M1 und M2 nur während der halben Stunde vor Vorstellungsbeginn beschäftigt sind. In der übrigen Zeit haben beide nichts zu tun. Wie kann man die Arbeiten, die in dieser halben Stunde anfallen, auf einen längeren Zeitraum verteilen?
Man hat beobachtet, daß etwa zwei Stunden vor Vorstellungsbeginn die ersten Besucher eintreffen. Wenn nun z.B. M1 von diesem Zeitpunkt an alle 10 Minuten den Briefkasten mit den Kartenwünschen leert und vorsortiert, verteilt sich die Gesamtarbeit über einen größeren Zeitraum und kann problemloser bewältigt werden. Genauer: M1 leert alle 10 Minuten (zum erstenmal 2 Stunden vor Vorstellungsbeginn, zum letztenmal eine halbe Stunde vor Vorstellungsbeginn) den Briefkasten. Bei der Sortierung arbeitet M1 mit zwei Zettelstapeln. Der eine ist immer nach den drei Kriterien Kinderzahl/von auswärts/Ordnungsnummer sortiert. Der andere Stapel besteht aus den Zetteln, die M1 gerade aus dem Briefkasten geholt hat. M1 sortiert nun diesen Stapel nach den erwähnten Kriterien. Anschließend mischt M1 beide Stapel so, daß ein einziger sortierter Stapel übrigbleibt.

Testphase.
Ergebnis der Analyse ist ein Verfahren zur Sortierung der Zettel, das in der vorgeschriebenen Zeit abgewickelt und überdies von nur *einer* Person (M2 wird offenbar nicht mehr benötigt) bewältigt werden kann.
Als man M (M1 heißt jetzt einfach M) das Verfahren erklärt und ihn bittet, es einmal probehalber nachzuvollziehen, kommt er mit allem gut zurecht. Nur die Bedeutung von "Mischen" hat M falsch verstanden (Wissen Sie genau, was Mischen bedeutet?). Statt aus zwei sortierten Stapeln einen sortierten zu machen, bringt M alle Zettel durcheinander, weil er "Mischen" wie "Spielkarten mischen" verstanden hat.

Zerlegung der Mischvorschrift in Einzelschritte.
Die Anweisung "Mischen zweier sortierter Zettelstapel" wird daher noch weiter zergliedert und *in einfachere Schritte zerlegt:*

> Seien A und B die beiden sortierten Zettelstapel, die zum neuen Stapel C zusammengemischt werden sollen.

Vergleiche fortwährend die beiden obersten Zettel von A und B und füge den vorrangigen hinten an den Stapel C an.

Mit dieser Präzisierung hat man offenbar das *Sprachniveau* von M getroffen, denn nun ist M in der Lage alles korrekt durchzuführen.

Phase 3: Ausrufen der Wartenden.

Diese Phase hatte schon, von Verständigungsschwierigkeiten (Stichwort: "Müller") abgesehen, immer reibungslos funktioniert und bedarf keiner Verbesserungen. M begibt sich mit seinem sortierten Zettelstapel an die Kasse, ruft, beginnend mit dem obersten Zettel, die Ordnungsnummern der Reihe nach auf und verkauft die gewünschte Kartenzahl.

Testphase.

Nach diesen gründlichen Überlegungen vertraut man der Leistungsfähigkeit des Verfahrens bereits sehr und will es bei der nächsten Wochenendvorstellung einsetzen. Eine erneute "Manöverkritik" belegt, daß sich die eingehende Analyse gelohnt hat. Das Verfahren weist keine nennenswerten Schwächen mehr auf. Trotzdem gibt es weitere Verbesserungsvorschläge, auf die man gerne eingeht.

Weitere Verbesserung der Effizienz und Fehleranfälligkeit.

Der Verteiler M gerät immer noch gelegentlich beim Mischen und beim Sortieren durcheinander. Er merkt seinen Fehler zwar immer sofort, trotzdem ist ihm das unangenehm. Sein Problem besteht darin, die unterschiedlichen Rangfolgen korrekt zu verarbeiten: Einerseits ist eine *große* Kinderzahl bevorrechtigt vor einer *kleinen*, andererseits aber eine *kleine* Ordnungsnummer bevorrechtigt vor einer *großen*, und drittens ist ein "ja" auf die Frage "von auswärts?" bevorrechtigt vor einem "nein". Nach kurzer Überlegung ändert man den obigen Lösungsansatz.

2.2.4 Lösungsansatz 3: Optimieren

Man ändert das *Modell* der Besucher erneut, indem man eine Repräsentation wählt, die die Sortierung erleichtert.

Codierung der Daten.

Immer wenn M den Briefkasten geleert hat, schreibt er sofort (noch vor dem Sortieren) die Information jedes dieser Zettel in *codierter* Form als Folge von 5 bis 7 Ziffern noch einmal darauf. Die ersten ein bis drei Ziffern sind das Ergebnis der Rechnung

100 - Kinderzahl.

Die nächste Ziffer ist eine 0, falls die Antwort auf die Frage "von auswärts" "ja" lautet, und eine 1, falls die Antwort "nein" ist. Die folgenden drei Ziffern bilden die Ordnungsnummer. Auf einen Zettel mit den Antworten

Kinderzahl=26/ja/Ordnungsnummer=031

würde also die Ziffernfolge

740031

geschrieben. Liest man die Ziffernfolge als Zahl, so ist das ursprüngliche Sortieren nach den Kriterien Kinderzahl/von auswärts?/Ordnungsnummer gleichbedeutend mit einem aufsteigenden Sortieren der Zahlen. Die Gruppe/Familie mit der kleinsten Zahl erhält also als erste Karten. Da M erklärt, daß er mit den Zahlen leichter zurecht kommt als mit den ursprünglichen Daten, fügt man diese Ergänzung in das alte Verfahren ein.
Die entwickelte Lösung erfüllt nun vollauf alle Rahmenbedingungen.

2.2.5 Formulierung des Verfahrens

Eine Aufgabe fehlt allerdings noch: Wir sollten den Gesamtablauf des Verfahrens so gliedern und aufschreiben, daß fortan in jeder Festspielsaison eine mit der Verteilung der Karten beauftragte ungeübte Hilfskraft das Verfahren bei Vorlage der Beschreibung korrekt durchführen kann.
Eine solche Beschreibung nennt man **Algorithmus**. Ein Algorithmus besteht stets aus zwei Teilen. Im ersten Teil gibt man die Hilfsmittel (**Objekte**) an, die für das Verfahren benötigt werden (z.B. roter Briefkasten, Wartenummern). Der zweite Teil ist die **Handlungsvorschrift**, die festlegt, welche Handlungen in welcher Reihenfolge und an welchen Objekten vorgenommen werden sollen.

Algorithmus für Kartenverteilung

Objektbeschreibung:

1. Briefkasten: (Farbe: rot,
   Aufschrift: "Ausgefüllte Vormerkungen bitte hier einwerfen");
2. Zettelspender;
3. Rolle von Vormerkzetteln nach Ordnungsnummern aufsteigend sortiert, wobei ein Vormerkzettel folgendermaßen strukturiert ist:

   ((oberer Teil:
   (Aufschrift1: "Diesen Abschnitt in den roten Kasten werfen.";
   Kinderfrage: "Wieviele Kinder begleiten Sie? ☐ ";
   Auswärtsfrage: "Kommen Sie von auswärts?
   Bitte ankreuzen. O O
   ja nein";

Ordnungsnummer: Zahl zwischen 001 und 500;
Codenummer: Zahl zwischen 00001 und 1001500);
(unterer Teil:
(Aufschrift: "Diesen Abschnitt abtrennen und aufbewahren.";
Ordnungsnummer: dreiziffrige Zahl zwischen 001 und 500,
die mit der Ordnungsnummer des oberen
Teils übereinstimmt)))

Handlungsvorschrift:

1. Briefkasten an gut sichtbarer Stelle aufhängen;
2. Wartenummernspender mit der Zettelrolle füllen und an gut sichtbarer Stelle aufhängen;
3. Erstmalig zwei Stunden vor Vorstellungsbeginn, danach alle 10 Minuten und zum letzten Mal eine halbe Stunde vor Vorstellungsbeginn tue folgendes:
   3.1 Leere den Briefkasten;
   3.2 Schreibe auf jeden Zettel eine Zahl (die Codenummer). Die Zahl erhältst du folgendermaßen:
      3.2.1 Die ersten Ziffern sind das Ergebnis der Rechnung
      100 minus der auf dem Zettel eingetragenen Kinderzahl;
      3.2.2 Wenn auf die Frage "Kommen Sie von auswärts?" "ja" angekreuzt ist, dann ist die nächste Ziffer eine Null, sonst ist die nächste Ziffer eine Eins;
      3.2.3 Die nächsten Ziffern sind die Ordnungsnummer;
   3.3 Sortiere die Zettel aufsteigend nach diesen Zahlen zu einem Stapel, die kleinste Zahl nach oben;
   3.4 Mische diesen Stapel und den Stapel, der aus den Zetteln besteht, die du bei früheren Briefkastenleerungen geholt hast, zusammen. Verwende den Algorithmus für Mischen;
4. Postiere dich mit dem Zettelstapel an der Kasse;
5. Solange noch Eintrittskarten vorhanden sind, tue folgendes:
   5.1 Rufe die Ordnungsnummer des obersten Zettels aus;
   5.2 Lege den Zettel beiseite;
   5.3 Verkaufe an die Gruppe mit dieser Ordnungsnummer die gewünschte Zahl an Eintrittskarten;
6. Das Verfahren ist beendet.

Algorithmus für Mischen

Objektbeschreibung:

Drei Stapel S1, S2 und S3 von oberen Vormerkzettelabschnitten, wobei ein Abschnitt folgendermaßen strukturiert ist:

(Aufschrift1: "Diesen Abschnitt in den roten Kasten werfen.";
Kinderfrage: "Wieviele Kinder begleiten Sie? ☐ ";
Auswärtsfrage: "Kommen Sie von auswärts?
Bitte ankreuzen. O O
ja nein";
Ordnungsnummer: Zahl zwischen 001 und 500;
Codenummer: Zahl zwischen 00001 und 1001500);

Handlungsvorschrift:
Zwei Zettelstapel S1 und S2 mischst du nach folgender Vorschrift zu einem neuen Stapel S3 zusammen:

M1. Solange auf beiden Stapeln S1 und S2 noch Zettel liegen, tue folgendes:
  M1.1 Nimm von den beiden obersten Zetteln von S1 und S2 den mit der kleineren Codenummer weg;
  M1.2 Füge diesen Zettel hinten an den neuen Stapel S3 an;
M2. Wenn einer der Stapel S1 oder S2 leer ist, dann hänge den anderen komplett hinten an den Stapel S3 an;
M3. Der Stapel S3 ist das Ergebnis des Mischens.

2.3 Analyse des Ergebnisses

Der Algorithmus in 2.2.5 stellt einen in besonderer Weise *strukturierten* Text dar. In der Objektbeschreibung sind alle für das Verfahren erforderlichen Gegenstände aufgeführt. Im Regelfall werden jedoch keine *konkreten* Objekte, sondern nur deren charakteristische Eigenschaften, sog. **Typen** definiert. Erst wenn jemand den Algorithmus ausführt, werden **konkrete** Objekte der geforderten Typen einbezogen.
Beispiel: Man benötigt nur irgendein Objekt vom *Typ* "Briefkasten mit roter Lackierung und entsprechender Aufschrift". Bei Ausführung des Algorithmus an einem bestimmten Vorstellungstermin wird dann ein *konkretes* Objekt des geforderten Typs benutzt, z.B. der runde, etwas zerbeulte rote Briefkasten mit altdeutscher Aufschrift, den Otto Müller gebastelt, lackiert und beschriftet hat.

Jedes bei Ausführung in einem Algorithmus verwendete Objekt ist also eindeutig gekennzeichnet durch seinen Namen, seinen Typ und seine Gestalt oder seinen Wert, dargestellt als Tripel, z.B. (Ordnungsnummer, Zahl, 031).

Ferner erkennt man Objekte, die über den gesamten Algorithmus eine feste Gestalt (z.B. einen festen Wert) haben, und solche, deren Gestalt oder Wert sich ändern kann.
Beispiel: "Aufschrift1" ändert sich im Verlaufe des Algorithmus nicht. Die Codenummer wird durch den Algorithmus geändert. Sie ist anfangs undefiniert und wird später gesetzt.

Definition A:
Objekte, die während der Abarbeitung eines Algorithmus nicht verändert werden können, heißen **Konstanten**. Alle übrigen Objekte sind **Variablen**.

2.3.1 Elementare Bestandteile von Objektbeschreibung und Handlungsvorschrift

Die Objektbeschreibung zerfällt in eine Reihe einfacherer Objekte und Objekttypen (sog. **elementare Objekte** oder **Typen**), wie z.B.:
- das elementare Objekt "Aufschrift1",
- das elementare Objekt "Kinderfrage",
- den elementaren Typ "Zahl zwischen 001 und 500".

Welche Typen als elementar angesehen werden, hängt vom *Sprachniveau* dessen ab, für den die Handlungsvorschrift gedacht ist.
Beispiel: Der Typ "Zahl" ist ein elementarer Typ. Wir halten diesen Typ für so allgemein bekannt, daß er nicht noch weiter zergliedert werden muß, obwohl dies ohne weiteres möglich wäre, z.B.: Zahl ist eine Folge von Ziffern, Ziffer ist eines der Symbole 0,1,2, ...,9.

Die Handlungsvorschrift zerfällt wie die Objektbeschreibung in eine Reihe einfacherer **Anweisungen** (sog. **Elementaranweisungen**), wie z.B.:
- Leere den Briefkasten
- Sortiere die Zettel aufsteigend ...
- Rufe die Ordnungsnummer des obersten Zettels aus.

Welche Anweisungen als elementar angesehen werden, hängt ebenfalls vom Sprachniveau dessen ab, für den die Handlungsvorschrift gedacht ist.

Definition B:
Objekte, Typen und Anweisungen, die bezgl. eines vorgegebenen Sprachniveaus nicht weiter zerlegt werden können, bezeichnet man als **elementar**.

Die elementaren Sprachelemente sind die Grundbausteine zur Konstruktion von Algorithmen. Weitere Bestandteile dieses „Algorithmenbaukastens“ sind die Konstruk-

toren, mit denen man Typen und Anweisungen nach ggf. mehrfacher Anwendung schrittweise zu komplexeren Typen und Anweisungen kombinieren kann. Dieses Bausteinprinzip ist eine fundamentale Idee der Informatik (s. Abschnitt 6) und wird Ihnen während des gesamten Studiums immer wieder begegnen.

2.3.2 Konstruktoren

Elementaranweisungen und -typen stehen nicht zusammenhanglos hintereinander, sondern sie sind durch unterschiedliche **Konstruktoren** miteinander **verknüpft**. Wir unterscheiden im Algorithmus Anweisungs- und Typkonstruktoren, mit denen man Anweisungen zu mächtigeren Anweisungen bzw. Typen zu mächtigeren Typen zusammensetzen kann.

2.3.2.1 Anweisungskonstruktoren

Folgende Konstruktoren sind in der Handlungsvorschrift erkennbar:

1) **Konkatenation**.

   Zwei Anweisungen (nicht notwendig elementar) werden durch Konkatenation (Hintereinanderschreiben, Sequenzbildung) mittels des Konstruktorsymbols ";" zu einer neuen Anweisung. Die Konkatenation legt zugleich die Reihenfolge fest, in der die Anweisungen ausgeführt werden müssen. Insbesondere folgt daraus, daß eine Handlungsvorschrift in der ersten Zeile beginnt.

   Beispiel: 5.1 Rufe die Ordnungsnummer des obersten Zettels aus;
   5.2 Lege den Zettel beiseite

2) **Iteration**.

   Dieser Konstruktor sorgt dafür, daß eine oder mehrere Anweisungen mehrfach nacheinander ausgeführt werden. Bei der einen Form des Konstruktors wird eine Anweisung eine vorher festgelegte Anzahl oft durchlaufen, bei der anderen Form liegt diese Anzahl nicht fest. Vielmehr sind hier eine oder mehrere Anweisungen so oft auszuführen, wie ein bestimmter Sachverhalt vorliegt. Ob und welcher Sachverhalt vorliegt, wird durch eine *Bedingung* (die sog. *Abbruchbedingung*) überprüft. Beispiele für Bedingungen sind z.B. in Anweisung 5 "noch Eintrittskarten vorhanden" oder in Anweisung M1 "auf beiden Stapeln noch Zettel liegen". Konstruktoren, die das mehrfache Ausführen einer oder mehrerer Anweisungen vorschreiben, bezeichnet man allgemein als Schleifen. Schleifen der ersten Form nennt man Zählschleifen, Schleifen der zweiten Form bedingte Schleifen.

   Beispiele: In 3.: Erstmalig ... danach alle ... zum letzten Mal ... tue folgendes ...,
   In 5. und M1.: Solange ... tue folgendes ... .

Definition C:
Eine **Schleife** ist eine Folge von Anweisungen, die mehrfach durchlaufen werden kann.
Bei der **Zählschleife** wird die Anzahl der Schleifendurchläufe durch einen Zähler bestimmt, der beginnend bei einem **Startwert** in jedem Durchlauf bis zum **Endwert** um einem festen Wert, die **Schrittweite**, erhöht oder erniedrigt wird.
Die **bedingte Schleife** wird so oft durchlaufen, wie eine **Bedingung** (die **Abbruchbedingung**) erfüllt (oder nicht erfüllt) ist.
Ein Konstruktor zur Konstruktion von Schleifen heißt **Iterationskonstruktor**.

3) **Alternative**.
Bei diesem Konstruktor hängt die Ausführung von Anweisungen von einer Bedingung ab. Man bezeichnet eine durch den Konstruktor Alternative erzeugte Anweisung als bedingte Anweisung.
Beispiel: In 3.2.2: Wenn ... dann ... sonst ... .

Definition D:
Eine **bedingte Anweisung** ist eine Anweisung, deren Ausführung von einer Bedingung abhängt.
Eine bedingte Anweisung erhält man durch Anwendung des Konstruktors **Alternative**.

4) **Abstraktion**.
Dieser Konstruktor erlaubt die Verwendung eines Algorithmus A1 (*Prozedur*) als Elementaranweisung innerhalb eines Algorithmus A2. A1 wird dabei in A2 *aufgerufen* und von A2 mit Eingaben (sog. *Parameter*) versorgt. Umgekehrt liefert A1 seine Ausgaben als Ergebnisse an A2 zurück.

Beispiel:
> 3.4 Mische diesen Stapel und den Stapel, der aus den Zetteln besteht, die du bei früheren Briefkastenleerungen geholt hast, zusammen. Verwende den Algorithmus für Mischen;

"Mischen" ist hier der Name der Prozedur, "dieser Stapel" und "der Stapel, der ..." sind die Parameter, die im Mischalgorithmus mit S1 und S2 assoziiert werden. S3 ist das Ergebnis der Prozedur.

Definition F:

Aufruf nennt man die Verwendung eines Algorithmus A1 in einem anderen Algorithmus A2. Der Aufruf eines Algorithmus besteht in der Regel aus der Nennung seines Namens und seiner Parameter. A1 nennt man **Prozedur** oder **Funktion**.
Prozedur und Funktion entstehen durch Anwendung des **Abstraktionskonstruktors**.

Zerlegt man einen Algorithmus in mehrere Prozeduren, die jeweils wiederum in Prozeduren zerlegt werden, so erhält man eine *Hierarchie* von Prozeduren. Diese Form der Strukturierung eines Algorithmus ist das zentrale Vorgehen zur überschaubaren Konstruktion größerer Algorithmen. Anschaulich betritt man durch die Definition von Prozeduren und Funktionen eine neue *Abstraktionsebene*, indem man sich neue Elementaroperationen schafft,

2.3.2.2 Typkonstruktoren

Da die Objektbeschreibung eines Algorithmus die Gegenstände der realen Welt und ihre vielfältige Struktur beschreibt, kann man den Aufbau der Objekte meist nicht auf einige wenige feste Konstruktoren zurückführen. Vielmehr ist hier nahezu alles möglich, was die Wirklichkeit zweckmäßig erfaßt. Später, wenn wir Algorithmen durch maschinell ausführbare Programme darstellen, werden wir uns jedoch auf einige wenige universell verwendbare Konstruktoren beschränken, mit denen man die gesamte Realität angemessen modellieren kann. Einige dieser Konstruktoren deuten sich bereits im Algorithmus an, allerdings recht verschwommen:

1) **Aggregation**.
   Man kann beliebige Typen oder Objekte geordnet miteinander zu Paaren. Tripeln, Quadrupeln etc. verknüpfen und einen neuen Typ (ein neues Objekt) bilden mit den ursprünglichen Typen als "Untertypen" (sog. Tupelbildung) (Abb. 4).
   Beispiele: Die Objekte "Farbe" und "Aufschrift" als Paar zusammengenommen bilden *ein* Merkmal des Briefkastens. "Unterer Teil" eines Zettelabschnitts wird gebildet aus dem Paar (Aufschrift1, Ordnungsnummer).

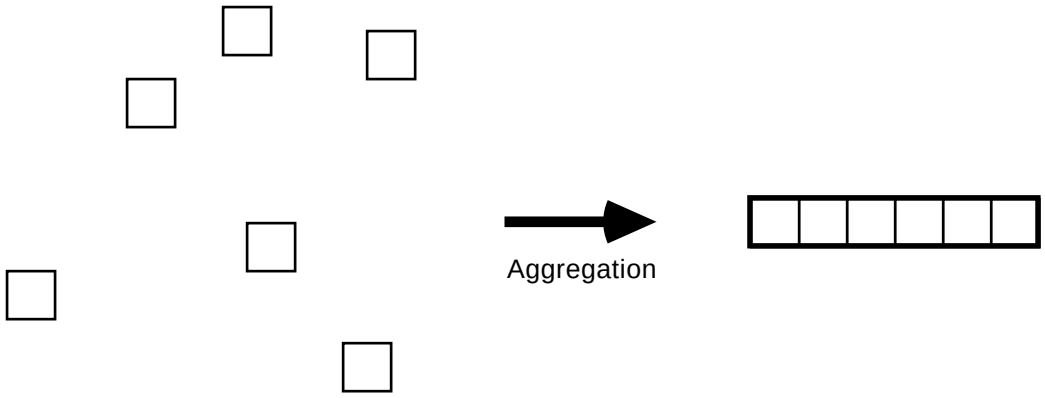


Abb. 4: Aggregation

2) **Iteration**.
Typen und Objekte kann man nicht nur endlich oft miteinander verknüpfen wie bei der Aggregation, sondern prinzipiell unbeschränkt. Zu den Objekten eines iterierten Typs gehören dann alle Objekte, die durch beliebige Aneinanderreihung von Objekten des zugrundeliegenden Typs gebildet werden können (Abb. 5).
Beispiel: Mit dem Sprachelement "Rolle von" bildet man beliebig lange Folgen von Vormerkzetteln.

Abb. 5: Iteration

2.4 Algorithmus und Prozeß

Wie kommt man von einem Algorithmus, d.h. von einer Handlungs*vorschrift*, zu den Handlungen selbst? Man benötigt eine Person oder ein Gerät, die bzw. das den Algorithmus *lesen*, *verstehen* und *ausführen* kann. Einen solchen Ausführenden bezeichnet man als **Prozessor**.

Im einzelnen muß ein Prozessor folgende Fähigkeiten besitzen:

1) Er muß den Algorithmus *lesen* können (als Mensch mit den Augen, als Gerät mit Hilfe eines Lesegerätes).
2) Er muß den Algorithmus *verstehen*, d.h., er muß die Bedeutung (sog. *Semantik*) der Elementaranweisungen kennen und jede zugehörige Handlung ausführen können. Der Prozessor muß ferner die Bedeutung der Konstruktoren kennen und durch *Interpretation* bei der Ausführung des Algorithmus die korrekte Abfolge der Elemen-

taranweisungen generieren. Diese Folge ist nicht immer gleich. Sie hängt vielmehr von der Ausgangssituation und von den Bedingungen innerhalb der Konstruktoren ab.

Beispiel: Wir erinnern uns, daß der Verteiler M den Anforderungen an einen Prozessor zunächst nicht gerecht wurde, denn er kannte die Semantik der Elementaranweisung "mischen" nicht

Definition F:
Bei Ausführung eines Algorithmus sind Bedingungen entweder **erfüllt** (**wahr**, engl.: **true**) oder **nicht erfüllt** (**falsch**, engl.: **false**).

Ein und derselbe Algorithmus kann also bei Ausführung zu unterschiedlichen Handlungsabfolgen führen. Eine *konkrete* Handlungsabfolge bezeichnet man als **Prozeß**. In diesem Sinne beschreibt ein Algorithmus also eine Menge verschiedener Prozesse.

Definition G:
Ein **Prozeß** ist eine Abfolge von Handlungen, die von einem Prozessor durchgeführt und von einem Algorithmus kontrolliert wird.

2.5 Eigenschaften von Algorithmen

In Vorlesungen über Theoretische Informatik wird der Begriff des Algorithmus mathematisch exakt definiert. Wir wollen uns hier mit einigen charakteristischen Eigenschaften begnügen.

Eigenschaft 1: Ein Algorithmus löst ein Problem. Im allgemeinen erfaßt ein Algorithmus aber nicht nur ein einzelnes Problem, sondern eine ganze **Problemklasse** und liefert zu jedem Problem dieser Klasse eine richtige Lösung. Die Auswahl eines einzelnen Problems der Klasse erfolgt durch Wahl der **Eingabedaten** (**Parameter**).
Beispiel: Unser Verteilungsalgorithmus löst nicht nur das Einzelproblem "Wie verteile ich bei der Vorstellung am 16.7. die nicht abgeholten 98 Eintrittskarten auf die 146 wartenden Besucher?", sondern allgemein jedes Problem der Problemklasse "Wie verteile ich x Eintrittskarten auf y Besucher?". Die Eingabedaten oder Parameter unseres Algorithmus sind die Zettelabschnitte, die wartende Besucher ausgefüllt haben, und die nicht abgeholten Eintrittskarten.

Mathematisch ausgedrückt beschreibt (man sagt: **berechnet**) ein Algorithmus P eine Funktion

$f_P: E \rightarrow A$

von der Menge der Eingabedaten E in die Menge der Ausgabedaten A. Als Funktionswert $f_P(e)$ faßt man alle die Daten auf, die der Algorithmus auf die Eingabe $e \in E$ geliefert hat, nachdem sein Ende erreicht ist. Insbesondere besagt der funktionale Charakter eines Algorithmus, daß es zu jedem Eingabewert $e \in E$ *höchstens* einen Ausgabewert $a=f_P(e) \in A$ gibt.

Liefert ein Algorithmus auf eine Eingabe $e \in E$ zwar Ausgabedaten, erreicht aber sein Ende nicht, so sagt man: Der Algorithmus ist für die Eingabe e nicht definiert, Schreibweise: $f_P(e)=\perp$. Kommt ein Algorithmus für eine Eingabe $e \in E$ zwar zum Ende, liefert aber keine Ausgabe, so ist der Algorithmus für e dennoch definiert, und man schreibt $f_P(e)=\varepsilon$. ε symbolisiert die leere Ausgabe (das **leere Wort**).

Beispiel: Im Verteilungsproblem ist E die Menge aller möglichen ausgefüllten oberen Zettelabschnitte, dargestellt durch eine Folge von Tripeln Kinderzahl/von auswärts?/Ordnungsnummer, und aller nicht abgeholten Eintrittskarten und A die Menge aller den Sortierkriterien genügenden Reihenfolgen dieser Zettel, ausgedrückt durch eine Folge von Ordnungsnummern.

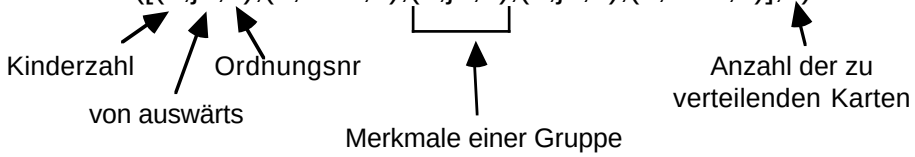
Formal: $E=(\{1,...,100\} \times \{ja,nein\} \times \{1,...,500\})^* \times IN$,

$A=\{1,...,500\}^*$.

Für eine Menge M symbolisiert M^* dabei die Menge aller endlichen Folgen, die man mit Elementen von M bilden kann.

Für die Eingabe $e \in E$ mit

e=([(2,ja,1),(3,nein,2),(2,ja,3),(1,ja,4),(4,nein,5)],9)



gilt dann

$f_P(e)=[5,2,1]$.

Die Gruppen erhalten die 9 Eintrittskarten also in der Reihenfolge Gruppe 5, Gruppe 2, Gruppe 1. Die Gruppen 3 und 4 gehen leer aus.

In neueren Ansätzen hat man die Forderung nach Funktionalität des Algorithmus teilweise aufgegeben. Bei diesen Algorithmen nimmt man bewußt in Kauf, daß sie unter gleichen Eingaben verschiedene Ergebnisse (auch falsche) liefern. Bei speziellen (den sog. *stochastischen*) Algorithmen verlangt man aber, daß sie nicht zu häufig (weniger als 50%) falsche Resultate ausgeben. Diese Algorithmen werden z.B. dann verwendet, wenn exakte Lösungsverfahren zu aufwendig oder nicht bekannt sind.

Eigenschaft 2: Die Beschreibung eines Algorithmus hat eine endliche Länge (**statische Finitheit**). Anderenfalls kann man den Algorithmus gar nicht aufschreiben und auch niemandem mitteilen.

Eigenschaft 3: Algorithmen sind tatsächlich durchführbar (**Effektivität**). Alle Einzelschritte sind verständlich formuliert und verlangen "nichts Unmögliches". Jeder Einzelschritt läßt sich in endlicher Zeit bearbeiten. An jeder Stelle bei der Ausführung des Algorithmus ist klar, was als nächstes zu tun ist. Dies schließt aber nicht aus, daß der Algorithmus an bestimmten Stellen Wahlmöglichkeiten läßt. Nur muß dann genau festliegen, wie die Auswahl einer Möglichkeit vonstatten gehen soll.

Vor allem die dritte Eigenschaft zeigt bereits einige Schwächen dieser Charakterisierung. Was heißt denn nun "tatsächlich durchführbar" oder "verständlich formuliert"? Der Begriff "Mischen" war mißverständlich und bedurfte der Erläuterung, das haben wir bereits gesehen. Kann unser Algorithmus aber nun von jedem ausgeführt werden? Auch von einem kleinen Kind? Im allgemeinen wird man einen Algorithmus für tatsächlich durchführbar und verständlich formuliert halten, wenn eine genügend große Zahl von Menschen dies bestätigt. Auf die angeschnittene Problematik, die sich hinter dieser Unpräzision verbirgt, wird in Vorlesungen über Theoretische Informatik genauer eingegangen.

Eigenschaft 1 besagt, daß ein Algorithmus ein Problem löst. Meist ist das Problem gelöst, wenn der Ausführende ein Resultat ermittelt und das Ende des Algorithmus erreicht ist. Einen Grenzfall bildet z.B. ein Algorithmus zur Steuerung einer Verkehrsampelanlage, der zwar unbegrenzt lange ablaufen soll, dessen Ablauf man jedoch in einzelne Aufrufe eines Algorithmus unterteilen kann, die jeweils nur endlich lange ablaufen. Daher:

Eigenschaft 4: Für die Praxis sind meist nur **korrekte** Algorithmen von Bedeutung, die das gewünschte Resultat liefern und zum Ende kommen (**dynamische Finitheit** oder **Terminierung**). Terminiert ein Algorithmus, der die Funktion $f: E \rightarrow A$ berechnet, für eine Eingabe $e \in E$ nicht, so ist f für e **undefiniert**; Schreibweise: $f(e)=\perp$. f ist dann eine **partielle** (d.h. nicht immer definierte) Funktion. Ist f für alle Eingaben definiert, d.h. terminiert der Algorithmus für alle Eingaben, so heißt f **totale** Funktion.

Überlegen wir noch einmal, wie wir vorgegangen sind, um eine Lösung für unser Verteilungsproblem zu entwickeln. *Einen* Algorithmus hatten wir sehr schnell gefunden.

Da dessen Ausführung aber die zeitlichen Vorgaben sprengte, war das Verfahren für die Praxis unbrauchbar. Der verbesserte Algorithmus kann wesentlich schneller und von einer einzigen Person durchgeführt werden, und es werden weniger sonstige Hilfsmittel (z.B. Notizpapier) benötigt. Dieser Algorithmus ist also kostengünstiger als der anfangs vorgeschlagene. Dies führt zu folgender

Eigenschaft 5: Praktisch interessierende Algorithmen müssen **effizient** sein, d.h., ein vorgegebenes Problem in möglichst kurzer Zeit und mit möglichst geringem Aufwand an Hilfsmitteln (sog. **Betriebsmittel**) lösen. Man sagt: Sie müssen eine möglichst geringe **Komplexität** besitzen.

Nicht immer kann man (wie in unserem Beispiel) den Aufwand an Betriebsmitteln gleichmäßig vermindern. Bei vielen Algorithmen kann man häufig ein erforderliches Hilfsmittel beliebig verringern, benötigt dann aber z.B. ständig mehr Zeit und umgekehrt. Man sagt dann: *Die Betriebsmittel konkurrieren miteinander*.

2.6 Zusammenfassung

Lassen wir die Inhalte dieses Abschnitts noch einmal Revue passieren:

- Problemstellung: Eintrittskartenverteilung; Rahmenbedingungen: Zeitvorgabe, Zwang, die Lösung exakt und so einfach zu formulieren, daß sie von ungeübten Hilfskräften nachvollzogen werden kann (*Sprachniveau*)
- erster Lösungsansatz unter der Maxime: *Probieren*
- Ergebnis: der Lösungsansatz ohne angemessene Vorüberlegung trägt nicht und sprengt fast alle Rahmenbedingungen
- zweiter Lösungsansatz unter der Maxime: *Präzisieren und Zerlegen*
- genauere *Spezifikation* des Problems
- *Zerlegung* des Problems in überschaubare Teile
- *Analyse* und Verbesserung der Teilprobleme durch elegantere Modellierung und präzisere Beschreibung der Abläufe
- ggf. weitere *Zerlegung* der Teile bis auf elementare Einheiten, sofern *Komplexität* des Problems oder *Sprachniveau* des Prozessors dies erfordern (Stichwort: Mischen)
- Ergebnis: *effizienterer* und *fehlertoleranterer* Algorithmus, der die Problemspezifikation voll erfüllt
- dritter Lösungsansatz: *Optimieren*
- Ergebnis: Erhöhung der *Sicherheit* des Algorithmus durch eine bessere *Datenstruktur* (Stichwort: Codierung der Daten)

- schriftliche Fixierung des Algorithmus, bestehend aus *Objektbeschreibung* und *Handlungsvorschrift*
- beliebig häufige Ausführung des Algorithmus durch einen *Prozessor* (oder auch mehrere), der durch Interpretation der Anweisungen eine Abfolge von Handlungen an konkreten Objekten (einen *Prozeß*) erzeugt.

Diese hier skizzierte Vorgehensweise, die wir durch mehr oder weniger naive Überlegungen gewonnen haben, enthält bereits im Kern die zentrale Methode bei der Entwicklung von Algorithmen, allgemein bei der Bewältigung informatischer Probleme.

Bevor wir die Probleme untersuchen, die entstehen, wenn wir in das Verteilungsverfahren einen Computer einbinden möchten, analysieren wir zunächst die Grenzen solch eines algorithmischen Vorgehens. Schon die bisherigen informellen Überlegungen führen hier zu verblüffenden Einsichten.