**20. Bundeswettbewerb Informatik 2001/2002, 2. Runde**

# Lösungshinweise

## Allgemeines

**Bewertungsbögen** Kein Kreuz in einer Zeile bedeutet, dass der entsprechende Aufgabenteil korrekt bearbeitet wurde. Häufig wurde nur das vermerkt, was von der korrekten Bearbeitung abweicht. Ein Kreuz in der Spalte „+“ bedeutet in der Regel Zusatzpunkte, ein Kreuz unter „ok“ bedeutet eine gute Lösung im Rahmen der korrekten Bearbeitung (also meist ohne Zusatzpunkte; für manche Dinge gab es bestenfalls ein „ok“) und ein Kreuz unter „–“ bedeutet Minuspunkte für Fehlendes oder Unzulängliches.

**Termin der 2. Runde** Es vermerken immer wieder Teilnehmer, dass die 2. Runde parallel zum Abitur liegt. Das ist uns bekannt und sicher nicht ideal, lässt sich aber leider nicht ändern. In der 2. Jahreshälfte läuft die 2. Runde des Mathewettbewerbs, dem wir keine Konkurrenz machen wollen. Also bleibt uns nur die erste Jahreshälfte. Und damit liegt der Abgabetermin der 2. Runde immer in der Zeit der Abiturtermine. Aber: Sie haben etwa vier Monate Bearbeitungszeit für die 2. Runde. Rechzeitig mit der Bearbeitung der Aufgaben zu beginnen ist der beste Weg, Konflikte mit dem Abitur zu vermeiden.

**Dokumentation** Eine Dokumentation beginnt nicht mit: „Die Prozedur abc übergibt einen Zeiger p auf ein Feld xy, worauf die Funktion f ...“. Wie in den allgemeinen Hinweisen zu den Aufgaben gesagt, sollte die Dokumentation mit der Idee beginnen, die Sie zur Lösung der jeweiligen Aufgabe entwickelt haben. Schildern Sie diese Lösungsidee erst grob und gehen dann darauf ein, wie Sie sie in ein abstraktes, computertaugliches Modell (in Form von Algorithmen und Datenstrukturen) umgesetzt haben. Ihre Implementierung dieses Modells als Programm beschreiben Sie anschließend in der Programmdokumentation, die die wichtigsten Funktionen, Variablen, Klassen, Objekte etc. in Bezug auf die Lösungsidee dokumentiert und idealerweise mitteilt, wo diese Komponenten im Quellcode zu finden sind. Beachten Sie: Wer nicht in der Lage ist, Idee und Modell präzise zu formulieren, bekommt auch keine saubere Umsetzung in welche Programmiersprache auch immer hin.

**Lösungshinweise** Bei den folgenden Erläuterungen handelt es sich um Vorschläge, nicht um die einzigen Lösungswege, die wir gelten ließen. Wir akzeptieren in der Regel alle Ansätze, die die gestellte Aufgabe vernünftig lösen und entsprechend dokumentiert sind. Einige Dinge gibt es allerdings, die – unabhängig vom gewählten Lösungsweg – auf jeden Fall diskutiert werden müssen.

# Aufgabe1: Stimmt die Chemie?

Die „BWINF-Chemie“ ist ein sehr grobes chemisches Modell und kann schon einfache Verbindungen wie Stickstoffmonoxid und -dioxid ($NO, NO_2$) nicht erklären. Bei dieser Aufgabe geht es aber auch weniger um die chemischen Zusammenhänge und eine möglichst exakte Beschreibung derselben, als um die dahinter steckenden informatischen Probleme. Bei der Bewertung gab es Abzüge für chemische Unkorrektheiten daher nur, wenn diese eindeutig im Widerspruch zur Aufgabenstellung standen. Erweiterungen der BWINF-Chemie bzw. deutliche Abweichungen waren erlaubt (eine ganze Reihe von Teilnehmern haben z.B. mit den Elektronegativitätswerten der Elemente gearbeitet), sie mussten aber gut dokumentiert und auch begründet sein. Erweiterungspunkte gab es nur für Abweichungen, die auf der Informatikseite das Problem komplexer machen.

## Datenstruktur zur Darstellung von Reaktionsgleichungen

Am natürlichsten lassen sich Reaktionsgleichungen (ähnlich wie mathematische Ausdrücke) im Computer in einer baumartigen Struktur darstellen. Das muss nicht einmal ein richtiger Syntaxbaum sein: alles, was dieselbe Information enthält und sich einfach weiterverarbeiten lässt, ist eine geeignete Datenstruktur. In der Präfix-Notation von Lisp könnte man die Reaktion $2H_2 + O_2 \rightarrow 2H_2O$ etwa folgendermaßen darstellen:

```
(rg (summe (term 2 (h . 2)) (term 1 (o . 2)))
    (summe (term 1 (h . 2) (o . 1))))
```

In anderen Programmiersprachen lassen sich gleichwertige Konstruktionen mit Hilfe verketteter Strukturen bzw. Records realisieren. Obige Darstellung vereinfacht die nachfolgende Bearbeitung noch dadurch, dass die Einsen, die in der Gleichung nicht geschrieben werden, explizit enthalten sind. Die Reaktionsgleichungen aus der Aufgabenstellung haben eine einfache Struktur und können daher ohne großen Aufwand in eine solche interne Darstellung übersetzt werden.

In der Aufgabenstellung war nicht vorgegeben, welche Teile einer Reaktionsgleichung durch Fragezeichen ersetzt werden dürfen. Das verlangte Beispiel enthält zwei verschiedene Fehlstellen:

$$? H_2 + O_2 \rightarrow ?$$

Links fehlt ein Faktor $2$ und rechts das komplette Produkt $2H_2O$. In der Notation von oben ließe sich das so darstellen:

```
(rg (summe (term ? (h . 2)) (term 1 (o . 2)))
    (summe ?))
```

Um die vorgegebene Reaktionsgleichung ergänzen zu können, müssen die Einsendungen also zumindest Koeffizienten und Terme, also Moleküle mit Vorfaktor, ergänzen können. Das andere Extrem ist, wenn die Fragezeichen für einzelne Buchstaben stehen können, N? also auch zu Na

ergänzt werden kann. Wegen der Vielzahl der Möglichkeiten muss gut dokumentiert werden, wofür die Fragezeichen in einer konkreten Lösung stehen können. Einige Teilnehmer haben die Fragezeichen mit Zusatzinformationen angereichert (mit der Begründung, dass Maggie Moder ja auch z.B. die Größe einer Lücke erkennen kann). Meist waren aber die Zusatzinformationen so detailliert, dass das Füllen der Lücken deutlich vereinfacht wurde. Das kann natürlich nicht belohnt werden.

## Ein- und Ausgabe von Reaktionsgleichungen

Zwischen chemischer Notation und interner Repräsentation steht noch die Ein- und Ausgabe. Hierfür kann eine grafische Schnittstelle sorgen, einfacher sind allerdings Textformate – reiner Text, wohlgemerkt: Sich bei Ein- und Ausgabe auf Textformatierung und z.B. Tiefstellung der Subskripte zu konzentrieren, bedeutet Aufwand an der falschen Stelle zu betreiben. Die einfachste Eingabeform ist etwas wie `2H2+O2>2H2O`. Viel kompliziertere Eingabeformate sollten dem Benutzer des Programms nicht abverlangt werden; auch eine Hobby-Geschichtsforscherin soll damit klar kommen.

Abschließend muss das Programm natürlich in der Lage sein, intern gespeicherte Reaktionsgleichungen wieder in lesbarer Form auszugeben. Auch hier gilt, dass das Ergebnis der chemischen Schreibweise ähneln sollte. Sinnvollerweise ist das Ausgabeformat mit dem Eingabeformat identisch; nur Fragezeichen sollten nicht mehr vorkommen. Spätestens bei der Ausgabe sind zudem die in der Aufgabe vorgegebenen Regeln (Geber vor Nehmer, aber OH-Gruppe als OH notiert) zu berücksichtigen – soweit nicht andere, eigene Vereinbarungen getroffen waren.

## Lösung der Reaktionsgleichungen

Der zweite Teil der Aufgabe bestand darin, Reaktionsgleichungen mit Fragezeichen sinnvoll zu ergänzen. Wer es vermeiden will, alle überhaupt möglichen Reaktionsgleichungen zu erzeugen und zu überprüfen, muss an dieser Stelle Annahmen zur allgemeinen Struktur von Reaktionsgleichungen und Fehlstellen treffen (und nach Möglichkeit auch erläutern). Einige Beispiele:

- Nur Terme und Faktoren dürfen fehlen.
- Alle an der tatsächlichen Reaktion beteiligten Atomarten sind auf mindestens einer Seite der Gleichung noch lesbar.
- Jede auftretende Molekülart kommt nur einmal in der Gleichung vor (keine Katalysatoren).

Natürlich sind andere Strategien und Heuristiken denkbar (etwa: neben den auftretenden Atomarten können auch H, C, N, O eingebaut werden). Ganz ohne Einschränkungen dieser Art ist die Aufgabe nur schwer lösbar: die Anforderungen an die Lösungen sind so komplex, dass es schwierig ist, den Lösungsraum systematisch zu durchsuchen.

Die folgenden Abschnitte enthalten Lösungsideen für einige Sonderfälle des allgemeinen Problems. Durch Kombination der einzelnen Schritte kann man zu einer Gesamtlösung kommen, die etwa wie folgt verfährt:

**do**
⟨ergänze fehlende Molekülrümpfe (ohne Koeffizienten und Indizes)⟩
⟨ergänze fehlende Indizes⟩
⟨ergänze fehlende Koeffizienten⟩
**while** ⟨weitere Gleichungen generierbar⟩

### Ergänzen fehlender Koeffizienten

Dieses Problem ist vielleicht aus dem Chemie-Unterricht bekannt: es ist das berüchtigte „Einrichten von Reaktionsgleichungen“. Mathematisch lässt es sich auf die Lösung eines *linearen Gleichungssystems* zurückführen. Wie das funktioniert, sei hier anhand der Gleichung

$$\alpha_1(\mathrm{H_2}) + \mathrm{O_2} \rightarrow \beta_1(\mathrm{H_2O})$$

erklärt. Die Variablen $\alpha_1, \beta_1$ stehen für die unbekannten Vorfaktoren in der Gleichung. Die einzelnen Atomsorten müssen auf beiden Seiten der Reaktionsgleichung gleich oft auftreten, also müssen $\alpha_1$ und $\beta_1$ folgende Gleichungen erfüllen

$$\alpha_1 \left(\begin{array}{c|c} \mathrm{H} & 2 \\ \mathrm{O} & 0 \end{array}\right) + \left(\begin{array}{c|c} \mathrm{H} & 0 \\ \mathrm{O} & 2 \end{array}\right) = \beta_1 \left(\begin{array}{c|c} \mathrm{H} & 2 \\ \mathrm{O} & 1 \end{array}\right),$$

oder, anders geschrieben,

$$\begin{aligned} 2\alpha_1 - 2\beta_1 &= 0 \quad \text{(H)}, \\ -\beta_1 &= -2 \quad \text{(O)}. \end{aligned}$$

In diesem Fall ist die Lösung direkt ablesbar: $\beta_1 = 2$, und daher auch $\alpha_1 = 2$.

Das einfachste allgemeine Verfahren zur Lösung solcher Gleichungssysteme ist der Gauß-Algorithmus. Seine genaue Funktionsweise wird zum Beispiel im Schülerduden Mathematik oder in R. Sedgewick, Algorithmen (1991) erklärt.

Bei diesem Lösungsansatz treten zwei zusätzliche Schwierigkeiten auf. Zum einen sind in Reaktionsgleichungen nur natürliche Zahlen als Koeffizienten erlaubt, während der Gauß-Algorithmus üblicherweise mit reellen Zahlen rechnet. Das kann man aber dadurch beheben, dass man beim Lösen der Gleichungen durchweg rationale Zahlen verwendet und die Reaktionsgleichung am Ende mit dem kleinsten gemeinsamen Vielfachen der Nenner durchmultipliziert.

Die zweite Schwierigkeit ist, dass die auftretenden Gleichungssysteme üblicherweise *unterbestimmt* sind, dass es also mehr Unbekannte als Gleichungen gibt. In diesem Fall gibt es prinzipiell unendlich viele Lösungen. Da uns aber nur die *eine* Lösung mit den niedrigsten natürlichen

Koeffizienten interessiert, kann man einfach so viele Variablen $\alpha_i$ oder $\beta_i$ auf Eins setzen, dass genauso viele Unbekannte wie Gleichungen übrig bleiben. Zum Beispiel ist

$$\alpha_1 \begin{pmatrix} \mathrm{H} \\ \mathrm{O} \end{pmatrix} \begin{pmatrix} 2 \\ 0 \end{pmatrix} + \alpha_2 \begin{pmatrix} \mathrm{H} \\ \mathrm{O} \end{pmatrix} \begin{pmatrix} 0 \\ 2 \end{pmatrix} = \beta_1 \begin{pmatrix} \mathrm{H} \\ \mathrm{O} \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix},$$

unterbestimmt. Setzt man aber (beliebig) $\beta_1 = 1$, so folgt $\alpha_1 = 1, \alpha_2 = 1/2$, also nach dem Durchmultiplizieren mit dem kgV der Nenner wieder die bekannte Lösung.

Wenn sich eine Gleichung *nicht* mit natürlichen kleinen (!) Zahlen einrichten lässt, ist sie mit großer Wahrscheinlichkeit keine chemisch korrekte Lösung, da sich chemische Gleichungen gerade durch solche Koeffizienten auszeichnen. Solche Lösungen können also verworfen werden.

Es gibt natürlich Alternativen zum Gauß-Verfahren. Eine Möglichkeit ist, per Backtracking verschiedene Kombinationen von Koeffizienten auszuprobieren. Dabei kann man wiederholt den Koeffizienten eines Terms erhöhen, in dem ein Atom vorkommt, das auf der anderen Seite häufiger auftritt. (Es kann mehrere solche Terme geben, also funktioniert ein iterativer Ansatz nicht.) Den Suchraum sollte man aber auch hier auf kleine Koeffizienten (etwa $\leq 6$) beschränken.

### Ergänzen fehlender Terme

Die Terme auf den Gleichungsseiten bestehen aus einem Faktor und einem Molekül. Wenn wir das Molekül richtig „raten“ können, reduziert sich das Problem auf die Bestimmung fehlender Koeffizienten aus dem vorigen Abschnitt. Der Einfachheit halber nehmen wir an, dass nur ein Term in der Gleichung fehlt.

Die Molekülerzeugung kann man in zwei Schritte zerlegen: zunächst entscheidet man sich, welche Atomarten in einem Molekül vorkommen sollen und berechnet dann im zweiten Schritt, wie oft jedes Atom vorkommen muss, damit genauso viele Elektronen abgegeben wie aufgenommen werden.

Ist also $A = \{a_1, \ldots, a_n\}$ die Menge der im fehlenden Molekül (möglicherweise) vorkommenden Atomsorten, kann man die Molekülrümpfe generieren, indem man geeignete Teilmengen der Menge $A$ erzeugt. Es macht dabei aber keinen Sinn, allzu große Teilmengen zu betrachten, denn bei Molekülen aus vielen verschiedenen Atomsorten bestimmt auch die räumliche Anordnung der Atome deren Bindungsverhalten (also die Abgabe oder Aufnahme von Elektronen). Das Bindungsmodell der Aufgabe berücksichtigt das überhaupt nicht, also ist es angemessen, nur Moleküle aus höchstens drei unterschiedlichen Atomsorten zu betrachten. Selbst mit dieser Einschränkung ist die Anzahl der möglichen Molekülrümpfe immer noch von der Ordnung $O(n^3)$. Man sollte die Menge $A$ also trotzdem so klein wie möglich halten.

### „Einrichten“ von Molekülen / Ergänzen fehlender Indizes

Aus einer Atomkombination wie $\{\mathrm{C,O}\}$ kann man verschiedene gültige Moleküle zusammenbauen, in diesem Fall sind das $\mathrm{CO_2}$ und $\mathrm{CO}$. Bei diesem „Zusammenbauen“ macht man eigentlich nichts anderes, als die Lücken„gleichung“ $\mathrm{C_?O_?}$ zu lösen.

Für *eine* Atomsorte ist das einfach: Die Aufgabenstellung gibt bereits vor, dass die Elemente H, N, O, F, Cl meist als zweiatomige Moleküle auftreten. Für diese wählt man also das Subskript 2, für alle anderen die 1.

Schwieriger ist die Sache schon bei *zwei* Atomsorten. Hier muss zunächst entschieden werden, welches der Atome als Geber und welches als Nehmer fungiert. Die Aufgabenstellung ist hier bewusst ungenau, weshalb jeder Teilnehmer das Kriterium in irgendeiner Weise präzisieren muss. Wer hier eine Elektronegativitäts-Tabelle aus seinem Chemiebuch abtippt, ist auf der sicheren Seite, ein einfaches Kriterium wie das folgende ist aber natürlich auch in Ordnung: Sind A und B die beiden Atome, dann ist A der Nehmer, falls

- A *mindestens* so weit rechts und oben steht wie B oder
- A in der „Nehmerhälfte“ der Tabelle liegt und B nicht.

Atomkonfigurationenen, für die diese Regeln keine Entscheidung bringen, müssen nicht weiter untersucht werden, solange das in der Dokumentation erläutert wird.

Für mehr als zwei Atomsorten reicht eine Heuristik, die die gebräuchlichsten Moleküle erklären kann und die Atome so arrangiert, dass sie die Vorgaben der BWINF-Chemie bzw. der eigenen Ergänzungen nicht verletzen. Zum Beispiel: Das Molekül enthält nur einen Nehmer, die beiden anderen Atome sind automatisch Geber (wie in NaOH, wo der Sauerstoff zwei Elektronen aufnimmt). Die getroffene Regelung sollte vernünftig sein; wie immer ist aber eine gute Beschreibung noch wichtiger.

Weiß man, welche Atome im Molekül Geber bzw. Nehmer sind, kann man alle gültigen Indexkombinationen erzeugen. Dabei sollte man natürlich Lösungen unterdrücken, bei denen die Indizes einen ggT $> 1$ haben (also etwa $\mathrm{C_2O_4}$).

## Bewertung von Gleichungen

Auch wenn ein Programm nur syntaktisch und (BWINF-Chemie-)semantisch korrekte Lösungen bestimmt, gibt es für eine Bewertung noch andere Möglichkeiten, als z.B. nur einen Prozentsatz für die Plausibilität der Lösung anzugeben (etwa 25%, wenn 4 Lösungen gefunden wurden). Am naheliegendsten ist es, die Kürze oder Kompaktheit von Reaktionsgleichungen zum Maßstab zu nehmen: möglichst wenig Atome, Atomsorten, Moleküle, etc. Wenn außerdem die Regeln der BWINF-Chemie (oder die eigenen Regeln) als nicht vollkommen verbindlich aufgefasst wurden, ist eine Bewertung anhand des Grades der Beachtung dieser Regeln denkbar.

Bei der Bewertung fremder Lösungsvorschläge hingegen sollte man auch mit „echten“ Fehlern rechnen und mit ihnen umgehen können. Hierzu muss eine Art „Fehlerfunktion“ definiert werden, die für ein Muster $M$ (also eine Gleichung mit Fragezeichen) und eine Gleichung $G$ ein numerisches Maß für die Übereinstimmung von $G$ mit $M$ zurückliefert. Wie diese Fehlerfunktion definiert wird, ist nicht entscheidend. Wichtiger ist zu erläutern, inwiefern die gewählten Kriterien die „Plausibilität“ der Gleichung bewerten. Dieser Aufgabenteil wurde von denen falsch verstanden, die Reaktionsgleichungen ohne Bezug zu einer Lückengleichung bewertet und dazu nur auf ihre Korrektheit hin untersucht haben.

# Aufgabe 2: Trau, schau, wem!

Es ist schon schwierig, „mega-in“ zu sein. Und zum Teil wohl auch relativ kostenintensiv. Schließlich ist SMSen ja nicht das preiswerteste Hobby. Immerhin: Wer diese Aufgabe ordentlich bearbeitet hat, kann nun gründlich ausprobieren, wie man sich in SMS-, Chat- oder sonstigen Communities zu verhalten hat – sofern sie halbwegs nach den regeln von St. Enga funktionieren.

## Simulationsablauf

Prinzipiell ist ein Tagesablauf für die Simulation recht einfach. Zunächst sucht ULI sich seine Clubster heraus und teilt ihnen den Ort und den Zeitpunkt für das abendliche Treffen mit. Man kann dabei davon ausgehen, dass alle Teilnehmer wissen, wann ULI seine Nachricht aussendet, so dass vorher auch keine Kommunikation stattfindet.

Anschließend werden fleißig SMSe zwischen den Clubstern hin und her geschickt. Diese SMSe sind nicht notwendigerweise synchronisiert. Jedoch muß der gesamte Zeitrahmen in der Simulation diskretisiert werden. Es bietet sich daher an, „SMS-Runden“ einzuführen. Jede Runde wird folgendermaßen ablaufen:

1. Jeder Teilnehmer überlegt für sich, ob und wem er eine SMS schreiben möchte. Natürlich kann ein Teilnehmer auch mehreren anderen eine Nachricht schicken – wir leben ja in Zeiten des Multi-Messagings.
2. Nachdem alle Teilnehmer ihre SMSe für diese Runde abgeschickt haben, lesen sie die SMSe, die sie bekommen haben und beantworten sie gegebenenfalls.

Diese Runden werden wiederholt, so lange noch Zeit ist und die Teilnehmer sich noch SMSen möchten (bzw. noch Guthaben auf ihren Prepaid-Karten haben).

Abends schickt ULI eine SMS mit dem korrekten Treffpunkt an alle. Zu diesem Zeitpunkt kann jeder für sich die Auswertung machen, wer ihm korrekte Treffpunkte genannt und wer eher falsche Daten übermittelt hat.

Eine zusammenfassende Darstellung des Tagesablaufs bietet Abbildung 1.

## Clubster: Struktur und Datenfluss

Aus dem Tagesablauf ergeben sich die nötigen Datenflüsse innerhalb eines Clubsters (s. auch Abbildung 2). Eingehende Nachrichten beeinflussen das Wissen des Clubsters: eve-Nachrichten geben Information über Ort und Zeit (diese sind gesichert, falls die Nachricht von ULI kommt) und bilden zusammen mit ULIs peve-Nachricht Grundlagen für Annahmen über die Verlässlichkeit anderer Clubster. Eingehende wuw-Fragen veranlassen ausgehende eve-Antworten, deren Inhalte vom Wissen über das Event und von den festgelegten Sympathiewerten abhängen.

ULI ist demgegenüber ein stark vereinfachter Clubster, der lediglich Wissen über Zeit und Ort verwendet, um Nachrichten zu erstellen, und für den Sympathie keine Rolle spielt.

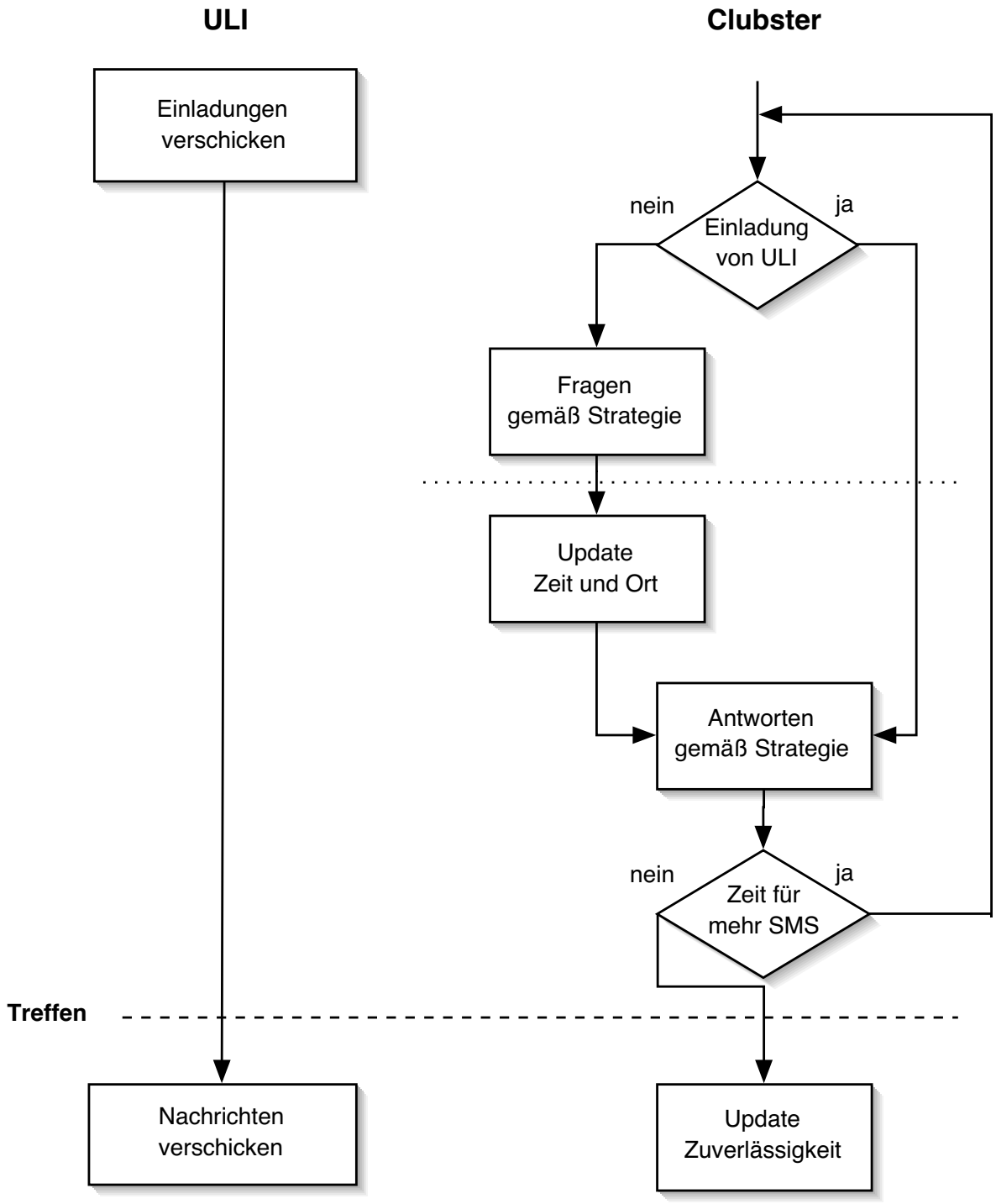


Abbildung 1: Tagesabläufe von ULI und den Clubstern

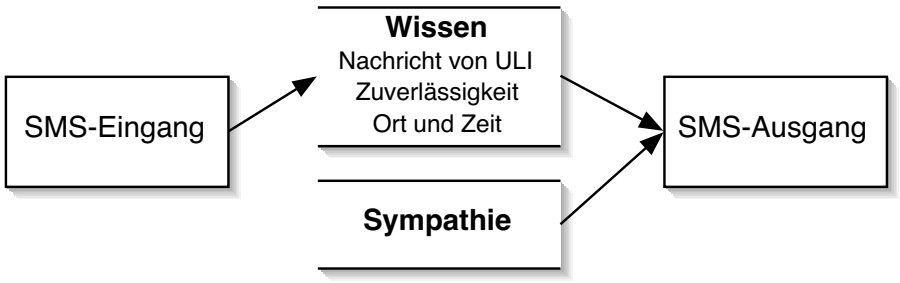


Abbildung 2: Clubster

Es ergibt sich für den allgemeinen Clubster ungefähr folgende Struktur (wobei die Sympathiewerte laut Aufgabenstellung konstant sein sollen):

```
CLUBSTER
    Liste    SMS-Eingang
    Liste    SMS-Ausgang
    Feld     Sympathiewerte
    Feld     Zuverlässigkeitswerte
    Bool     Nachricht_von_ULI_erhalten
    Enum     Ort
    Enum     Zeit
    Formel   Fragestrategie
    Formel   Antwortstrategie
    [Formel  Wissensabgleich]
```

Um in den Teilaufgaben 2 und 3 vernünftige Untersuchungen durchführen zu können, wird eine flexible Möglichkeit zur Spezifikation von Clubstern benötigt. Sehr heterogene Clubster-Mengen werden möglich, wenn die oben genannten Größen für einzelne Clubster individuell festgelegt werden können. Bei umfangreicheren Clubster-Mengen, die für die Experimente besonders interessant sind, ist das aber eher aufwändig. Eine sinnvolle Alternative ist die Festlegung von Clubster-Typen und ihres Anteils in der Gesamtmenge.

Welche Variante auch gewählt wird, wichtig ist eine Sicherungsmöglichkeit für Clubster-Spezifikationen, am besten in les- und editierbaren Dateien. Solche Dateien müssen dann natürlich auch eingelesen werden können. Nicht akzeptabel sind

- im Quelltext verdrahtete Clubsterbeschreibungen,
- jedes Mal neu einzutippende Clubster und
- schlecht verständliche Eingabeformate.

Das folgende Beispiel demonstriert ein akzeptables Format für individuelle Clubsterspezifikation. Ein Clubster hat hier einen Namen. Die binären Sympathiewerte werden als + und - angegeben; es gibt insgesamt nur zwei Clubster. Die Reihenfolge der Werte entspricht der Reihenfolge

der Clubster in der Spezifikationsdatei; einer der Werte sagt also, ob der Clubster sich selbst sympathisch ist oder nicht. Die Fragestrategie wird durch Wahrscheinlichkeiten (Werte von 0 bis 100) bestimmt, die angeben, mit welcher Wahrscheinlichkeit zuverlässige, unzuverlässige, sympathische und unsympathische Clubster gefragt werden. Für Antworten sieht es ähnlich aus; hier geht dem Wahrscheinlichkeitswert noch ein Code für die Art der Antwort voraus: 1 steht für eine falsche und 2 für eine richtige (also dem eigenen Wissensstand entsprechende) Antwort.

```
CLUBSTER
Thomas
Sympathie +-
Fragen zuverlaessig: 100
Fragen unzuverlaessig: 50
Fragen sympathisch: 0
Fragen unsympathisch: 0
Antworten zuverlaessig: 2 100
Antworten unzuverlaessig: 0 0
Antworten sympathisch: 2 100
Antworten unsympathisch: 1 100
```

Bei den Strategien muss eine Auswahl möglich sein. Schön ist, wenn die Strategien wie im Beispiel parametrierbar sind. Der Idealfall ist wohl mit einer komplexen Beschreibungssprache oder mathematischen Darstellung der verwendeten Strategien im Eingabe-File erreicht.

## Kommunikation unter den Clubstern

Im Diagramm von Abbildung 1 wird davon ausgegangen, dass Clubster, die von ULI benachrichtigt wurden, nicht mehr per SMS wuw-Fragen an andere Clubster stellen. So werden Kosten für SMS gespart. Andererseits kann es gerade für einen informierten Clubster sinnvoll sein zu fragen, um gezielt die Zuverlässigkeit anderer Clubster zu testen. Wenn also Clubster gezielte Fragen stellen, insbesondere an solche anderen Clubster, deren Zuverlässigkeit nicht sicher eingeschätzt werden kann, ist das eine positiv zu wertende Erweiterung. Wird hingegen planlos gefragt, z.B. ohne dass ein Clubster überprüft, ob er von ULI benachrichtigt wurde, so ist dies als eine Verschwendung von SMS anzusehen.

Einige mögliche Fragestrategien sind bereits auf dem Aufgabenblatt beschrieben. Generell hat man für die Strategie die Auswahl, immer zu fragen, nie zu fragen oder nur mit einer gewissen Wahrscheinlichkeit zu fragen. Und dies in Abhängigkeit von Sympathie, Zuverlässigkeit, bisherigem Wissen über Ort und Zeit und Anzahl bereits verschickter SMS. In Formeln könnte man sagen, dass die Wahrscheinlichkeit $p_{i,j}$, dass Clubster $i$ an Clubster $j$ eine WUW-Nachricht schickt, sich wie folgt berechnet:

$$p_{i,j} = f(\text{Sympathie}(j), \text{Zuverlässigkeit}(j), \text{Sicherheit über Treffpunkt}, \text{bereits verschickte SMS})$$

Im einfachsten Fall ist $f$ eine Sprungfunktion, d.h. es wird abgefragt, ob Sympathie vorliegt, ob $j$ zuverlässig ist, ob bereits eine hinreichende Sicherheit über den Ort und die Zeit des Treffens besteht und ob noch nicht zu viele SMS verschickt wurden.

Selbstverständlich können hier auch beliebig komplexe kontinuierliche Funktionen stehen. De facto ist dies kein besonderer Mehraufwand in der Realisierung. Die einzige Schwierigkeit liegt im entsprechenden Parser beim Einlesen entsprechender Clubster-Spezifikationen.

Es ist sinnvoll, eine SMS-Runde in zwei Teilphasen zu untergliedern. In der Teilphase 1 werden die SMS verschickt und erst in der Teilphase 2 werden sie gelesen. Sollte man das nicht tun, sind Clubster, die in der Reihenfolge der SMS-Absender weiter hinten stehen u.U. bevorteilt, da sie aus in dieser Runde bereits an sie gerichteten Fragen Rückschlüsse auf das Wissen ihrer Freunde ziehen können. Alternativ können natürlich auch Flags verwendet werden, um diesen Effekt zu vermeiden. Ansonsten ist mit leichten Verfälschungen von Simulationsergebnissen zu rechnen. Generell war das realisierte „Kommunikationsprotokoll“, also die Steuerung des Nachrichtenaustausches unter den Clubstern, gut zu dokumentieren.

Bei der Organisation der Clubster-Kommunikation kann auch mit Kanonen auf Spatzen geschossen und (unnötigerweise) technisch aufwändige Kommunikationsmechanismen implementiert werden (TCP/IP-Kommunikation, Clubster als Threads oder gar Prozesse, etc.) Das ist kein Fehler; es ist aber in diesen Fällen besonders darauf zu achten, dass die Kommunikation vernünftig und im Sinne zuverlässiger Simulationsergebnisse organisiert ist.

Beim Update von Zeit und Ort berechnet ein Clubster aus allen Antworten den wahrscheinlichsten Ort und die wahrscheinlichste Zeit des aktuellen Treffens, sinnvollerweise unter Berücksichtigung der Zuverlässigkeit der Clubster, die geantwortet haben. Hierbei können entweder nur die Ergebnisse der letzten Fragerunde berücksichtigt werden (weil sie besonders aktuell sind) oder aber sämtliche bisher erhaltenen Antworten.

Schließlich müssen noch die Anfragen der anderen Clubster beantwortet werden. Hierzu kann sich ein Clubster wie beim Fragen auch eine Wahrscheinlichkeit für Antworten an andere Clubster ausrechnen. Zusätzlich muss entschieden werden, ob der korrekte Ort oder ein vermeintlich falscher Ort genannt wird. Ansonsten gilt das gleiche wie für die Fragerunde.

## Auswertung

Die abschließende Auswertung dient lediglich dem Update der Zuverlässigkeitswerte. Hier gibt es sehr viele verschiedene, sinnvolle Möglichkeiten. Beispiel: Ein Zuverlässigkeitswert entspricht der Anzahl der korrekten Antworten minus der Anzahl der fehlerhaften Antworten. Alternativ lässt sich auch eine Struktur modellieren, um älteren Antworten weniger Gewicht zu geben als neueren. Nicht akzeptabel ist eine Strategie, die immer nur die letzte Antwort für den Zuverlässigkeitswert heranzieht.

## Experimente

Damit man in den Teilaufgaben 2 und 3 interessante Fragestellungen und Strategien bzw. Strategiekombinationen untersuchen kann, muss das Simulations-Programm zu jedem Zeitpunkt zumindestens die folgenden Punkte ausgeben können:

- aktuelle Zuverlässigkeitswerte für einen Clubster
- prozentuale Häufigkeit der Anwesenheit bei den Treffen für jeden Clubster
- Anzahl der verbrauchten SMS je Clubster (Achtung: WUW- und EVE-SMS zählen!!!)
- Für jede Runde, wem ULI eine SMS geschickt hat und wer zum Treffen erschienen ist.

Anstelle von Angaben für einzelne Clubster sind bei größeren Clubster-Mengen (prozentuale) Angaben für die ganze Menge bzw. für eventuell vorhandene Clubster-Typen sinnvoll. Interessant, aber aufwändig, ist darüber hinaus die Darstellung des aktuellen „Zuverlässigkeits-Netzwerks“ (ein vollständiger gerichteter Graph mit den Zuverlässigkeitswerten als Gewichten), das zwischen den Clubstern existiert. Übersichtlich ist das allerdings nur bis zu einer gewissen Größe der Clubster-Menge.

Die Ausgaben müssen nicht in eine Datei geschrieben, sondern können auch am Bildschirm dargestellt werden. Interessant ist die Möglichkeit, die statistische Auswertung erst nach einer Einschwingphase zu starten, in der die Clubster ihre Zuverlässigkeitswerte lernen.

Soweit zur 1. Teilaufgabe. In den beiden anderen Teilaufgaben sollen die bisher implementierten Werkzeuge angewandt werden. Dabei gibt es keine Vorgaben, welche Art von Fragen bzw. Strategien untersucht werden sollen; Abzüge gab es nur, wenn die Fragen oder Strategien allzu trivial waren. Ansonsten sind die Aufgabenstellungen zu beachten: In Teilaufgabe 2 ist explizit gefordert, dass die Ergebnisse beschrieben und die Untersuchungsmethodik begründet werden. Letzteres ist auch für Teil 3 wichtig. Hier gibt es außerdem noch die Fragen nach erfolgreichen und nach extremen Strategien, die vernünftig beantwortet sein wollen. Besonders aufwändige und gut durchgeführte Untersuchungen wurden mit Pluspunkten belohnt.

# Aufgabe 3: Wie steht's um den IQ?

Die Lösungshinweise zu dieser Aufgabe sind wie die Aufgabe selbst in drei Teile gegliedert:

1. Grundlagen zur Erzeugung von IQ-Tests
2. Werkzeug zur Erzeugung von IQ-Tests
3. Lösen von IQ-Tests

## Grundlagen zur Erzeugung von IQ-Tests

### Katalog einfacher Basisobjekte

Als einfache Basisobjekte bieten sich die in den Beispielen der Aufgabenstellung verwendeten Objekte an: Kreis, Quadrat, (gleichseitiges) Dreieck, Haus. Natürlich können auch andere Objekte, wie z. B. ein Achteck, ein Trapez, ein Parallelogramm, etc. verwendet werden. Wichtig ist, dass die Objekte in verschiedenen Größen eingesetzt werden können. Es soll sich ausdrücklich nur um einfache Basisobjekte handeln, es geht nicht um besonders komplexe oder schöne Objekte, sondern um die Transformationen, die auf die Objekte angewandt werden können.

Eine Erweiterungsmöglichkeit besteht darin, den Objekten verschiedene Farben zuzuweisen (und entsprechende Transformationen zu definieren).

### Regeln zum Zusammensetzen der Basisobjekte zu Figuren

Zur Zusammenstellung von Objekten zu Figuren und zur internen Repräsentation der Figuren gibt es verschiedene Möglichkeiten:

1. Man gibt Lagebeziehungen der Objekte relativ zueinander an. Mögliche Lagebeziehungen sind z. B. IN (üblicherweise konzentrisch interpretiert), UEBER, UNTER, LINKS_VON, RECHTS_VON, AUF. Eine Figur wird dann durch ihre Objekte (Typ, Größe) und die Lagebeziehungen beschrieben. Die Figur B aus dem 1. Beispiel kann z.B. folgendermaßen beschrieben werden: Objekte: Kreis A (Größe 4), Kreis B (Größe 3), Kreis C (Größe 2); Lagebeziehungen: Kreis C IN Kreis B, Kreis B IN Kreis A.
2. Man gibt die Ausrichtung der Objekte relativ zur Zeichenfläche an, z. B. vertikal TOP, MIDDLE, BOTTOM, horizontal LEFT, CENTER, RIGHT. Damit wird die Zeichenfläche in ein Raster (hier mit 9 Feldern) eingeteilt. Die Figur B aus dem 1. Beispiel kann dann folgendermaßen beschrieben werden: Kreis A (Größe 4, MIDDLE, CENTER), Kreis B (Größe 3, MIDDLE, CENTER), Kreis C (Größe 2, MIDDLE, CENTER).
3. Ungeeignet hingegen ist die absolute Positionierung der Objekte (z. B. durch die Angabe von Pixelkoordinaten), da Transformationen dadurch erschwert werden.

Wichtig ist bei dieser Teilaufgabe, dass die interne Repräsentation den Transformationen entgegenkommt und dass die wichtigsten Lagebeziehungen (vgl. 1) möglich sind. In der Dokumentation muss angegeben werden, wie die einzelnen Beziehungen genau interpretiert bzw. implementiert sind.

Nicht vergessen darf man, dass zu der Beschreibung der Objekte auch die Drehung gehört: Man muss angeben können, ob das Objekt in Bezug auf eine Grundstellung gedreht wurde ($0°, 90°, 180°, 270°$), ansonsten könnte bspw. die Figur B des 3. Beispiels nicht repräsentiert werden. Für entsprechend symmetrische Objekte (wie Kreis und Quadrat) ist die Angabe einer solchen Drehung allerdings sinnlos.

Auf die Möglichkeit, Objekte auch gespiegelt darstellen zu können, kann ggf. verzichtet werden, wenn die Objekte die entsprechenden Symmetrieeigenschaften erfüllen, so dass die Drehung ausreicht, um eine Spiegelung zu berücksichtigen.

**Transformationen**

Einfache Transformationen sind:

- Typ eines Objekts ändern (z. B. Kreis durch ein Quadrat ersetzen)
- Hinzunahme (vgl. 1. Beispiel) / Weglassen (vgl. 2. Beispiel) eines Objekts
- Drehen ($90°, 180°, 270°$) der gesamten Figur (vgl. 3. Beispiel)
- Spiegeln (horizontal, vertikal) (vgl. 3. Beispiel)
- Vergrößern / Verkleinern eines Objekts

Eine komplexe Transformation besteht dann aus der Hintereinanderausführung einfacher Transformationen. Komplexe Transformationen wurden nur von wenigen Teilnehmern realisiert. Der Punktabzug für ihr Fehlen fiel deshalb recht moderat aus.

Transformationen können sich auf einzelne Objekte, aber auch auf ganze Figuren beziehen; eine Drehung (um den Mittelpunkt) einer Figur ist dann nicht mehr das gleiche wie die Drehungen aller einzelnen Bestandteile.

Nicht alle Transformationen sind auf alle Objekte bzw. Figuren sinnvoll anwendbar. Die Drehung eines Kreises, aber auch eines Quadrats verändert das Objekt nicht sichtbar. Die Sichtbarkeit der Veränderungen durch eine Transformation ist aber für unsere IQ-Tests entscheidend. Dies muss berücksichtigt werden.

**Mehrdeutigkeit von Transformationen**

Bei den betrachteten Testaufgaben lässt sich nicht immer eine eindeutige Lösung bestimmen. Beim 2. Beispiel (der Aufgabenstellung) drängt sich als Lösung auf, einfach die innere Figur zu streichen. Dann ist Figur 3 die richtige Antwort. Man kann aber auch sagen, dass man Figur B

aus Figur A erhält, indem man die äußere Figur streicht und die innere vergrößert. Wenn man diese komplexere Transformation dann auf die Figur C anwendet, erhält man als Lösung Figur 4. Intuitiv schätzt man wohl die erste Lösung als „einfacher“ erreichbar ein.

Das 3. Beispiel demonstriert, dass es auch Fälle mit zwei gleichberechtigten Lösungen gibt: Entweder man dreht die Figur um $180°$ (und erhält Figur 2 als Lösung) oder man spiegelt die Figur an der horizontalen Achse (und erhält Figur 1). Keine der beiden Lösungen ist „einfacher“.

Mit Mehrdeutigkeiten muss also schon bei der Generierung von Tests konstruktiv umgegangen werden. Darüber hinaus sollte sichergestellt werden, dass bei einem Test keine „falsche“ Lösung als Alternative angeboten wird, die eigentlich intuitiv „einfacher“ zu erreichen ist als die „richtige“ Lösung. Bei dem 2. Beispiel sollte also nicht Figur 4 als richtige Lösung und Figur 3 als falsche angesehen werden.

Um dies zu erreichen, bietet sich eine Bewertung des Schwierigkeitsgrades einer Transformation an. Zunächst kann man zwischen den einfachen Transformationen differenzieren und verschiedene Schwierigkeitsgrade zuordnen. Z.B. kann man festlegen, dass das Vergrößern eines Objekts schwieriger nachvollziehbar ist als die Hinzunahme eines Objekts. Außerdem ist eine komplexe Transformation auf jeden Fall schwieriger als eine einfache Transformation.

**Ähnlichkeiten**

Um möglichst „nahestehende“ Figuren als Alternativen zur richtigen Lösung erzeugen zu können, muss man sich zunächst einmal eine Definition für „nahestehend“ (bzw. „ähnlich“) überlegen:

1. Zwei Figuren sind umso ähnlicher, je einfacher sie ineinander transformiert werden können. Bspw. ist bei dem 2. Beispiel die Figur 5 sehr ähnlich zur Figur 3, da sie durch die einfache Transformation „Typ eines Objekts ändern“ entstanden ist.

2. Interessant ist es, bei der Definition die „äußeren Umstände“ zu berücksichtigen; gesucht wird ja eine ähnliche Figur X zu einer Figur Y, die durch eine Transformation aus einer gegebenen Figur C entstanden ist. Deshalb bietet sich folgendes an: X ist ähnlich zu Y, wenn X durch eine ähnliche Transformation wie Y aus C entstanden ist. Ähnliche Transformationen sind dann z. B. Drehen und Spiegeln, Vergrößern und Verkleinern, Weglassen verschiedener Objekte. Nach dieser Definition ist z. B. bei dem 2. Beispiel die Figur 2 ähnlich zur Figur 3 (obwohl die beiden Figuren, wenn man sie wie bei der ersten Definition isoliert betrachtet, nicht ähnlich sind), da die Figur 3 aus der Figur C durch Weglassen des Quadrats und die Figur 2 durch Weglassen des Kreises entsteht und das „Weglassen verschiedener Objekte“ als ähnlich definiert wurde.

3. Beide Definitionen können kombiniert werden, um bspw. zwei Alternativen entsprechend der Definition 1 und zwei Alternativen entsprechend der Definition 2 zu erzeugen.

Wurde der Begriff der Ähnlichkeit nicht nur auf die Figuren 1 bis 5, sondern auch auf A und C angewandt, gab es Pluspunkte. Dies wird nämlich durch die Aufgabenstellung zwar nicht nahegelegt, ist aber sinnvoll. Immerhin soll auf C die gleiche Transformation anwendbar sein wie auf A. Das muss (zur Not auch ohne Ähnlichkeitsüberlegung) gesichert werden.

**Möglichkeiten zur programmtechnischen Umsetzung**

An dieser Stelle muss die Brücke von den eher abstrakten Überlegungen der bisherigen Teilaufgaben zu der konkreten Realisierung in Teilaufgabe 2 geschlagen werden. Es muss beschrieben werden, wie die bisherigen Überlegungen realisiert werden können, wobei sich diese Beschreibung natürlich mit der tatsächlichen Realisierung in Teilaufgabe 2 decken muss.

Es gibt keine Vorgaben, wie die programmtechnische Umsetzung auszusehen hat, solange sie geeignet (d. h. die abstrakten Begriffe „Basisobjekt“, „Figur“, „Transformation“ und „Ähnlichkeit“ können vollständig und nicht unnötig kompliziert erfasst werden) und nachvollziehbar ist.

## Werkzeug zur Erzeugung von IQ-Tests

Dies Auswahlfiguren wollen wir jetzt nicht mehr mit den Zahlen 1 bis 5 bezeichnen, die sich auf eine Reihenfolge in der Darstellung eines IQ-Tests beziehen, sondern mit den Buchstaben D bis H. Dann besteht eine Testaufgabe aus den 8 Figuren A, B, C, D, E, F, G, H mit den Eigenschaften

- A verhält sich zu B wie C zu D
- E, F, G, H sind ähnlich zu D

Bei der Aufgabenstellung war an eine Software gedacht worden, die den Benutzer bei der eigenständigen Konstruktion von IQ-Tests unterstützt. Diese sollte auf der Basis der in der ersten Teilaufgabe erarbeiteten Konzepte erstellt werden. Ein solches Werkzeug kann also Unterstützung bei der Konstruktion insbesondere der Figuren A und C leisten, die Figuren B und D anhand der vom Benutzer ausgewählten Transformationen generieren und bei der Erstellung der Figuren E bis H auf der Grundlage des realisierten Ähnlichkeitsbegriffes helfen.

Häufig wurde aber auch eine vollautomatische Testgenerierung entwickelt. Das ist eine sinnvolle Erweiterung. Die so gewonnenen Pluspunkte wurden aber wieder abgezogen, wenn die automatische Generierung die einzige Möglichkeit war.

Mit dem entwickelten Werkzeug sollen mindestens drei vollständige Testaufgaben generiert werden. Diese Beispiele müssen (wie immer) nachvollziehbar sein, insbesondere bei komplexen Transformationen sollten die Transformationen explizit angegeben sein (bspw. “Weglassen des größeren Objekts und Vergrößern des kleineren Objekts”). Natürlich werden die Figuren als Bilder und nicht in der internen Beschreibung ausgegeben, da man sonst wenig damit anfangen kann. Wichtig ist auch, dass die Beispiele eine gewisse Vielseitigkeit des Werkzeugs demonstrieren.

Nun zu den einzelnen Aspekten dieser Teilaufgabe:

**Figur A erzeugen** Die Figur A muss von Hand vorgegeben werden (entweder durch die interne Beschreibungssprache oder durch eine GUI). Zusätzlich kann man die Möglichkeit der zufälligen Erzeugung einer Figur anbieten.

**Transformation festlegen** Auch die Transformation muss der Benutzer von Hand vorgeben können. Dabei sollte auch die Vorgabe einer komplexen Transformation, also einer Folge von einfachen Transformationen, möglich sein. Auch hier kann man zusätzlich die zufällige Auswahl einer Transformation anbieten.

**Figur C erzeugen** Auch Figur C kann von Hand eingegeben werden. Besser ist aber, wenn aus Figur A eine ähnliche Figur C, auf die die gewählte Transformation gut angewandt werden kann, erzeugt wird. Auf jeden Fall werden Verknüpfungen zwischen den Objekten der Figur A und der Figur C benötigt. Beim 2. Beispiel der Aufgabenstellung muss z. B. das kleine Dreieck der Figur A mit dem kleinen Quadrat der Figur C verknüpft sein. Nur so kann die Transformation „Entferne das kleine Dreieck“ (aus der Figur A) analog bei der Figur C (nämlich „Entferne das kleine Quadrat“) durchgeführt werden.

**Figur B und D erzeugen** Die Anwendung der gewählten Transformation auf die Figur A bzw. C erzeugt dann die Figur B bzw. D.

**Figuren E, F, G, H erzeugen** Die zu D ähnlichen Figuren müssen dann in Abhängigkeit von der gewählten Definition für “Ähnlichkeit” (entweder aus D nach Definition 1 oder aus C nach Definition 2) erzeugt werden. Wichtig ist, dass spätestens an dieser Stelle auf die Problematik der Mehrdeutigkeit eingegangen wird (siehe oben).

## Lösen von IQ-Tests

### Welchen IQ kann eigentlich ein Computerprogramm haben?

Diese Frage war eher einleitend gemeint; beantwortet werden musste sie nicht. Für eine vernünftige Beantwortung gab es Pluspunkte, wenn die Antwort durchdacht, die Behauptungen begründet und die Argumentation nicht zu einseitig waren.

Ein Argument für einen hohen IQ: Der IQ wird gewöhnlich mit standardisierten Tests ermittelt. Aufgrund dieser Standardisierung ist es für ein Programm möglich, sehr gute Ergebnisse zu erzielen und damit einen hohen IQ bescheinigt zu bekommen. So kann bspw. ein Programm (wie im folgenden gezeigt wird) visuelle Analogien finden, wenn alle in Frage kommenden Transformationen bekannt sind.

Ein Argument gegen einen hohen IQ: Dass ein solches Programm aber nur scheinbar einen hohen IQ hat, stellt sich schnell heraus, wenn es mit neuen Aufgaben, die vom Standard abweichen, gefüttert wird: man verwende z.B. eine Transformation, die dem Programm nicht bekannt ist. Ein Mensch ist im allgemeinen fähig, sich darauf einzustellen, ein einfaches Programm (wie in dieser Teilaufgabe gefordert) hingegen nicht.

**Programm zum Lösen von IQ-Tests**

Als Eingabe erhält das Programm die 8 Figuren, die durch das Werkzeug aus Aufgabenteil 2 erzeugt wurden. Es ist vollkommen ausreichend, wenn die Figuren in der internen Darstellung eingegeben werden. Die Analyse eines externen Formats (Bitmap) war nicht verlangt. Bildanalyse ist bekanntermaßen nicht einfach; die automatische Generierung von Strukturbeschreibungen aus vorgelegten Bildern ist sicher eine sinnvolle Erweiterung.

Einige Teilnehmer haben allerdings mit Bilddaten *statt* struktureller Beschreibungen gearbeitet. Daran war nicht gedacht. Wer auf der Basis von Bilddaten aber kreative und interessante Lösungsideen entwickelt hat, wurde natürlich nicht bestraft.

Zum Lösen einer Testaufgabe bietet sich folgende prinzipielle Vorgehensweise an:

1. „einfachste“ Transformation von A nach B ermitteln (dies ist der entscheidende Punkt in dieser Teilaufgabe);
2. gefundene Transformation auf C anwenden, liefert Figur X; und
3. Figur X mit den angebotenen 5 Figuren vergleichen und die richtige ermitteln; wenn erfolglos, ggf. mit der nächstschwereren Transformation (von A nach B) fortfahren

zu (1): Um die einfachste Transformation zu finden, kann man eine Art Breitensuche in einem gewichteten Graphen verwenden, wobei die Knoten die Figuren repräsentieren (Figur A ist der Startknoten, Figur B der Zielknoten), die gerichteten Kanten die Transformationen und die Gewichte den Schwierigkeitsgrad der einfachen Transformationen.

zu (2): Beim Anwenden der gefundenen Transformation auf C verbleibt eine Schwierigkeit: Wenn sich die Transformation auf ein Objekt bezieht, muss zunächst das analoge Objekt in C durch Vergleich der Lagebeziehungen gefunden werden.

zu (3): Der Vergleich der Figur X mit den angebotenen Figuren ist recht einfach: Die Objekte und die Lagebeziehungen müssen übereinstimmen.

**Mehrdeutigkeit** Auch hier musste die Problematik der Mehrdeutigkeit erkannt und ein Algorithmus gewählt werden, der Lösungen, die durch einfachere Transformationen erreicht werden, bevorzugt. Bei gleichwertigen Transformationen reicht eine willkürliche Entscheidung aus.

**Komplexität** Die Laufzeit des angegebenen Algorithmus hängt exponentiell von der Suchtiefe ab, also von der Anzahl der nacheinander ausgeführten einfachen Transformationen. Bei vielen Objekten, vielen verschiedenen Transformationen und v.a. bei großer Suchtiefe kann die Laufzeit sehr groß werden. Es muss sichergestellt sein, dass das Programm abbricht, wenn keine passende Transformation (bis zu einer gewissen Suchtiefe) gefunden wurde. Bei konkreten Tests wird die Komplexität der Transformationen aber eher begrenzt sein. Deshalb ist es akzeptabel, sich auf eine maximale Suchtiefe festzulegen.

Um die Laufzeit des Programms zu reduzieren, sollte man auch den 1. Schritt genauer betrachten: Bei der Breitensuche kommen viele Transformationen mit dem gleichen Schwierigkeitsgrad in Frage. Welche Transformation soll man zuerst ausprobieren? Welche Transformation soll man überhaupt nicht ausprobieren? Um diese Fragen zu beantworten, muss man das Soll (Figur B) mit dem Ist (Figur A) vergleichen: Welche Objekte (Typ, Größe) werden benötigt, welche sind schon da? Offensichtlich unsinnige Transformationen (bspw. ein Dreieck hinzufügen, wenn B überhaupt kein Dreieck enthält) werden unabhängig vom Schwierigkeitsgrad auf keinen Fall ausprobiert. Sinnvolle Transformationen (die die Differenz von Soll zu Ist verkleinern) hingegen werden bevorzugt.

### Erweiterungen

Einige der hier zusammengestellten Erweiterungsideen sind auch schon bei der zugehörigen Teilaufgabe erwähnt worden:

- Objekten eine Farbe zuordnen (und geeignete Transformationen definieren)
- komplett automatische Generierung von IQ-Tests
- automatisches Erzeugen von C aus A in Abhängigkeit von der gewählten Transformation
- IQ-Test unter „realistischen“ Bedingungen (bspw. mit Zeitbeschränkung und Auswertung unter Berücksichtigung der Zeit und des Schwierigkeitsgrades der Transformationen)
- Bildanalyse (sehr aufwendig !)

## Perlen der Informatik – aus den Einsendungen

**Allgemeines**

Weiter habe ich noch einige Datentypen definiert, um mir das Programmieren zu erleichtern.

Diejenigen, die nicht am richtigen Ort sind, sind vor allem die ersten in der For-Schleife.

Sollten Sie mit dieser Einstellung Probleme haben, können Sie den Wert hochsetzen und neukompilieren.

Das Programm arbeitet mit HTML. Damit unterstützt das Programm zukunftsträchtige Datenformate und ist internetfähig.

Das muss wohl aus dem üblichen Wahnsinn entstanden sein, den ich beim Programmieren regelmäßig entwickelt habe.

**Quellcode-Schätze**

```
unsympathy
While (1) // Tag fuer Tag vergeht das Leben
OBJEKT ****oFeld;
procedure OHExtrawurst (var Gleichung: TGleichungsseite);
```

**Aufgabe 1**

Dazu war in der Aufgabenstellung … ein furchtbar umständliches System erklärt worden.

Dadurch, dass manche Atome eine negative Wertigkeit haben, kommt es dazu, dass der Wert mancher Atome nun auch negativ ist.

In Ermangelung eines Oberbegriffs für die beiden Seiten der Reaktionsgleichung (Edukte und Produkte) habe ich das Wort Dukt bzw. Dukte eingeführt.

Prinzipiell lässt sich so ein Gleichungssystem lösen. Aber das überlasse ich lieber den bewundernswerten Programmierern von Mathematica und Derive.

Das blöde Word ändert meine zwei Bindestriche + > immer in diesen hässlichen Pfeil.

Insgesamt distanziere ich mich sehr stark von der Zeichenkette, die der Benutzer eingibt, und löse diese sofort zu Datenstukturen auf, die eher chemie-orientiert sind.

Eine erste Reaktion auf die Frage: „Kann man ihr helfen?“ war also: „Ich nicht“.

Vielleicht wäre die beste Lösung, sie würde einen Chemiker (z.B. Karlchen Knallgas) heiraten und mit ihm zusammen die Aufzeichnungen durchgehen.

**Aufgabe 2**

Dort, wo sich Werte gleich 0% ergeben, muss es sich um Rundungsfehler handeln. *Warum dieser fehlerhafte Wert dann aber als 0,000% mit drei Stellen Genauigkeit angegeben wird …*

Dabei habe ich eine prinzipiell unlogische Entscheidung getroffen, die aber dennoch die Qualität der Ergebnisse anheben dürfte.

Nach dem Programmstart wird der Benutzer zunächst in eine Art Smalltalk verwickelt, …

*BWINF proudly presents:* Der Herr der Simulationen. Teil 1 – Die Agenten

**Aufgabe 3**

Dazu habe ich zunächst angenommen, dass die Rotation von Sinus und Kosinus abhängig sein muss – das ist in der Regel immer so bei solchen Winkelgeschichten.

… benutzt das Programm vier Basisobjekte: Quadrat, Kreis, Dreieck und Haus.

Drehung um $90°$ im Uhrzeigerunsinn.

Dazu habe ich eine übergeordnete Klasse „Hirn“ erstellt.