Reduktion*

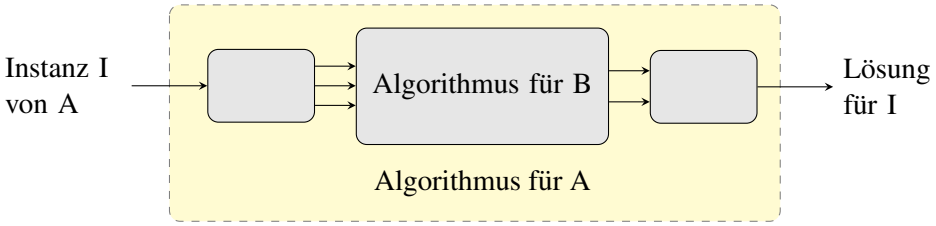
Gib mir einen Hebel der lange genug ist, und einen Stützpunkt um ihn aufzusetzen, und ich werde die Welt bewegen. -Archimedes

Hat man ein neues Problem gegeben, so fragt man sich ob man es lösen kann, indem man es einfach in ein anderes Problem transformiert, von dem wir schon wissen wie es zu lösen ist. Wir geben nun eine informelle Einführung zu den Begriffen die hinter diesem Prozess stehen.

Wir sagen, ein Problem A ist *reduzierbar auf* ein Problem B, und schreiben $A \leq B$, wenn wir, in effizienter Weise, einen Algorithmus der B löst benützen können um einen Algorithmus zu finden der A löst. Zwei Probleme heißen *äquivalent*, wenn sie gegenseitig aufeinander reduzierbar sind.

"In effizienter Weise" kann hier verschiedene Dinge bedeuten, die auch auf unterschiedliche Reduktionsbegriffe führen. Wir verstehen hier das folgende: innerhalb einer Zeitspanne die die maximale Laufzeit des Algorithmus der B löst um nicht mehr als schlimmstenfalls ein konstantes Vielfaches übersteigt.

Reduktion von A auf B kann man wie folgt darstellen:



Task 1 Zeige, dass die Berechnung des Quadrates einer Zahl auf Multiplikation reduziert, und das Multiplikation zweier Zahlen zur Berechnung des Quadrates einer Zahl reduziert (dabei sei vorausgesetzt, dass man subtrahieren und durch 2 dividieren in höchstens so vielen Schritten durchführen kann wie zur Berechnung eines Quadrates nötig sind).

Task 2 Zeige, dass die Berechnung der Summe aller Zahlen zwischen 1 und 100

*Das auf diesen Seiten aufbereitete Material stammt aus verschiedenen Quellen im internet. Von den meisten kann man den ursprünglichen Autor nicht mehr feststellen; wir zitieren diese Arbeiten pauschal!

die nicht durch 3 teilbar sind auf die Berechnung der Summe der ersten n Zahlen für gegebenes n, $n \leq 100$, reduziert.

Task 3 Sie wollen ihren Computer an einen Projektor anschliessen. Ihr Computer hat einen VGA Ausgang, der Projektor aber keinen VGA Eingang. Sie haben eine Schachtel mit Adapters. Jeder Adapter hat einen Eingang eines gewissen Typs, und einen Ausgang eines gewissen Typs (diese könnten auch gleich sein). Sie wollen feststellen ob es möglich ist eine Kette von Adaptern so zu bilden, dass Sie Ihren Computer mit dem Projektor verbinden können. Um das Problem konkreter zu formulieren, sei angenommen, dass Sie die folgenden Sammlung an Adaptern haben, wobei (I, O) bedeutet, dass der Adapter Eingang I und Ausgang O hat:

(HDMI, USB)	(DB13W3, X)	(VGA, HDMI)
(VGA, DisplayPort)	(DVI, DB13W3)	(DVI, DisplayPort)
(DB13W3, CATV)	(S-Video, DVI)	(USB, S-Video)
(DisplayPort, HDMI)	(Firewire,SDI)	(SDI,X)

Entwerfe mittels einer geeigneten Reduktion einen Algorithmus, der bestimmt ob es möglich ist den Computer mit dem Projektor zu verbinden.

Hier ist ein kleiner Einblick in Anwedungen von Reduktionen:

(1) positiv: Wenn $A \leq B$, dann gibt uns ein (effizienter) Algorithmus zur Lösung von B einen (effizienten) Algorithmus zur Lösung von A. Hier bedeutet "effizient" Lösbarkeit mit einer gewissen Komplexität, und Reduktion ermöglicht es daher *obere Schranken* für die Komplexität von A anzugeben.

(2) negativ: Wenn es keinen (effizienten) Algorithmus zur Lösung von A gibt, dann kann es auch keinen (effizienten) Algorithmus zur Lösung von B geben. Auch hier verstehen wir "effizient" im Sinne einer gewissen Komplexität, und erhalten somit die Möglichkeit *untere Schranken* für die Komplexität von B anzugeben. Noch interessanter ist wenn man weiss, dass es keinen Algorithmus gibt um A zu lösen, denn so kann man die Grenzen der Berechenbarkeit untersuchen und beweisen, dass Problem B auch *unentscheidbar* ist.

Beispiel für eine obere Schranke: Das Finden der k-t größten Zahl in einer Liste von Zahlen (bekannt als das *Auswahlproblem*) läuft darauf hinaus die Liste in absteigender Ordnung zu sortieren und dann das k-te Element zu retournieren. Da wir Sortieralgorithmen kennen, die höchstens $n \log(n)$ Schritte benötigen, hat das Auswahlproblem eine Komplexität von höchstens $n \log(n)$ (obere Schranke).

Beispiel für eine untere Schranke:[1] *Es gibt keinen Sortieralgorithmus der in linearer Zeit abläuft*; *Sortieren ist auf Bildung der konvexen Hülle reduzierbar*.

[1] Im folgenden benützen wir mehrere Behauptungen die wir höchstwahrscheinlich in dieser Lehrveranstaltung nicht beweisen werden können. Diese werden in italics geschrieben, und dienen nur dazu die Bedeutung der Beispiele harauszustreichen.

Daher gibt es keinen in linearer Zeit laufenden Algorithmus um konvexe Hüllen zu berechnen.

Beispiel für Unentscheidbarkeit: *Das Halteproblem (gegeben eine Turing Maschine M und ein Eingabewort w, retourniere genau dann wahr, wenn M nach Eingabe von w irgendwann einen Haltezustand erreicht) ist unentscheidbar*; *Das Halteproblem ist auf das Leerheitsproblem (gegeben eine Turing Maschine M, retourniere genau dann wahr, wenn M kein einziges Eingabewort w akzeptiert) reduzierbar*; Daher ist das Leerheitsproblem unentscheidbar.

Alles Gute, viel Spass und Freude, für Ihr weiteres Studium !