Universität des Saarlandes
FR Informatik

Kurt Mehlhorn und Corinna Coupette

WiSe 2021/22

Übungen zu Ideen der Informatik

https://www.mpi-inf.mpg.de/departments/algorithms-complexity/teaching/winter21/ideen/

Blatt 2 Abgabeschluss: 8.11.2021

Die Übung ist umfangreicher als normal, weil Sie zwei Wochen zur Bearbeitung haben.

Aufgabe 1 (*7 Punkte*) Betrachte folgendes Programm. Wir nehmen an, dass im Register R0 eine natürliche Zahl n steht, d.h., eine der Zahlen 0, 1, 2, 3, ... Im Programm steht der Stern im Zeile 2 für Multiplikation.

1 R1 $\leftarrow$ 0

2 if R1 * R1 $\geq$ R0, BZ $\leftarrow$ 5

3 R1 $\leftarrow$ R1 + 1

4 BZ $\leftarrow$ 2

5 drucke R1

6 STOP

- Was druckt das Programm, wenn in R0 steht: 0, 1, 4, 7, 9?
- Was gibt das Programm aus, wenn in R0 die Zahl n steht?
  - Überlegen Sie Sich zunächst, welche Werte R1 nacheinander annimmt.
  - Wird der Wert von R0 jemals abgeändert?
  - In Zeile 2 wird das Quadrat von R1 mit R0 verglichen. Wann springt man zum Ende des Programms in Zeile 5? Man springt, wenn das Quadrat von R1 ..... R0 ist. Hier sind ein paar Möglichkeiten: kleiner als, größer als, größer gleich. Welche ist richtig?
  - Genauer, man springt, wenn die Bedingung "das Quadrat von R1 ist .... R0" zum .... Mal erfüllt ist.
  - Das Program druckt ...

Lösung:

- Die Ausgaben sind 0, 1, 2, 3, 3. Ich rechne das für die Eingabe 7 genauer vor.
  Der BZ wird auf 1 gesetzt und in R1 wird 0 abgespeichert. Der BZ wird auf 2 erhöht.
  0 * 0 $\geq$ 7 ist falsch. Daher wir der BZ auf 3 erhöht
  Wir erhöhen R1 auf 1, erhöhen den BZ auf 4. Der Sprungbefehlt in Zeile 4 setzt den BZ auf 2.
  1 * 1 $\geq$ 7 ist falsch. Daher wir der BZ auf 3 erhöht
  Wir erhöhen R1 auf 2, erhöhen den BZ auf 4. Der Sprungbefehlt in Zeile 4 setzt den BZ auf 2.
  2 * 2 $\geq$ 7 ist falsch. Daher wir der BZ auf 3 erhöht
  Wir erhöhen R1 auf 3, erhöhen den BZ auf 4. Der Sprungbefehlt in Zeile 4 setzt den BZ auf 2.
  3 * 3 $\geq$ 7 ist richtig. Daher wir der BZ auf 5 erhöht. Wir drucken 3, erhöhen den BZ auf 6 und halten an.

- Das Programm druckt die kleinste ganze Zahl größer gleich $\sqrt{n}$. In Formeln, es druckt $\lceil \sqrt{n} \rceil$. Die Klammern $\lceil$ und $\rceil$ heißen Gaussklammern. Sie runden jede Zahl auf die nächste ganze Zahl auf. Also $\lceil 3.5 \rceil = 4$, $\lceil 2.6 \rceil = 3$ und $\lceil 3 \rceil = 3$.
  - R1 durchläuft die natürlichen Zahlen. In Zeile 1 setzen wir R1 auf 0 und in Zeile 3 erhöhen wir R1 um 1.
  - Der Wert von R0 wird nie geändert.
  - Man springt, wenn das Quadrat von R1 *größer gleich* R0 ist.
  - Genauer, man springt, wenn die Bedingung "das Quadrat von R1 ist größer gleich R0 zum *ersten* Mal erfüllt ist.
  - Das Program druckt *die kleinste ganze Zahl R1 mit R1 * R1* $\geq n$.

Aufgabe 2 (*7 Punkte*) In der Vorlesung sahen wir eine Turingmaschine, die zählt. Dabei war die Annahme, dass das Band mit

$$\ldots 0000000BBBBBBB\ldots$$

initialisiert ist und der Kopf der Maschine am Anfang auf der rechtesten Null steht. Modifizieren Sie die Turingmaschine so, dass sie für den Anfangsbandinhalt

$$\ldots BBBBBBBBBBBBBBBBBBBBBBBBBB\ldots$$

funktioniert.

Zur Erinnerung: In der Vorlesung nahmen wir an, dass das Band am Anfang mit ....00000BBBBBB..... beschriftet ist, der Kopf auf der rechtesten Null steht und die Maschine im Zustand q1 ist. Das Program war

```
q1 0 q2 1 S
q1 1 q1 0 L
q2 0 q2 0 R
q2 1 q2 1 R
q2 B q1 B L
```

Lösung: Wir geben der Maschine einen neuen Anfangszustand q0. Sie druckt eine 0 und geht in den Zustand q1 über. Danach kann sie fast so arbeiten, wie in der Vorlesung. Wenn wir im Zustand q1 eine 1 lesen, schreiben wir eine 0 und gehen nach links. Sobald wir eine 0 oder ein B (DAS IST NEU) lesen, drucken wir eine 1 und gehen in den Zustand q2 über. Im Zustand q2 gehen wir nach rechts bis wir ein B lesen. Wir gehen eins nach links und wieder in den Zustand q1. Die ganze Tafel ist

q0	B	q1	0	S	neue Zeile
q1	0	q2	1	S	
q1	1	q1	0	L	
q1	B	q2	1	R	neue Zeile
q2	0	q2	0	R	
q2	1	q2	1	R	
q2	B	q1	B	L	

Einer der Hörer hat noch eine elegantere Lösung gefunden. Er braucht keinen neuen Zustand, sondern startet die Maschine wie bisher im Zustand q_1. Es kommt nur eine Zeile dazu. Falls die Maschine im Zustand q_1 ein B liest, druckt sie eine Eins und geht nach rechts und in den Zustand q_2. Also

q1	1	q1	0	L
q1	0	q2	1	R
q1	B	q2	1	R
q2	0/1	q2	0/1	R
q2	B	q1	B	L

Aufgabe 3 (*4 Punkte*) Ein Schaltnetz besteht aus Gattern. Wir arbeiten mit drei Arten von Gattern, Und-Gatter, Oder-Gatter, und Nicht-Gatter. Und-Gatter ($\wedge$) und Oder-Gatter ($\vee$) haben je zwei Eingänge und einen

Ausgang, Nicht-Gatter ($\neg$) haben einen Eingang und einen Ausgang. Die Gatter operieren auf den booleschen Werten (auch Bits genannt) 0 und 1 gemäß folgenden Regeln.

x	y	$x \wedge y$	$x \vee y$	$\neg x$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Der Ausgang eines Und-Gatters ist also genau dann 1, wenn beide Eingänge 1 sind, und der Ausgang eines Oder-Gatters ist 1, wenn mindestens ein Eingang 1 ist. Das Nicht-Gatter dreht den Eingang um.

Mit Hilfe der Gatter können wir kompliziertere Funktionen bilden, z. B. $f(x, y) = \neg x \vee y$. Hier ist die Funktionstafel, schrittweise aufgebaut.

x	y	$\neg x$	$\neg x \vee y$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

Die ersten beiden Spalten enthalten die vier möglichen Kombinationen für x und y. Die dritte und vierte Spalten sind dann die Werte von $\neg x$ und $\neg x \vee y$.

- Gibt es für die Funktion f einen gebräuchlichen Namen?
- Geben Sie einen Ausdruck für die Funktion $x < y$ an. Sie ist 1, genau wenn x Null ist und y gleich 1 ist.

Lösung:

- Gebräuchliche Namen: kleiner gleich, logische Implikation (man setzt 1 mit wahr und 0 mit falsch gleich. Implikation bedeutet: wenn x wahr ist, dann ist auch y wahr, wenn x falsch ist, dann darf y beliebig sein.
- $(\neg x) \wedge y$.

Aufgabe 4 (*6 Punkte*)

Haben die Ausdrücke $\neg(x \vee y)$ und $\neg x \wedge \neg y$ immer den gleichen Wert? Geben Sie die Funktionstabellen für beide Ausdrücke an.

Haben die Ausdrücke $\neg(x \wedge y)$ und $\neg x \vee \neg y$ immer den gleichen Wert? Geben Sie die Funktionstabellen für beide Ausdrücke an.

Haben die Ausdrücke x und $\neg(\neg x)$ immer den gleichen Wert?

Haben die Ausdrücke $x \vee \neg x$ und 1 immer den gleichen Wert?

Haben die Ausdrücke $x \wedge \neg x$ und 0 immer den gleichen Wert?

Diese Gleichheiten sind unter dem Namen *Regeln von de Morgan* bekannt.

Vereinfachen Sie folgenden Ausdruck mit den Regeln von de Morgan: $\neg(\neg((\neg x \vee y) \vee y) \wedge (\neg x)$. Der erste Schritt ist

$\neg(\neg((\neg x \vee y) \vee y) \wedge (\neg x)) = \neg\neg((\neg x \vee y) \vee y) \vee \neg(\neg x)$ Negation eines Unds wird Oder der Negationen

Beachten Sie dabei, dass der äußerste Ausdruck die Gestalt $\neg(.... \wedge ...)$ hat.

Lösung: Ja, die Werte sind in beiden Fällen immer gleich. Hier kommen die Funktionstabellen. Die Inhalte der Spalten 7 und 9, 8 und 10, 1 und 11 sind gleich. In 12 steht die Konstante 1 und in 13 die Konstante 0.

(1)						(7)	(8)	(9)	(10)	(11)	(12)	(13)
x	y	$\neg x$	$\neg y$	$x \vee y$	$x \wedge y$	$\neg(x \vee y)$	$\neg(x \wedge y)$	$\neg x \wedge \neg y$	$\neg x \vee \neg y$	$\neg(\neg x)$	$x \vee \neg x$	$x \wedge \neg x$
0	0	1	1	0	0	1	1	1	1	0	1	0
0	1	1	0	1	0	0	1	0	1	0	1	0
1	0	0	1	1	0	0	1	0	1	1	1	0
1	1	0	0	1	1	0	0	0	0	1	1	0

$$
\begin{aligned}
\neg(\neg((\neg x \vee y) \vee y) \wedge (\neg x)) &= \neg\neg((\neg x \vee y) \vee y) \vee \neg(\neg x) && \text{Negation eines Unds wird Oder der Negationen} \\
&= ((\neg x \vee y) \vee y) \vee x && \text{doppelte Negationen heben sich auf} \\
&= \neg x \vee x \vee y \vee y && \text{Reihenfolge ist unwichtig bei mehrfachem Oder} \\
&= \neg x \vee x \vee y && y \vee y = y \\
&= 1 \vee y && x \vee \neg x = 1 \\
&= 1 && 1 \vee ... = 1
\end{aligned}
$$

Aufgabe 5 (*6 Punkte*) Die Eingabe eines Volladdierers besteht aus drei Bits x, y und z und die Ausgabe besteht aus zwei Bits c und s. Die Ausgabe ist die Binärdarstellung der Summe der Eingabebits:

$$2c + s = x + y + z.$$

Im Kurs sahen wir, dass man einen Volladdierer aus zwei Halbaddierern und einem Oder-Gatter aufbauen kann. Wir lernen nun eine alternative Realisierung kennen.

Das Summenbit s ist Eins, genau wenn $x+y+z \in \{1,3\}$, und das Carrybit c ist Eins, genau wenn $x+y+z \geq 2$.

Zur Erinnerung: $\{1,3\}$ bezeichnet die Menge, die aus den Zahlen 1 und 3 besteht. $\in$ liest man als *in* oder *ist Element der Menge* oder *gehört zur Menge* oder *liegt in der Menge*. Ausgeschrieben liest sich die erste Satzhälfte also als: Das Summenbit s ist Eins, genau wenn $x + y + z$ zur Menge $\{1,3\}$ gehört. Oder noch anders. Das Summenbit s ist Eins, genau wenn $x + y + 1$ entweder Eins oder Drei ist.

a) Sei $\mathit{MindestensZwei}(x,y,z) = 1$ genau wenn mindestens zwei der Variablen Eins sind. Dann ist $c = \mathit{MindestensZwei}(x,y,z)$. Argumentieren Sie, dass

$$\mathit{MindestensZwei}(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z).$$

b) Sei $\mathit{MindestensEine}(x,y,z) = 1$ genau wenn mindestens eine der Variablen Eins ist. Sei $\mathit{Alle}(x,y,z) = 1$ genau wenn alle Variablen Eins sind. Geben Sie Formeln für $\mathit{MindestensEine}$ und Alle an.

c) Zeigen Sie, dass $s = \mathit{Alle}(x,y,z) \vee (\mathit{MindestensEine}(x,y,z) \wedge \neg \mathit{MindestensZwei}(x,y,z))$.

d) Wie viele Gatter benötigt die Realisierung eines Volladdierers gemäß a) bis c). Wieviele Gatter braucht die Realisierung mit Hilfe von Halbaddierern. Warum könnte die Realisierung durch Halbaddierer vorzuziehen sein?

Lösung:

a) Einer der Klammerausdrücke ist Eins genau wenn die beiden darin enthaltenen Variablen Eins sind. Wir haben Klammerausdrücke für jedes der drei Paare. Die Klammerausdrücke sind mit Oder verbunden.

b) $MindestensEine(x, y, z) = x \vee y \vee z$ und $Alle(x, y, z) = x \wedge y \wedge z$.

c) s is Eins wenn $x+y+z \in \{1, 3\}$. Wir haben $x+y+z = 3$ genau wenn $Alle(x, y, z) = 1$ und $x+y+z = 1$, wenn $x + y + z \geq 1$ und $\neg(x + y + z \geq 2)$. Dieser Fall wird durch den zweiten Teilausdruck abgedeckt.

d) Ein Halbaddierer besteht aus drei Gattern mit zwei Eingängen und einer Negation. Damit braucht man für den Volladdierer sieben Gatter mit zwei Eingängen und zwei Negationen.

Die Realisierung aus dieser Übung braucht fünf Gatter für $MindestensZwei$, je zwei Gatter für $Alle$ und $MindestensEine$ und dann noch zwei Gatter und eine Negation für s. Dann sind 11 Gatter mit zwei Eingängen und eine Negation.

Die Realisierung mit Halbaddierern bracht weniger Gatter und könnte daher vorzuziehen sein.

Ich habe für die Videos, die Nachbereitung und das Übungsblatt etwa ☐ Stunden gebraucht.

(Ann-Sophie fertigt aus diesen Zahlen eine Statistik an. Kurt und Corinna sehen nur diese Statistik. Wir möchten wissen, ob der Schwierigkeitsgrad in etwa richtig ist.)

Rechner war spannend ☐ okay ☐ langweilig ☐
schwierig ☐ okay ☐ einfach ☐