**Technische Universität München**
**Institut für Informatik I-16**
**Dr. Tobias Lasser**
**Jakob Vogel**

**WiSe 2012/13**
**Lösungsblatt 3**
**5. November 2012**

# Algorithmen und Datenstrukturen

## Aufgabe 1 Boolsche Logik (Beispiellösung)

In dieser Aufgabe benutzen wir Wahrheitstabellen, die alle möglichen Eingabewerte explizit aufzählen. Dies ist wegen der kleinen Menge möglicher Werte (0/1, *wahr/falsch*, *true/false*) so möglich, und wir schreiben sie – eine Wertekombination pro Zeile – auf die linke Seite der Tabelle.

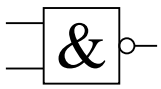
Auf der rechten Seite der Tabelle stehen dann verschiedene Spalten, wobei bei den einfachsten logischen Ausdrücken begonnen wird, und dann nach rechts immer komplexere, zusammengesetzte Ausdrücke untersucht werden. Dabei werden einfach die Eingaben der entsprechenden Zeile in den Ausdruck eingefügt.

Wenn die Gleichheit zweier logischer Ausdrücke bewiesen werden soll, wird der eben beschriebene Ablauf für beide Seiten des Gleichheitszeichen durchgeführt, und der Beweis ist erbracht, wenn die jeweils vollständigen Ausdrücke die gleichen Spalten ergeben.

a) Die Verknüpfung NAND (NOT-AND, $\uparrow$) ist eine boolsche Operation, aus der die Operationen NOT, AND und OR gebildet werden können. Die Wahrheitstabelle von NAND ist wie folgt:

| $a$ | $b$ | $a \uparrow b = \neg(a \wedge b)$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

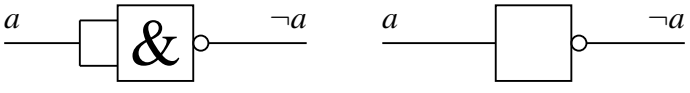
Als logisches Gatter in einer Schaltung verwendet man dann etwa folgendes Symbol, wobei das „&“ in dem Kasten die logische Operation AND[1] auf den Eingangssignalen (links) bezeichnet, und der kleine Kreis am Ausgangssignal (rechts) das dort stattfindende NOT:




Das Symbol für das AND-Gatter ist praktisch gleich, lediglich der Kreis fehlt! Ebenso kann das NOT-Gatter durch einen Kasten nur mit Kreis am Ausgang dargestellt werden, wobei dann natürlich nur ein Eingangssignal vorhanden ist.

Dementsprechend kann das NOT erzeugt werden als $\neg a = a \uparrow a$:

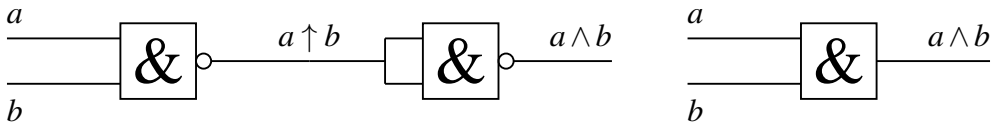
| $a$ | $a \uparrow a$ | $\neg a$ |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 0 | 0 |




[1] In diesem Fall ist das Symbol „&“ offensichtlich. Beim OR dagegen verwendet man „$\geq 1$“, da eines oder mehrere der Eingangssignale „1“ sein müssen.

Das AND ist ebenfalls einfach abzubilden als NOT-NAND, wobei das NOT natürlich wie eben gesehen auch durch NAND gebildet. Das ergibt $a \wedge b = \neg(a \uparrow b) = (a \uparrow b) \uparrow (a \uparrow b)$:

| $a$ | $b$ | $a \uparrow b$ | $(a \uparrow b) \uparrow (a \uparrow b)$ | $a \wedge b$ |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |




Das OR ist etwas komplexer, und wir nutzen das De-Morgan-Gesetz

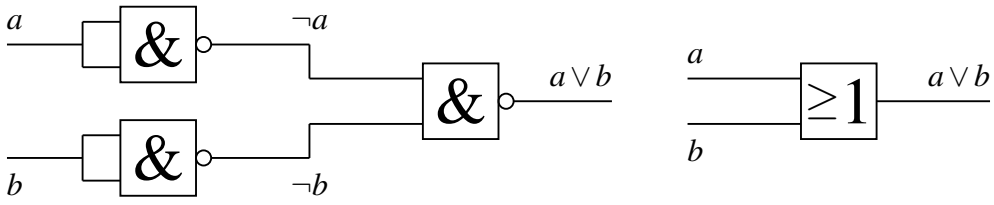
$$\neg(a \vee b) = (\neg a) \wedge (\neg b)$$

zur Herleitung. Daraus folgt durch Negation direkt, dass

$$\begin{aligned} a \vee b &= \neg((\neg a) \wedge (\neg b)) \\ &= (\neg a) \uparrow (\neg b) \\ &= (a \uparrow a) \uparrow (b \uparrow b)\,, \end{aligned}$$

und die entsprechende Wahrheitstabelle lautet:

| $a$ | $b$ | $a \uparrow a$ | $b \uparrow b$ | $(a \uparrow a) \uparrow (b \uparrow b)$ | $a \vee b$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |



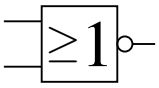

Offensichtlich lassen sich durch NAND-Gatter also die drei grundliegenden logischen Operationen (NOT, AND, OR) und darauf aufbauend alle komplexen logischen Operationen (XOR, Implikation, Äquivalenz, etc.; Addierer) aufbauen[2]. Ähnliches geht auch mit NOR-Gattern (NOT-OR, $\downarrow$) und man bezeichnet diese beiden Gatter folglich auch als Universal-Gatter.

Die Wahrheitstabelle für die den NOR-Gattern zugrunde liegende Operation ist, wie Sie bereits in der Vorlesung gesehen haben:

| $a$ | $b$ | $a \downarrow b = \neg(a \vee b)$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

[2] Eine Übersicht über alle Grundschaltungen finden Sie etwa in der englisch-sprachigen Wikipedia unter `http://en.wikipedia.org/wiki/NAND_logic` und `http://en.wikipedia.org/wiki/NOR_logic`, beide abgerufen am 4. November 2012.

Als Schaltungs-Symbol verwendet man dann folgendes Piktogramm, wobei das „$\geq 1$“ die logische Operation OR auf den Eingangssignalen bezeichnet, und der kleine Kreis am Ausgangssignal wieder das dort stattfindende NOT:




Die Abbildungen von NOT, OR (als NOT-OR) und AND (via De-Morgan $\neg(a \wedge b) = (\neg a) \vee (\neg b)$) sind sehr ähnlich zu den oben ausgeführten NAND-Versionen:

$$\begin{aligned} \neg a &= a \downarrow a \\ a \vee b &= \neg(a \downarrow b) \\ &= (a \downarrow b) \downarrow (a \downarrow b) \\ a \wedge b &= \neg((\neg a) \vee (\neg b)) \\ &= (\neg a) \downarrow (\neg b) \\ &= (a \downarrow a) \downarrow (b \downarrow b)\ , \end{aligned}$$

Bitte führen Sie zur Übung der Methode die entsprechenden Tabellen selbst aus!

b) Nun geht es darum, die Richtigkeit der bereits oben verwendeten De-Morgan-Gesetze zu beweisen. Wir betrachten also zunächst:

$$\neg(a \wedge b) = (\neg a) \vee (\neg b)$$

| $a$ | $b$ | $a \wedge b$ | $\neg(a \wedge b)$ | $\neg a$ | $\neg b$ | $(\neg a) \vee (\neg b)$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Da also die beiden Spalten für $\neg(a \wedge b)$ und $(\neg a) \vee (\neg b)$ gleich sind, also ist der Gleichheits-Beweis erbracht.

Bitte führen Sie zur Übung analoges für das andere De-Morgen-Gesetz $\neg(a \vee b) = (\neg a) \wedge (\neg b)$ selbst aus!

c) Für XOR (eXklusives OR, $\oplus$) wurde in der Vorlesung die Wahrheitstabelle

| $a$ | $b$ | $a \oplus b$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

und die Formel

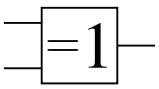
$$a \oplus b = \neg(a \wedge b) \wedge (a \vee b)$$

angegeben. Diese überprüfen wir nun mit einer Wahrheitstabelle:

| $a$ | $b$ | $a \wedge b$ | $\neg(a \wedge b)$ | $a \vee b$ | $\neg(a \wedge b) \wedge (a \vee b)$ | $a \oplus b$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 |

Da auch hier die beiden Spalten für $\neg(a \wedge b) \wedge (a \vee b)$ und $a \oplus b$ gleich sind, ist auch für diese Formel der Gleichheits-Beweis erbracht.

Man nutzt in Schaltplänen im Übrigen gerne folgendes Symbol für XOR-Gatter, wobei das „=1“ hier natürlich dafür steht, dass genau eines der beiden Eingangssignale auf 1 stehen muss:




**Aufgabe 2 Lazy Evaluation (Beispiellösung)**

*Die Lösung steht primär als Quellcode zur Verfügung, den Sie über die Website beziehen können. Dieser Text umfasst lediglich einige ergänzende Anmerkungen.*

Bei einer AND-Verknüpfung muss das Ergebnis *falsch* sein, wenn der erste Operand bereits *falsch* ist. Ebenso muss bei einer OR-Verknüpfung das Ergebnis *wahr* sein, wenn der erste Operand bereits *wahr* ist. Eine Prüfung des zweiten Operanden kann dann jeweils entfallen, was man als *„lazy evaluation“* bezeichnet.

Der Unterschied ist also der, dass die Operatoren `&&` und `||` diese verkürzende Auswertung benutzen, die Operatoren `&` und `|` dagegen immer beide Teile auswerten. Daher sollten Sie normalerweise in logischen Ausdrücken keine Funktionen aufrufen, wenn deren Ausführung (bzw. Nicht-Ausführung) Auswirkungen auf den Programmzustand („Seiteneffekte“) hat.

**Aufgabe 3 Objektorientierung II (Beispiellösung)**

*Die Lösung steht als Quellcode zur Verfügung, den Sie über die Website beziehen können.*