5. Ergänzungen zu booleschen Funktionen

Neben UND ($\cdot, \wedge$), ODER ($+, \vee$) und NOT ($\neg, ^-$) sind die

Implikation ($\Rightarrow$) mit

p	q	$p \Rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

und die

Äquivalenz ($\equiv$) mit

p	q	$p \equiv q$
0	0	1
0	1	0
1	0	0
1	1	1

mit die wichtigsten binären booleschen Funktionen.

Implikation und Äquivalenz werden auch sehr häufig in Beweisen, Argumentationen und Ableitungen benützt. Während die logische Bedeutung sich dabei nicht ändert, gibt es in der Notation jedoch ein paar Besonderheiten, die zu beachten sind.
Wir verwenden im Folgenden die folgende Präzedenz der angegebenen binären Operatoren:

$$\neg \text{ vor } \wedge \text{ vor } \vee \text{ vor } \Rightarrow \text{ vor } \equiv$$

Wenn wir dabei $p = q$ schreiben, bedeutet das, dass mit den gegebenen Axiomen und Ableitungsregeln (sowie Substitution) q aus p sowie p aus q abgeleitet werden kann.

Beispiel 26

$$p \wedge q \vee p \wedge \neg q \equiv p \;=\; (((p \wedge q) \vee (p \wedge (\neg q))) \equiv p\,.$$

In Beweisen verwenden wir häufig Ketten der Form

$$\begin{aligned} A_0 &\Rightarrow A_1 \\ &\Rightarrow A_2 \\ &\Rightarrow A_3 \\ &\equiv A_4 \\ &\vdots \\ &\Rightarrow A_n \end{aligned}$$

Diese Kette steht für

$$A_0 \Rightarrow A_1 \text{ und } A_1 \Rightarrow A_2 \text{ und } \ldots \text{ und } A_{n-1} \Rightarrow A_n \ ,$$

während für den booleschen Ausdruck

$$A_1 \Rightarrow A_2 \Rightarrow A_3 \equiv A_4 \equiv A_5 \Rightarrow A_6$$

keine eindeutige Klammerung definiert ist!

Es ist jedoch leicht aus der entsprechenden Wertetabelle zu sehen, dass $\equiv$ assoziativ ist, d.h.

$$(p \equiv q) \equiv r \;=\; p \equiv (q \equiv r)\;,$$

während Assoziativität für $\Rightarrow$ nicht gilt! $\Rightarrow$ ist auch nicht

- kommutativ (bzw. symmetrisch) (symmetrisch: aus $p \Rightarrow q$ folgt stets $q \Rightarrow p$)
- antisymmetrisch (antisymmetrisch: wenn $p \Rightarrow q$ und $q \Rightarrow p$, dann $p = q$)
- asymmetrisch (asymmetrisch: aus $p \Rightarrow q$ folgt stets $q \not\Rightarrow p$)

Sprachliche Umsetzung der Implikation

Die boolesche/materiale Implikation $p \Rightarrow q$ wird sprachlich ausgedrückt durch

- (schon) wenn p, dann q;
- p impliziert q;
- aus p folgt q;
- p ist hinreichend für q;
- p nur wenn q;
- q ist notwendig für p;
- $\neg q$ impliziert $\neg p$;
- ...

Einige formale Eigenschaften der Implikation

1. $$p \Rightarrow q \;=\; \neg p \vee q \;.$$

   Beweis durch Wertetabelle!

2. $$p \Rightarrow q \;=\; \neg q \Rightarrow \neg p \;,$$

   da (mit (1)) $\neg q \Rightarrow \neg p = q \vee \neg p = \neg p \vee q = p \Rightarrow q.$

3. $$p \Rightarrow q \;=\; \neg(p \wedge \neg q) \;,$$

   da (mit De Morgan)
   $\neg(p \wedge \neg q) = \neg p \vee \neg\neg q = \neg p \vee q = p \Rightarrow q.$

4

$$p \Rightarrow q \;=\; p \wedge q \equiv p \;,$$

da

$$\begin{aligned} p \wedge q \equiv p &= (p \wedge q \Rightarrow p) \wedge (p \Rightarrow p \wedge q) \\ &= (\neg(p \wedge q) \vee p) \wedge (\neg p \vee p \wedge q) \\ &= (\neg p \vee \neg q \vee p) \wedge (\neg p \vee p \wedge q) \\ &= \neg p \vee p \wedge q = (\neg p \vee p) \wedge (\neg p \vee q) \\ &= \neg p \vee q \\ &= p \Rightarrow q \;. \end{aligned}$$

5. $$p \Rightarrow q \;=\; \neg p \wedge \neg q \equiv \neg q \;;$$

   die Argumentation verläuft hier analog zum vorherigen Fall, da $\neg p \wedge \neg q \equiv \neg q \equiv \neg q \Rightarrow \neg p$.

6. $$p \Rightarrow (q \Rightarrow r) \equiv (p \Rightarrow q) \Rightarrow (p \Rightarrow r) \;.$$

   Übungsaufgabe!

7. $$p \Rightarrow (q \Rightarrow r) \equiv p \wedge q \Rightarrow r \;.$$

   Übungsaufgabe!

Ebenso lässt sich leicht zeigen:

8. $$p \Rightarrow 1 \;=\; 1\,,$$

   d.h. 1 ist Rechtsnullelement von $\Rightarrow$, und

9. $$1 \Rightarrow p \;=\; p\,,$$

   d.h. 1 ist Linkseinselement von $\Rightarrow$, sowie

10. $$p \Rightarrow 0 \;=\; \neg p$$

    (das Prinzip des Widerspruchsbeweises), und

11. $$0 \Rightarrow p \;=\; 1\,,$$

    und zwar für alle p! (Aus etwas Falschem kann man alles folgern!)

Eine wichtige Ableitungsregel ist schließlich der

modus ponens:

$$p \wedge (p \Rightarrow q) \Rightarrow q \ .$$

Kapitel III Automatentheorie

1. Begriff des Automaten

- der Automat ist ein abstraktes Modell eines sehr einfachen Computers
- der Automat verfügt über Eingabe- und Ausgabemöglichkeiten
- Eingabe und Ausgabe hängen durch den inneren Zustand des Automaten zusammen

2. Eigenschaften von Automaten

2.1 Ein- und Ausgabe

- ein Automat kann eingehende Informationen in Form von Symbolen an einem (oder mehreren) Eingängen verarbeiten
- die an den Eingängen erlaubten Symbole sind durch das Eingabealphabet bestimmt
- die interne Berechnung kann den internen Zustand des Automaten verändern
- das Ergebnis der internen Berechnung kann als Symbol ausgegeben werden
- alle möglichen Ausgabesymbole bilden zusammen das Ausgabealphabet

2.2 Innere Zustände

- die Abhängigkeit der Ausgabe alleine von der Eingabe ermöglicht keine Aufgaben, die ein "Gedächtnis" benötigen
- verschiedene innere Zustände eines Automaten erlauben eine Abhängigkeit der Ausgabefunktionen von vorhergehenden Ereignissen
- die Menge der internen Zustände kann als "Kurzzeitgedächtnis" angesehen werden
- der Zustand transportiert Informationen zwischen zwei zeitlich aufeinanderfolgenden Operationen
- einer der inneren Zustände muss als Startzustand definiert sein
- für besondere Automaten werden spezielle Zustände als Endzustände bezeichnet

2.3 Ausgabefunktion

- definiert das nach jedem Schritt auszugebende Zeichen
-
  - Moore-Automat: die Ausgabefunktion hängt nur vom inneren Zustand des Automaten ab
  - Mealy-Automat: die Ausgabefunktion hängt vom inneren Zustand und der aktuell gelesenen Eingabe (an allen Eingängen) ab

2.4 Zustandsübergangsfunktion

- abhängig von allen vorhandenen Eingängen sowie dem aktuellen inneren Zustand
- definiert den Zustand, in den nach Abschluss des Rechenschrittes gewechselt werden soll

2.5 Determinismus

- Determinismus garantiert die Eindeutigkeit des Zustandsübergangs, d.h. für jede Kombination von Eingangswerten gibt es maximal einen Zustandsübergang
- nicht-deterministische Automaten erlauben mehr als einen Zustandsübergang für ein und dieselbe Kombination von Eingangswerten, die Zustandsübergangsfunktion und die Ausgabefunktion werden dann zu Relationen
- jeder nicht-deterministische endliche Automat kann in einen deterministischen Automaten überführt werden (zeigen wir später)