Rechneraufbau und Rechnerstrukturen

Von
Walter Oberschelp,
Gottfried Vossen

10., überarbeitete und erweiterte Auflage

Oldenbourg Verlag München Wien

Prof. Dr. Gottfried Vossen lehrt seit 1993 Informatik am Institut für Wirtschaftsinformatik der Universität Münster. Er studierte, promovierte und habilitierte sich an der RWTH Aachen und war bzw. ist Gastprofessor u.a. an der University of California in San Diego, USA, an der Karlstad Universitet in Schweden, an der University of Waikato in Hamilton, Neuseeland sowie am Hasso-Plattner-Institut für Softwaresystemtechnik in Potsdam. Er ist europäischer Herausgeber der bei Elsevier erscheinenden Fachzeitschrift Information Systems.

Prof. Dr. Walter Oberschelp studierte Mathematik, Physik, Astronomie, Philosophie und Mathematische Logik. Nach seiner Habilitation in Hannover lehrte er als Visiting Associate Professor an der University of Illinois (USA). Nach seiner Rückkehr aus den USA übernahm er den Lehrstuhl für Angewandte Mathematik an der RWTH Aachen, den er bis zu seiner Emeritierung im Jahr 1998 inne hatte.

Bibliografische Information Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

Rosenheimer Straße 145, D-81671 München
Telefon: (089) 45051-0
www.oldenbourg-wissenschaftsverlag.de

Lektorat: Margit Roth
Herstellung: Anna Grosser
Umschlagkonzeption: Kraxenberger Kommunikationshaus, München
Gedruckt auf säure- und chlorfreiem Papier
Druck: Oldenbourg Druckerei Vertriebs GmbH & Co. KG
Bindung: R. Oldenbourg Graphische Betriebe Binderei GmbH

ISBN 3-486-57849-9
ISBN 978-3-486-57849-2

Kapitel 4

OBDDs und Komplexität

Schaltfunktionen besitzen als eine Standard-Darstellung, ihre Funktionstabelle. Da sich jede Schaltfunktion $f(x_1, \ldots, x_n) : B^n \to B^m$ aus m Booleschen Funktionen zusammensetzen lässt, kann man das Darstellungsproblem für Schaltfunktionen prinzipiell auf das für Boolesche Funktionen reduzieren. Wir besitzen außer der Tabellendarstellung noch eine Reihe anderer Darstellungsmöglichkeiten für Boolesche Funktionen: Mit Hilfe des Darstellungssatzes findet man eine Minterm- bzw. Maxterm-Darstellung (Abschnitt 1.3), ferner bietet die Ringsummendarstellung (Abschnitt 1.5) eine weitere Alternative. Hinzu kommen noch Darstellungsmöglichkeiten über Schaltnetze sowie die später zu diskutierenden PLAs.

In diesem Kapitel betrachten wir eine alternative Darstellung Boolescher Funktionen, welche auch in ganz anderen Bereichen der Informatik als der Theorie der Schaltwerke, z. B. in der Verifikation endlicher Automaten oder im so genannten *Model Checking*, Anwendung gefunden hat: geordnete binäre Entscheidungsdiagramme. Wir werden diese generisch herleiten und sodann durch zunächst heuristische Überlegungen so verbessern, dass sich die Darstellung der zugrunde liegenden Booleschen Funktion vereinfacht. Der Versuch, dieses Vereinfachen zu systematisieren, scheitert allerdings, denn es stellt sich heraus, dass wir hier auf inhärent schwierige Probleme stoßen. Da uns im letzten Kapitel auch bereits derartige Probleme begegnet sind, liegt es nahe zu fragen, ob es zwischen diesen möglicherweise ein Zusammenhang besteht. Diesen werden wir andeutungsweise herstellen.

4.1 Geordnete binäre Entscheidungs-Diagramme

Wir beginnen unsere Betrachtungen mit Überlegungen, wie sich auf einer diskreten Struktur wie dem Booleschen Körper B eine Art Differentiation einführen lässt und leiten daraus eine neue kompakte (und manipulierbare) Darstellung für Boolesche Funktionen her.

4.1.1 Boolesche Differentiation und Shannon-Entwicklung

Wir haben in Kapitel 1 erwähnt, dass $B = \{0, 1\}$ ein Körper mit zwei Elementen ist, wenn man $\dotplus$ und $\cdot$ als zweistellige Verknüpfungen erklärt, in dem 0 das Null-

und 1 das Einselement ist. Über dieser Struktur lassen sich Polynome bilden; man kann ferner den Begriff des Differentialquotienten übertragen und damit festlegen, was unter einer Booleschen Ableitung verstanden werden soll.

Wir werden in diesem Abschnitt durch Hinweise Analogien zur klassischen Analysis über dem Körper der reellen Zahlen aufzeigen. Wir führen im Folgenden einige Schreibweisen ein:

Definition 4.1 Sei $f : B^n \to B$ eine Boolesche Funktion und $a \in B$.

$$f(x_i/a) := f(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_n)$$

d. h. $f(x_i/a)$ entsteht aus f durch *feste* Belegung der Variablen x_i mit dem Wert $a \in B$. Für $a = 0$ heißt $f(x_i/a)$ auch der *negative*, für $a = 1$ der *positive Kofaktor* von f bzgl. x_i.

Beispiel 4.1 Sei $f(x_1, x_2, x_3) = \overline{x}_1 x_3 + x_2$. Dann gilt:

$$f(x_1/0) = \overline{0} \cdot x_3 + x_2 = x_3 + x_2,$$

$$f(x_1/1) = \overline{1} \cdot x_3 + x_2 = x_2$$

□

Damit können wir nun festlegen:

Definition 4.2 Sei $f : B^n \to B$ eine Boolesche Funktion. Dann heißt die Funktion

$$f_{x_i} : B^{n-1} \to B$$

definiert durch

$$f_{x_i} := f(x_i/0) \dotplus f(x_i/1)$$

die (partielle) Ableitung von f nach x_i.

Beispiel 4.2 Wir betrachten die aus Beispiel 1.13 bekannte Funktion f:

$$\begin{aligned}
f &= \overline{x}_1 x_2 x_3 + x_1 \overline{x}_2 x_3 + x_1 x_2 x_3 \\
f_{x_1} &= f(x_1/0) \dotplus f(x_1/1) \\
&= (\overline{0} x_2 x_3 + 0 \overline{x}_2 x_3 + 0 x_2 x_3) \dotplus (\overline{1} x_2 x_3 + 1 \overline{x}_2 x_3 + 1 x_2 x_3) \\
&= \overline{x}_2 x_3 \\
f_{x_2} &= \overline{x}_1 x_3 \\
f_{x_3} &= \overline{x}_1 x_2 \dotplus x_1 \overline{x}_2 \dotplus x_1 x_2 = \overline{x}_1 x_2 \dotplus x_1
\end{aligned}$$

□

Auffallend ist eine starke Analogie zu den Ergebnissen partieller Differentiationen in der klassischen Analysis. Der tiefere Grund hierfür ist ein enger Zusammenhang zwischen den Regeln der hier betriebenen diskreten Mathematik und der klassischen kontinuierlichen Infinitesimalrechnung, der bereits Newton aufgefallen war: Denkt man sich in der Definition von f_{x_1} einen fiktiven Nenner $1 = 1 \dotplus 0$ hinzu, so lässt sich die Boolesche Ableitung als Differenzenquotient lesen, der in der kontinuierlichen Analysis zum Differentialquotienten gemacht werden muss.

Wir notieren einige Eigenschaften Boolescher Ableitungen:

Satz 4.1 Sei $f : B^n \to B$ eine Boolesche Funktion. Dann gilt:

(a) Für alle $i = 1, \ldots, n$ gilt: f ist unabhängig von x_i, d. h. für jede Belegung von x_i liefert f den gleichen Wert genau dann, wenn $f_{x_i} \equiv 0$ ist.

(b) Die Ableitung von f ist gleich der Ableitung von $\overline{f}$, d. h. für $i = 1, \ldots, n$ gilt

$$f_{x_i} = (\overline{f})_{x_i}$$

(c) Die Ableitungen der konstanten Funktionen (in Tabelle 1.4 die Funktionen f_0 und f_{15}) sind konstant 0.

(d) Mehrfache Ableitungen können in beliebiger Reihenfolge gebildet werden, d. h.

$$(f_{x_i})_{x_j} = (f_{x_j})_{x_i} \quad \text{für} \quad i, j = 1, \ldots, n$$

Beweis: Wir zeigen hier nur (a) und überlassen den Rest dem Leser als Übungsaufgabe (vgl. Aufgabe 4.1).

(a) f ist unabhängig von x_i genau dann, wenn gilt:

$$f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

Zur Vereinfachung setzen wir $i = 1$.

„$\Rightarrow$": Sei f unabhängig von x_1. Dann gilt für alle $x_2, \ldots, x_n \in B$:

$$f(0, x_2, \ldots, x_n) = f(1, x_2, \ldots, x_n)$$

Hieraus folgt

$$\begin{aligned} 0 &= f(0, x_2, \ldots, x_n) \dotplus f(1, x_2, \ldots, x_n) \\ &= f(x_1/0) \dotplus f(x_1/1) \\ &= f_{x_1} \end{aligned}$$

„$\Leftarrow$": Wir zeigen die Behauptung in der logischen Kontraposition:
Sei f abhängig von x_1, so gibt es Werte $x_2, \ldots, x_n \in B$ mit

$$f(0, x_2, \ldots, x_n) \neq f(1, x_2, \ldots, x_n)$$

Für diese Werte folgt dann:

$$f_{x_1} = f(0, x_2, \ldots, x_n) \dotplus f(1, x_2, \ldots, x_n) = 1$$

Damit ist f_{x_1} nicht identisch Null. □

Die bisher behandelten Darstellungen einer Booleschen Funktion f verwenden im Allgemeinen viele Bausteine, aus denen sich f zusammensetzt. Für das praktische Arbeiten mit Booleschen Funktionen ist noch eine Darstellung, die so genannte *Shannon-Entwicklung*, von Bedeutung, bei der die Funktion als Summe zweier Bestandteile, und zwar von Unterfunktionen, dargestellt wird. Die Shannon-Entwicklung, welche de facto auf George Boole zurück geht, lässt sich mit den oben eingeführten Schreibweisen wie folgt angeben:

Satz 4.2 Jede Boolesche Funktion $f : B^n \to B$ ist wie folgt darstellbar:

$$f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) = x_i \cdot f(x_i/1) \dotplus \overline{x}_i \cdot f(x_i/0)$$

Beispiel 4.3 Wir betrachten die Funktion $f(x_1, x_2, x_3) = \overline{x}_1 x_3 \dotplus x_2$. Dann gilt:

$$\begin{aligned} f(x_1/0) &= \overline{0} x_3 \dotplus x_2 = x_3 \dotplus x_2 \\ f(x_1/1) &= \overline{1} x_3 \dotplus x_2 = x_2 \\ \Rightarrow f(x_1, x_2, x_3) &= x_1 x_2 \dotplus \overline{x}_1 (x_3 \dotplus x_2) \end{aligned}$$

□

Die Shannon-Entwicklung lässt sich als eine diskretisierte Version des aus der Analysis bekannten Taylorschen Entwicklungs-Satzes auffassen. Im einfachsten Fall einer einstelligen Booleschen Funktion lässt sich die Entwicklung

$$f(x) = x \cdot f(x/1) \dotplus \overline{x} \cdot f(x/0)$$

umschreiben zu

$$\begin{aligned} f(x) &= x \cdot f(x/1) \dotplus (1 \dotplus x) \cdot f(x/0) \\ &= f(x/0) \dotplus x \cdot (f(x/1) \dotplus f(x/0)) \\ &= f(x/0) \dotplus x \cdot f_x \\ &= f(x/0) \dotplus (x \dotplus 0) \cdot f_x \end{aligned}$$

Die in die Sprache der reellen Mathematik übersetzte Analogie lautet

$$f(x) = f(x_0) + (x - x_0) \cdot f_x$$

bzw.

$$f_x = \frac{f(x) - f(x_0)}{x - x_0}$$

und durch den Übergang $x \to x_0$ wird dann im Reellen für differenzierbare Funktionen eine korrekte Gleichung daraus, nämlich die Definition der reellen Ableitung.

Man beachte, das sich die Shannon-Entwicklung einer Booleschen Funktion rekursiv fortsetzen lässt, denn es gilt für $i \neq j$:

$$\begin{aligned} f(x_1, \ldots, x_n) &= x_i \cdot f(x_i/1) \dotplus \overline{x}_i \cdot f(x_i/0) \\ &= x_i \cdot (x_j \cdot f(x_i/1, x_j/1) \dotplus \overline{x}_j \cdot f(x_i/1, x_j/0)) \\ &\quad \dotplus \overline{x}_i \cdot (x_j \cdot f(x_i/0, x_j/1) \dotplus \overline{x}_j \cdot f(x_i/0, x_j/0)) \end{aligned}$$

Im letzten Beispiel gilt damit

$$\begin{aligned} f(x_1/1, x_2/1) &= 1 \\ f(x_1/1, x_2/0) &= 0 \\ f(x_1/0, x_2/1) &= 1 \\ f(x_1/0, x_2/0) &= x_3 \end{aligned}$$

und damit insgesamt

$$f(x_1, x_2, x_3) = x_1 x_2 \dotplus \overline{x}_1 x_2 \dotplus \overline{x}_1 \overline{x}_2 x_3$$

Der Leser prüfe selbst nach, dass diese Darstellung von f die gleiche Funktion ergibt wie die oben angegebene.

Die Darstellung einer Booleschen Funktion über die Shannon-Entwicklung macht es möglich, die Variablen $x_1, x_2, ..., x_n$ in irgendeiner Reihenfolge zu bearbeiten. Beim Differenzieren in unterschiedlichen Reihenfolgen können sich Darstellungen unterschiedlicher Einfachheit ergeben. Im Allgemeinfall muss man mit einem sehr hohen Darstellungsaufwand rechnen; für n-stellige Boolesche Funktionen kann dieser von der exponentiellen Größenordnung 2^n sein (was wir als nächstes demonstrieren), so dass sich hier schon für relativ kleine n ein effizientes Arbeiten verbietet. Wir werden nun aber ein Werkzeug vorstellen, welches neue Wege zum Auffinden einfacher Darstellungen eröffnet.

4.1.2 Entscheidungsbäume

Wir machen zunächst die Shannon-Entwicklung zur Grundlage einer graphischen Darstellung einer Booleschen Funktion f als binären Baum, in welchem die Wurzel f selbst repräsentiert, der linke Teilbaum der Wurzel den negativen Kofaktor von f nach einer Variablen x_i und der rechte Teilbaum den positiven Kofaktor von f nach x_i: diese Variable x_i wird in diesem Zusammenhang auch als *Testvariable* bezeichnet. Dieses Prinzip ist in Abbildung 4.1 gezeigt.

Offensichtlich lässt sich mit Hilfe dieser Darstellung ein binärer Baum entwickeln, welcher von Ebene zu Ebene, also mit zunehmender Entfernung von der Wurzel, jeweils eine weitere Testvariable betrachtet und diese Variable aus der Darstellung der gegebenen Booleschen Funktion in dem Sinne eliminiert, dass nach dieser Variablen verzweigt wird. Es sei bemerkt, dass sich ein „Test auf x_i" formal durch einen if-then-else-Operator beschreiben lässt (vgl. Aufgabe 4.2). Betrachtet man der Reihe nach alle Variablen $x_1, \ldots, x_n$ einer Booleschen Funktion $f : B^n \to B$, so erhält man einen Entscheidungsbaum zur Variablenmenge $\{x_1, \ldots, x_n\}$.

- dessen Wurzel die Funktion f repräsentiert,
- dessen Blätter (welche die unterste Ebene bilden) mit 0 oder 1 (in rechteckigen Kästchen) markiert sind und keine ausgehenden Kanten haben und
- dessen Nicht-Blatt-Knoten mit einer der Variablen markiert sind und genau zwei ausgehende Kanten haben.

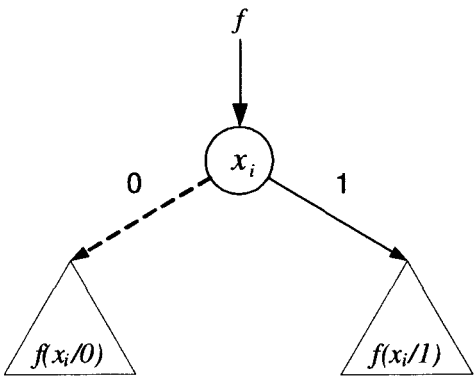


Abbildung 4.1: Baumdarstellung einer Booleschen Funktion anhand der Kofaktoren.

Intuitiv repräsentiert ein Pfad von der Wurzel zu einem Blatt eine konkrete Eingabe für die betrachtete Funktion bzw. eine Variablenbelegung und das Blatt selbst repräsentiert den zugehörigen Funktionswert.

Beispiel 4.4 Wir betrachten die durch die folgende Tabelle gegebene Boolesche Funktion f:

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Aus dieser Tabelle lässt sich leicht ermitteln:

$$\begin{aligned} f(x_1, x_2, x_3) &= \overline{x}_1 x_2 x_3 + x_1 \overline{x}_2 \overline{x}_3 + x_1 x_2 x_3 \\ &= (\overline{x}_1 + 1) x_2 x_3 + x_1 \overline{x}_2 \overline{x}_3 \\ &= x_2 x_3 + x_1 \overline{x}_2 \overline{x}_3 \end{aligned}$$

Die Vereinfachung der Funktion, die in den letzten beiden Zeilen angegebenen ist, wurde hier allein durch Anwendung der Rechenregeln der Booleschen Algebra ermittelt. Wir wollen als nächstes eine andere Methode beschreiben, solche Vereinfachungen herzustellen.

Abbildung 4.2 zeigt dazu zunächst einen Entscheidungsbaum, welcher f repräsentiert. Man beachte, dass in diesem Baum jeder Pfad von der Wurzel zu einem Blatt einen Minterm repräsentiert. Pfade, die in einem mit 1 markierten Blatt enden, repräsentieren sogar Minterme zu einschlägigen Indizes.

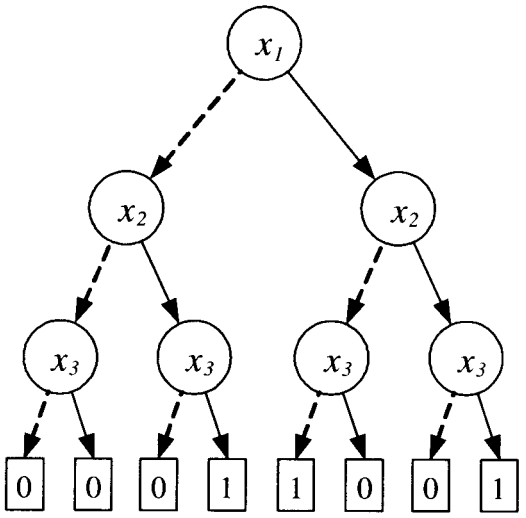


Abbildung 4.2: Darstellung der Funktion aus Beispiel 4.4 als Entscheidungsbaum.

Wir gehen bei der Vereinfachung schrittweise vor und legen zunächst Blätter zusammen. Es ist nämlich zu beachten, dass es lediglich zwei „Typen" von Blättern gibt: die mit 0 bzw. die mit 1 markierten. Wir erhalten als erste Vereinfachung den in Abbildung 4.3 gezeigten neuen Baum.

Wesentlich ist als nächstes die Beobachtung, dass es in diesem Baum identische Teilbäume gibt, beispielsweise die beiden (von den mit x_2 markierten Knoten) nach rechts abgehenden Teilbäume. Diese stehen beide für die Belegung $x_2 = 1$ und führen für gleiche x_3-Werte auf den gleichen Funktionswert. Deshalb legen wir die Teilbäume, die in x_2 beginnen und sich innerhalb der markierten Fläche befinden, zusammen mit dem in Abbildung 4.4 gezeigten Ergebnis (man beachte, dass diese Transformation auch bereits in dem in Abbildung 4.2 gezeigten Baum hätte angewendet werden können).

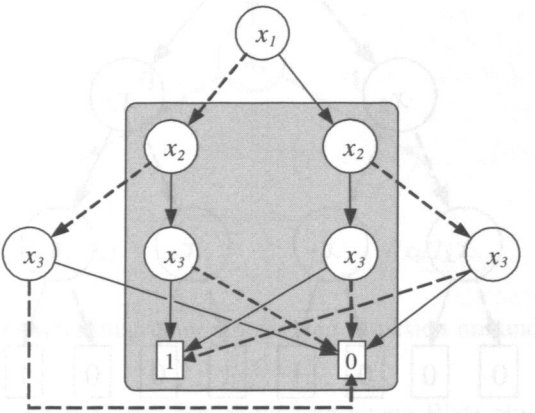


Abbildung 4.3: Baum aus Abbildung 4.2 nach Zusammenlegen der Blätter.

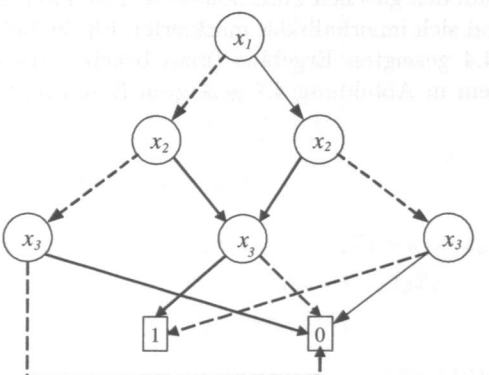


Abbildung 4.4: Baum aus Abbildung 4.3 nach Zusammenlegen identischer Teilbäume.

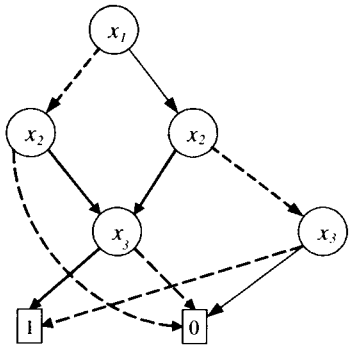


Abbildung 4.5: Baum aus Abbildung 4.4 nach Elimination des linkesten x_3-Teilbaums.

In dem so erhaltenen Baum stellen wir jetzt fest, dass der Funktionswert von f bei Belegung von x_1 und x_2 mit 0 von der Variablen x_3 nicht mehr abhängt, dass sich also $f = 0$ ergibt sowohl für $x_3 = 0$ als auch für $x_3 = 1$. Wir können den in Abbildung 4.4 gezeigten Baum also weiter vereinfachen und erhalten den in Abbildung 4.5 gezeigten Baum. □

Beispiel 4.5 Als weiteres Beispiel betrachten wir die durch die folgende Tabelle gegebene Boolesche Funktion f:

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Man erkennt hier, dass f im Fall $x_1 = x_2 = 0$ konstant den Wert 0 hat, im Fall $x_1 = 0, x_2 = 1$ und auch für $x_1 = 1$ von x_3 abhängt. Als SOP-Darstellung ergibt sich

$$f(x_1, x_2, x_3) = x_1 x_3 + x_2 x_3.$$

Den Entscheidungsbaum von f kann man auf Grund der gerade angestellten Überlegungen auf die in Abbildung 4.6 gezeigte kompakte Form bringen. Man beachte, dass sich die angegebene SOP-Darstellung aus dem Baum herauslesen lässt. □

4.1.3 OBDDs

Das letzte Beispiel liefert einen wichtigen Hinweis auf Anwendungen der bisher angestellten Überlegungen im Hinblick auf die Gewinnung einfacher Darstellungen gegebener Boolescher Funktionen. Viele Anwendungen von Schaltfunktionen erfordern heute

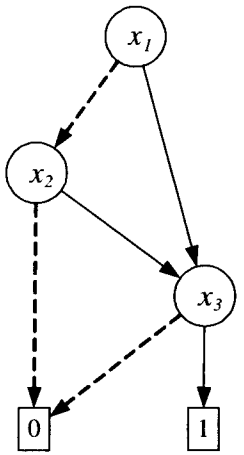


Abbildung 4.6: Darstellung der Funktion aus Beispiel 4.5.

Strategien zum Arbeiten mit äußerst unterschiedlichen Funktionstypen. Für diese ist die Entwicklung spezieller Verarbeitungstechniken (wie dies z. B. bei der Additionsfunktion geschieht) nicht mehr angezeigt. Die 1986 erstmals von R.E. Bryant in die Informatik eingeführten OBDDs (*Ordered Binary Decision Diagrams*) bieten eine (weitere) Darstellungsmöglichkeit für Schaltfunktionen, mit der sich in vielen Fällen effizient arbeiten lässt. Sie verzichtet völlig auf die Verwendung logischer Funktoren für Boolesche Funktionen und verwendet stattdessen als einziges Grundkonzept die oben eingeführte Entscheidung zwischen zwei Alternativen, zwischen 0 und 1. Da diese Darstellungen für viele unterschiedliche Typen von Booleschen Funktionen recht klein sind, bietet sich eine attraktiv erscheinende Alternative zur oft schwerfälligen und voluminösen „klassischen" Arbeitstechnik mit Funktoren an. Allerdings muss man hier vorsorglich vor zu viel Euphorie warnen: Voraussetzung für ein effizientes Manipulieren mit OBDDs ist — wie der Name bereits andeutet — die vorherige Vereinbarung einer „brauchbaren" Ordnung für die n Variablen. Bekanntlich kann man n Objekte auf $n! = n \cdot (n-1) \cdot (n-2) \ldots 3 \cdot 2 \cdot 1$ unterschiedliche Weisen anordnen, und mit wachsendem n wächst die Funktion $n!$ äußerst schnell (vgl. Beispiel 1.10). Trotzdem ist dieses Manko in vielen Fällen nur von akademisch-theoretischer Bedeutung, weil man eine gute Ordnung oft leicht erkennen kann. Die OBDDs stellen damit eine wichtige Ergänzung der bisher beschriebenen Möglichkeiten dar.

Definition 4.3 Ein OBDD zur Variablenordnung $x_1 < \ldots < x_n$ ist ein markierter, gerichteter, zykelfreier Graph (DAG) mit einem Startknoten (Wurzel) und zwei Endknoten (Blättern). Die beiden Blätter sind mit 1 bzw. 0 markiert, die anderen Knoten haben je zwei unterscheidbare Ausgänge 0 und 1 und sind mit Variablen derart markiert, dass bei jedem vollen Weg (d. h. einem Pfad, der von der Wurzel zu einem Blatt führt) die Reihenfolge der hierbei an den Knoten auftretenden Variablen verträglich mit der gegebenen Ordnung ist, d. h. die Indizes treten in der vorgegebenen Reihenfolge (evtl. mit Auslassungen) auf.

Beispiel 4.6 Wir betrachten die durch folgende Tabelle gegebene Boolesche Funktion f:

x_1	x_2	x_3	x_4	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Abbildung 4.7 zeigt ein OBDD für f zur Variablenordnung $x_1 < x_2 < x_3 < x_4$. Die durchgezogenen Linien bilden dabei wie vorher den 1-Ausgang, die unterbrochenen Linien den 0-Ausgang eines Knotens. Es ist nicht vorgeschrieben, z. B. den 1-Ausgang immer nach links abzuzweigen, wie es oft geschieht. Der hervorgehobene Weg bedeutet z. B., dass die dem Minterm $\overline{x_1}x_2\overline{x_3}x_4$ entsprechende Belegung 0101 den Wert 1 erzeugt. □

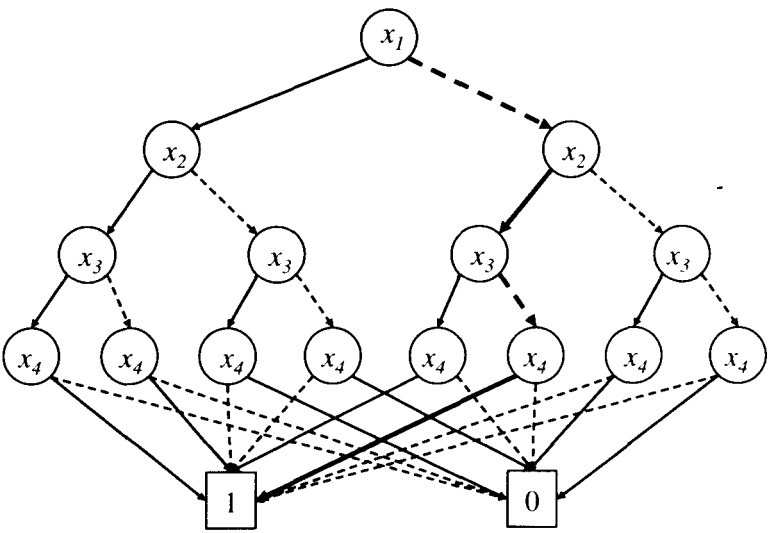


Abbildung 4.7: OBDD zu Beispiel 4.6 (zur Variablenordnung $x_1 < x_2 < x_3 < x_4$).

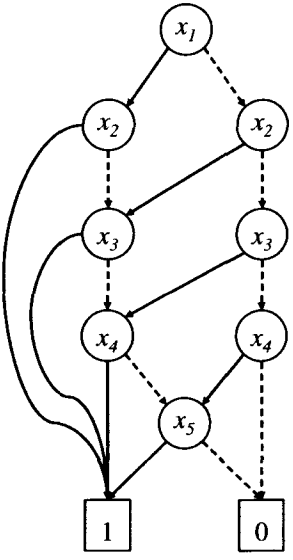


Abbildung 4.8: OBDD für die „Schwellenwert-Funktion" $f(x_1, x_2, x_3, x_4, x_5) = T_2^5$.

Man kann die Äquivalenz von OBDDs über die dargestellte Boolesche Funktion wie folgt fassen:

Definition 4.4 Ein OBDD stellt eine Boolesche Funktion f dar, wenn bei jedem vollen Weg die Variablenbelegung zu demjenigen Blatt führt, das in der Funktionstabelle von f festgelegt ist. Im OBDD fehlende Variablen können dabei übergangen werden.

Ein großer Vorzug von OBDDs ist z. B. die einfache Darstellung von symmetrischen Funktionen. Diese kommen in Anwendungen sehr häufig vor. Als Beispiel geben wir in Abbildung 4.8 ein OBDD für die „Schwellenwert-Funktion"

$$f(x_1, x_2, x_3, x_4, x_5) = T_2^5$$

an, welche genau dann den Wert 1 hat, wenn wenigstens 2 der 5 Variablen den Wert 1 haben. Ein weiteres Beispiel einer symmetrischen Funktion ist die „ungerade Parität", welche wie folgt definiert ist:

> $f(x_1, x_2, x_3, x_4) = 1$, falls die Anzahl der Einsen im Argument (x_1, x_2, x_3, x_4) gerade ist und 0 sonst.

Abbildung 4.9 zeigt ein OBDD für diese Funktion.

OBDDs haben eine Reihe von interessanten Eigenschaften, von denen wir hier auf einige kurz eingehen wollen. Die wichtigste ist, dass die verwendete Variablenordnung maßgeblich die Größe eines OBDD bestimmt. Als Beispiel betrachten wir die Funktion

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = x_1x_4 + x_2x_5 + x_3x_6 :$$

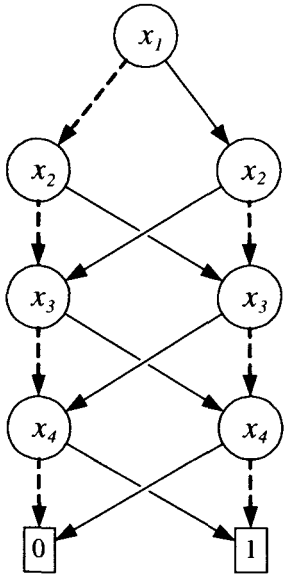


Abbildung 4.9: OBDD für die „Ungerade-Paritäts-Funktion".

Für die Variablenordnung

$$V_1 : x_1 < x_4 < x_2 < x_5 < x_3 < x_6$$

zeigt Abbildung 4.10 das zugehörige OBDD. Für die Variablenordnung

$$V_2 : x_1 < x_2 < x_3 < x_4 < x_5 < x_6$$

dagegen sieht das entsprechende OBDD wie in Abbildung 4.11 gezeigt aus.

Man erkennt an diesem Beispiel, dass sich ein OBDD nicht nur aus der Wahrheitstafel einer Booleschen Funktion, sondern auch aus einer gegebenen Funktionsdarstellung unmittelbar ableiten lässt. Darüber hinaus lassen sich OBDDs auch aus Schaltnetzen herleiten; eine Optimierung des OBDD, was wir als nächstes angehen wollen, führt dann auf ein verbessertes Schaltnetz.

4.2 Vereinfachung und Komposition von OBDDs

OBDDs erscheinen häufig keineswegs einfacher als die Darstellung einer Booleschen Funktion f über die DNF oder die Reed-Muller-Form. Es gibt aber Möglichkeiten, diese Darstellung zu vereinfachen. Wir geben zunächst zwei einfache Regeln an, mit denen man ein OBDD äquivalent umformen kann:

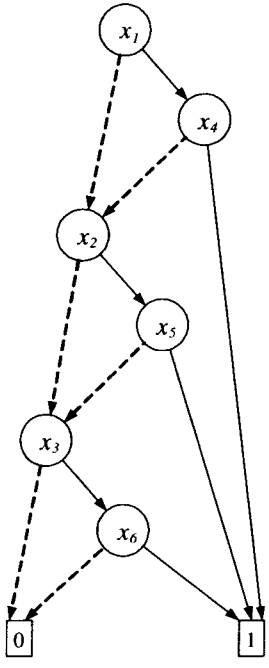


Abbildung 4.10: OBDD zur Variablenordnung V_1.

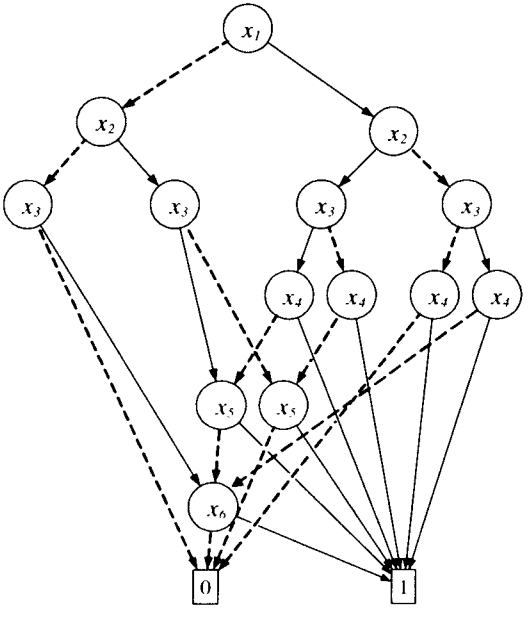
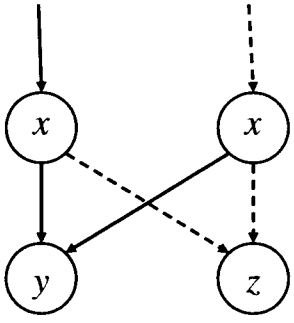
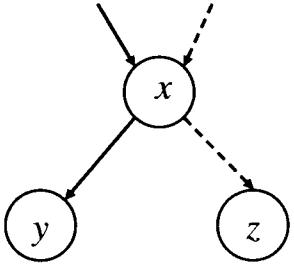


Abbildung 4.11: OBDD zur Variablenordnung V_2.

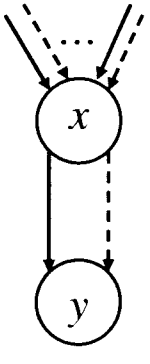
Regel 1 (Verjüngung oder 4-3-Regel):
Ist eine Teil-Konfiguration der Form



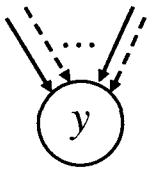
gegeben, so darf man die beiden Knoten zur Variablen x „verschmelzen" zu



Regel 2 (Elimination oder 2-1-Regel):
Ist eine Teil-Konfiguration der Form



mit beliebigen Eingängen bei x gegeben, so darf der Knoten x eliminiert werden, und die Eingänge nach x laufen direkt nach y:

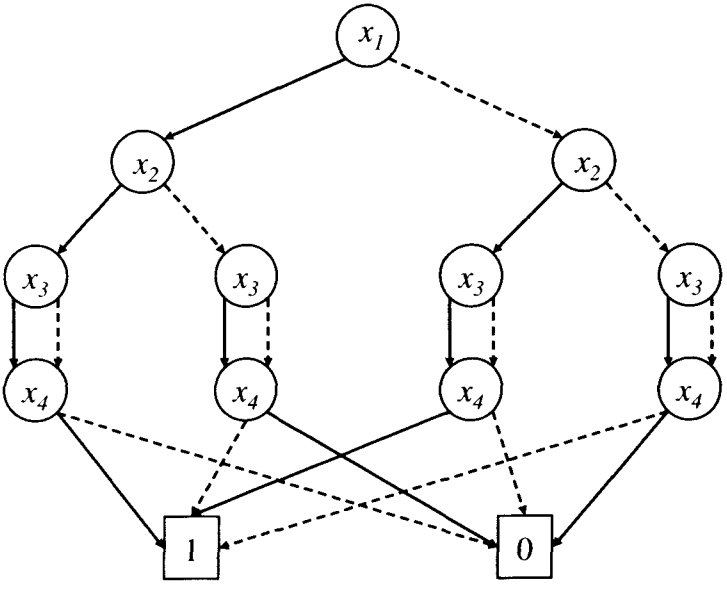


y und z dürfen bei beiden Regeln auch Blätter sein.

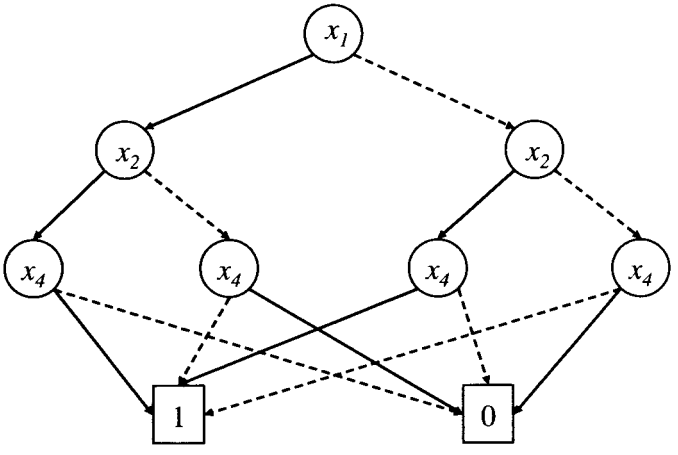
Es ist direkt einzusehen, dass die Anwendung irgendeiner dieser beiden Regeln die Funktionalität eines OBDD nicht ändert, da offensichtlich jede Belegung, die vor der Anwendung einer Regel zu einem Blatt führte, auch nachher zum gleichen Blatt führt; die Regeln überführen ein OBDD in ein äquivalentes OBDD.

Weniger offensichtlich als die Korrektheit der beiden Vereinfachungsregeln — aber umso bemerkenswerter — ist die Tatsache, dass durch sukzessives Anwenden dieser Regeln bis zum „Geht-Nicht-Mehr" ein reduziertes OBDD entsteht, das sogar bis auf Isomorphie eindeutig bestimmt ist. Diese Tatsache wird hier nicht bewiesen. Geht man beim Reduzieren systematisch von unten nach oben vor, so gilt ferner, dass man niemals zurück zu gehen braucht, um das reduzierte OBDD zu erhalten. Diese Tatsache garantiert die algorithmische Einfachheit des Reduktionsverfahrens und damit die Effizienz des Äquivalenztests für in gleicher Weise geordnete OBDDs. Hier liegt einer der entscheidenden Vorzüge gegenüber der Vereinfachungstechnik von Quine und McCluskey für Boolesche Funktionen, die mit der Funktorentechnik dargestellt werden. Man kann darüber hinaus zeigen, dass man auch für zwei in unterschiedlicher Weise geordnete OBDDs einen effizienten Äquivalenztest entwickeln kann.

Die Vereinfachungsregeln sollen nun auf das OBDD aus Abbildung 4.7 angewendet werden. Zunächst ist unten die Verjüngungs-Regel viermal anwendbar. Man erhält:



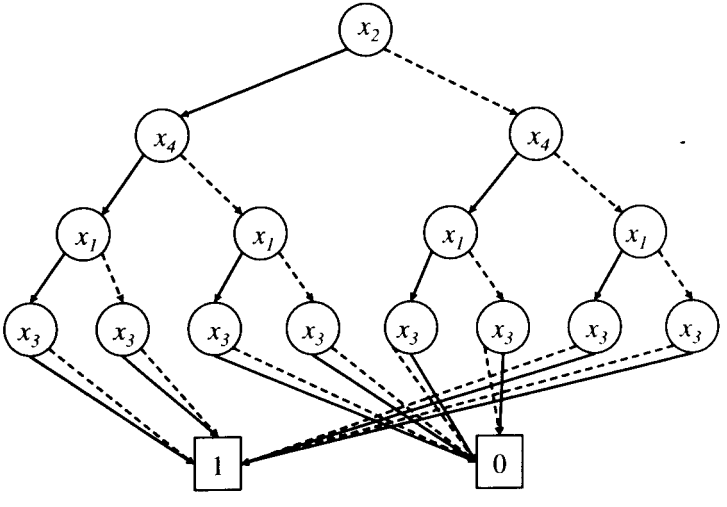
Nun können mit vier Anwendungen der Eliminations-Regel die Variablen x_3 entfernt werden:



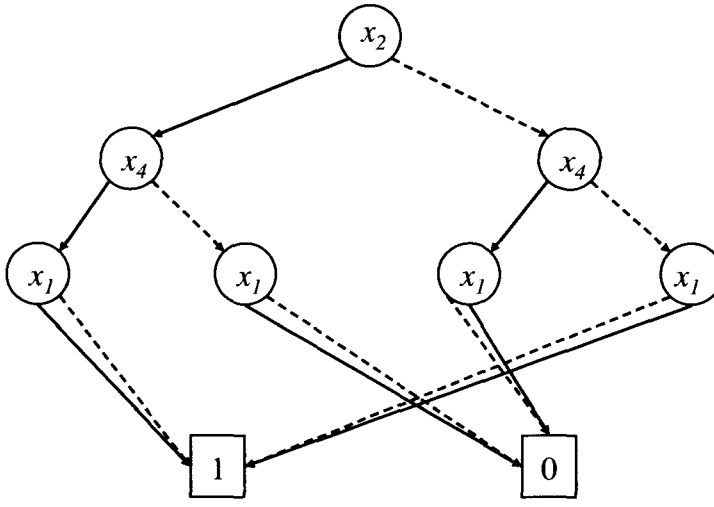
Das so erhaltene OBDD ist reduziert. Man kann aus ihm eine vereinfachte Darstellung von f, nämlich

$$x_1x_2x_4 + x_1\overline{x}_2\overline{x}_4 + \overline{x}_1x_2x_4 + \overline{x}_1\overline{x}_2\overline{x}_4$$

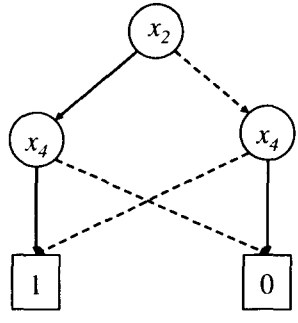
ablesen. Nun wählen wir eine andere Anordnung der Variablen, nämlich $x_2 < x_4 < x_1 < x_3$ und erstellen das zugehörige OBDD:



Hier kann man sofort unten achtmal x_3 eliminieren und erhält

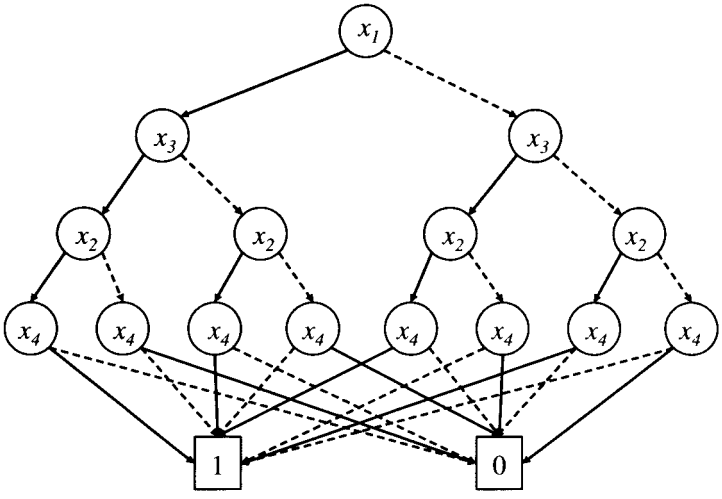


Nun ist noch weitere viermal die Eliminations-Regel anwendbar, und man erhält:



Dieses (reduzierte) OBDD beschreibt die Boolesche Funktion $x_2x_4 + \overline{x}_2\overline{x}_4$. Dies ist, wie wir in Beispiel 3.2 mit Hilfe der Karnaugh-Technik gesehen haben, eine einfachste Darstellung für f.

Wählt man schließlich die Reihenfolge $x_1 < x_3 < x_2 < x_4$, so ist bereits das zugehörige Ausgangs-OBDD irreduzibel. Es ist weder die Eliminations- noch die Verjüngungs-Regel irgendwo anwendbar:



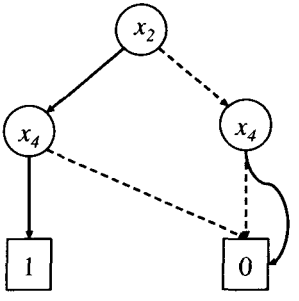
Man kann also, wie wir auch bereits am Ende des letzten Abschnitts gesehen haben, u.U. durch geschickte Anordnung der Variablen ein OBDD und damit die dadurch dargestellte Boolesche Funktion deutlich vereinfachen. Es gibt zwar effiziente Algorithmen zur systematischen Konstruktion und Manipulation von OBDDs, jedoch ist das Auffinden einer „optimalen" Variablenordnung schwierig und exakt sogar bis heute auf effiziente Weise nur mit Heuristiken lösbar. Es existieren sogar umfangreiche Software-Pakete, die das Arbeiten mit OBDDs unterstützen und die für manche klassische Probleme gute Lösungstechniken anbieten.

Besonders erwähnt sei hier die Verwendung von OBDDs im so genannten *Model-Checking*, das insbesondere mit der Logik der Berechnungsbäume CTL (*Computation-Tree Logic*) arbeitet. Hiermit können (möglicherweise unendlich lange) Zustandspfade und deren Eigenschaften beschrieben werden für Systeme, deren Entwicklungsperspektive durch baumartige Verzweigungen modelliert werden kann.

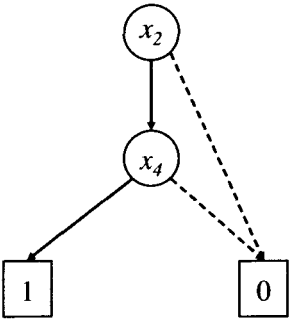
Zunächst sei hier die Erreichbarkeitsanalyse in der Automatentheorie genannt. Allgemeiner geht es im Model-Checking um die *Verifikation* von Implementationen für dynamische Systeme. Hier kann man auch an interagierende Systeme denken, die nach festgelegten Regeln (Protokollen) miteinander kommunizieren. Die CTL-spezifischen Ausdrucksmittel, mit denen insbesondere Eigenschaften von (zeitlichen) Folgezuständen, aber auch „Irgendwann"-Ereignisse auf Berechnungspfaden formuliert werden können, lassen sich auf natürliche Weise mit der OBDD-Technik symbolisieren. Der „Beginn" eines solchen Diagramms an der Wurzel und die Logik der Verzweigung stehen in enger Analogie zum Baum-Paradigma der CTL. Durch die Möglichkeit von Querverbindungen in einem OBDD und durch die Vereinfachungsregeln, die dabei das Auftreten von Zykeln ausschließen, kann die Modellierung von Bäumen (diese haben *keine* Querverbindungen!) modifiziert werden, ohne dass ein Verlust an Verifikations-Potenzial auftritt. Besonders diese sich in der Praxis häufig ergebende Möglichkeit, ein OBDD schmal zu halten, bietet bei der stets zu befürchtenden Zustands-Explosion für die Programm-Verifikation eine attraktive Perspektive. Methoden — und seien es auch nur Heuristiken — zur Erzeugung günstiger (verschlankender und vereinfachender) Variablen-Ordnungen können somit direkt eine Beschleunigung des korrespondierenden Model-Checkers bewirken.

Vereinfachung von Funktionen ist aber nicht das alleinige Ziel des Arbeitens mit OBDDs. Eine besonders wichtige Aufgabe ist das Zusammensetzen von Booleschen Funktionen aus einfacheren. Es soll hier angedeutet werden, dass es systematische Verfahren gibt, die Zusammensetzung (Komposition) zweier Funktionen mittels einer zweistelligen Verknüpfung „graphisch" vorzunehmen, wobei eventuell „nachreduziert" werden muss.

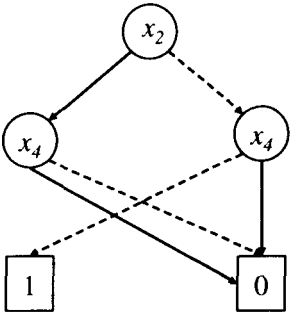
Als Beispiel für dieses Vorgehen betrachten wir die OBDDs für x_2x_4 und für $\overline{x}_2\overline{x}_4$ mit der Variablenordnung $x_2 < x_4$, welche zunächst reduziert werden und dann mit dem angedeuteten Verfahren (siehe oben) zusammengesetzt werden.
Zunächst $f = x_2x_4$:



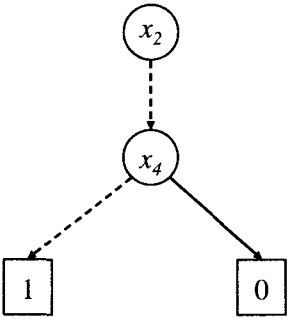
Dieses geht über in:



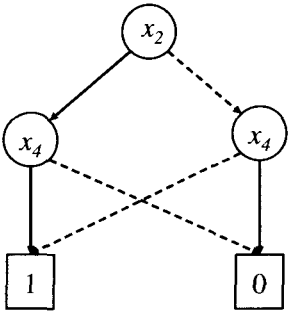
Ferner: $g = \overline{x}_2\overline{x}_4$



geht über in



Der (hier nicht explizit erläuterte) Zusammensetzungs-Algorithmus liefert für „+" als zweistellige Verknüpfung das OBDD



Zu Grunde liegt der aus der Shannon-Entwicklung nach x_2 folgende Zusammenhang

$$f \cdot g = x_2 f(1, x_4) \cdot g(1, x_4) + \overline{x}_2 f(0, x_4) \cdot g(0, x_4) = x_2(x_4 \cdot 0) + \overline{x}_2(0 \cdot \overline{x}_4).$$

Für „+" als Verknüpfung ergibt sich $x_2 x_4 + \overline{x}_2 \overline{x}_4$.

Man hat allgemein den 1-Ausgang von x_2 zu „koppeln" mit der „$\cdot$"-Verknüpfung der 1-Ausgänge von x_2 in f und in g, ferner den 0-Ausgang von x_2 mit der „$\cdot$"-Verknüpfung der 0-Ausgänge von x_2 in f und in g. Das hier zu implementierende Verfahren ist im Allgemeinen deutlich schneller als wenn man von der vollen Booleschen Expansion der Funktionen f bzw. g ausgeht und auf diesen die Verknüpfung bearbeitet. Im Zeitgewinn des Zusammensetzungs-Verfahrens liegt neben der effizienten Vereinfachung für gleich geordnete OBDDs einer der Hauptgründe für die Akzeptanz des OBDD-Paradigmas.

4.3 Überdeckungsmatrizen und Minimalüberdeckungen

Wir wollen uns nun zu einigen der im vorigen Kapitel sowie bisher in diesem Kapitel behandelten Verbesserungsprobleme einen etwas systematischeren Zugang verschaffen. Dazu machen wir zwei grundlegende Beobachtungen: